A Scalable Solution to Detect Microbursts

Jeronimo Bezerra, Italo Valcy, Renata Frez, Julio Ibarra

<jbezerra@fiu.edu>, <idasilva@fiu.edu>, <renata.frez@rnp.br>, <julio@fiu.edu>
Outline

- The Motivation
- The Challenge
- The Solution
- Conclusion & Future Work
The Motivation.
The Motivation

- AmLight is an international research and education network funded by NSF with a SLA-driven application

- The SLA (Service Level Agreement) is specific about the performance metrics

- Packet drops and bottlenecks caused by microbursts can lead to poor performance and SLA penalties

- Perfect motivation to build a new network monitoring solution to enable sub-second network visibility.
Running a small proof-of-concept of a microburst

- First: Let’s assess the challenge: Simulating a microburst
- A microburst traffic of 70Gbps for **40ms**
- And a constant 10Gbps background flow
  - Total: 80Gbps
- Two instances of the “current” NMS:
  - One gathering counters every 500 ms (default)
  - One gathering counters every 100 ms (5x faster)
- The results are provided in the figure.
  - None of the instances came close to **80 Gbps**
  - *Found the gap in our monitoring!*
The Challenge.
What is a microburst?

• Microbursts are sporadic bursts of traffic that occurs in _very short_ time-scales:
  • Most publications suggest a **millisecond** time-scale (1/1000\textsuperscript{th} of a second)
  • However most network monitoring solutions (commercial and open-source) work on a **second** time-scale
    • Extreme challenging to detect millisecond-based microbursts with traditional tools!

• Why should we monitor microbursts:
  • Microbursts deplete bandwidth, fill out buffers, and introduce delay, jitter, and **packet drops**
The Challenges of detecting a microburst

• How fast should we query network devices for interface/flow counters?
  • 60s? 30s? 1s? Sub-second? (assuming the network device supports these short intervals)
  • In our experiment, even every 100ms wasn’t enough to detect some microbursts
  • SNMP, Automation, API queries

• How small should be the interval or sample to export interface/flow counters?
  • 1:1000? 1:500?
  • sFlow, JTI

• Is disk space utilization a concern?
  • The more granular the measurement (smaller gathering interval or streaming frequency), more data must be stored.
  • To avoid excessive disk space consumption over time, retention policies delete “old” data (losing historical data) or create trends (losing accuracy)
How fast should we query network devices for interface/flow counters?

Sending 10GB of data over 10Gbps link: **8 seconds**

Let’s query a network interface’s counters every 1s, 15s, 60s, and 150s:
Is disk space utilization a concern?

A typical interface counter has 4 bytes for timestamp (8 bytes if nanosec is required) plus 8 bytes for the counter (octets or packets).

For bandwidth, we query for incoming and outgoing traffic, per bytes and per packets (at least).

<table>
<thead>
<tr>
<th>Interval</th>
<th># of queries in a day per monitored item</th>
<th>Amount of data collected in a day (in Bytes) for ONE monitored item</th>
<th>Amount of Data Collected in a DAY for AmLight for all monitored items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every 1s</td>
<td>86,400</td>
<td>1.04 Mbytes</td>
<td>69.97 Gbytes</td>
</tr>
<tr>
<td>Every 15s</td>
<td>5,760</td>
<td>0.07 Mbytes</td>
<td>4.6 Gbytes</td>
</tr>
<tr>
<td>Every 60s</td>
<td>1,440</td>
<td>0.01 Mbytes</td>
<td>1.2 Gbytes</td>
</tr>
<tr>
<td>Every 150s</td>
<td>576</td>
<td>0.007 Mbytes</td>
<td>0.5 Gbytes</td>
</tr>
</tbody>
</table>

More Accuracy -> More Disk Space, I/O, and CPU utilization
The Solution.
AmLight INT Collector 2.0 – Microburst edition

- Goal 1: Leveraging In-band Network Telemetry (INT) to detect microbursts
- Goal 2: No more static gathering interval (every $N$ milliseconds)
- Goal 3: Monitoring bandwidth utilization in a very short time interval but saving the bandwidth utilization counter only in case of pattern changes (user-defined)
The Solution

• Version 1.0:
  1. Reads counters from NIC every 500 ms
  2. Saves counters to disk

• Version 2.0:
  1. Reads counters from NIC every 20 ms
  2. Search for microbursts and pattern changes.
  3. If found, saves a summary to disk.
Algorithm 1 searches for *pattern changes*: when current bandwidth grows above a predefined $\beta$ factor (microburst factor) using the last $\Delta$ values as reference.

$\alpha = \text{loop interval to collect counters}$

$\gamma = \text{minimal bandwidth}$

$\beta = \text{growing factor}$

$\Delta = \text{number of previous measurements}$

$\theta = \text{microburst margin}$
Algorithm 1: Field Trial in Production

2022-10-09 17:10:00 to 17:32:30

\[ \alpha = 20 \text{ ms} \]
\[ \gamma = 14 \text{ Gbps} \]
\[ \beta = 4 \]
\[ \Delta = 3 \]
\[ \theta = 60\% \]
Innovation 1: Field Trial in Production [2]

<table>
<thead>
<tr>
<th>Start Time (UTC)</th>
<th>Duration (s)</th>
<th>Max BW (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2022-10-09T13:10:37.304385768</td>
<td>0.02</td>
<td>16.35</td>
</tr>
<tr>
<td>2022-10-09T13:10:37.400937960</td>
<td>0.02</td>
<td>17.44</td>
</tr>
<tr>
<td>2022-10-09T13:10:37.499991784</td>
<td>0.02</td>
<td>18.88</td>
</tr>
<tr>
<td>2022-10-09T13:10:37.598316288</td>
<td>0.02</td>
<td>19.01</td>
</tr>
<tr>
<td>2022-10-09T13:10:37.696891136</td>
<td>0.02</td>
<td>18.97</td>
</tr>
<tr>
<td>2022-10-09T13:10:37.795097088</td>
<td>0.02</td>
<td>18.91</td>
</tr>
<tr>
<td>2022-10-09T13:10:37.893028608</td>
<td>0.02</td>
<td>19.09</td>
</tr>
<tr>
<td>2022-10-09T13:10:37.992322792</td>
<td>0.02</td>
<td>18.66</td>
</tr>
<tr>
<td>2022-10-09T13:10:38.091456256</td>
<td>0.02</td>
<td>18.91</td>
</tr>
<tr>
<td>2022-10-09T13:11:51.925128192</td>
<td>0.04</td>
<td>37.87</td>
</tr>
<tr>
<td>2022-10-09T13:11:58.794430952</td>
<td>0.06</td>
<td>53.41</td>
</tr>
<tr>
<td>2022-10-09T13:12:01.507265768</td>
<td>0.04</td>
<td>41.48</td>
</tr>
<tr>
<td>2022-10-09T13:13:21.666561768</td>
<td>0.04</td>
<td>20.83</td>
</tr>
</tbody>
</table>
The adaptive approach is keep processing the bandwidth counters and dynamically deciding whether to save them.

Our strategy compares the observed current bandwidth $BW_{curr}$ to previous value $BW_{previous}$ to understand if bandwidth has varied “significantly” using a user-defined margin $BW_{Factor}$.

If bandwidth has not varied significantly, the strategy decreases or increases the interval via user-defined $DecreaseFactor$ and $IncreaseFactor$.

We consider that bandwidth varies when it increases or decreases more than a user-defined threshold, for instance, more than 15%.

$DecreaseFactor$ and $IncreaseFactor$ are key variables since they define how fast to react to bandwidth variation.

```
Algorithm 2: Adaptive Measurement Interval
1 while True do
2     $BW_{curr}$ ← GetCurrentBW();
3     $BW_{previous}$ ← GetPreviousBW();
4     if $BW_{curr}$ > ($BW_{previous}$ * $BW_{Factor}$) or $BW_{previous}$ > ($BW_{curr}$ * $BW_{Factor}$) then
5         if not (Interval − $DecreaseFactor$) < $MinInterval$ then
6             Interval ← Interval − $DecreaseFactor$;
7         else
8             if not (Interval − $IncreaseFactor$) > $MaxInterval$ then
9                 Interval ← Interval + $IncreaseFactor$;
10            SaveCounter($BW_{curr}$);
11            Sleep(Interval);
12        end
13     end
```
Algorithm 2: Field Trial in Production

- To evaluate our solution, we used two well-known image comparison algorithms: PSNR-B and SSIM.
  - Peak Signal-to-Noise Ratio (PSNR-B) and Structural Similarity Index (SSIM) are used for video quality analysis and image comparison
  - PSNR-B assigns higher values whenever images are more similar
    - > 37 – Excellent
    - 31 – 37 – Good
    - 25 – 31 – Acceptable
  - SSIM returns the similarity percentage
    - > 90% – Excellent
    - 77–89% – Good
    - 61 – 76% – Acceptable

<table>
<thead>
<tr>
<th>Time Frame</th>
<th>Parameters</th>
<th>PSNR-B</th>
<th>SSIM</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last 30</td>
<td>Setup 4</td>
<td>26.03</td>
<td>81.36%</td>
<td>61.64%</td>
</tr>
<tr>
<td>seconds</td>
<td>Setup 6</td>
<td>26.23</td>
<td>81.48%</td>
<td>64.38%</td>
</tr>
<tr>
<td></td>
<td>Setup 7</td>
<td>26.10</td>
<td>81.28%</td>
<td>64.38%</td>
</tr>
<tr>
<td>Last 30</td>
<td>Setup 4</td>
<td>34.88</td>
<td>97.92%</td>
<td>53.90%</td>
</tr>
<tr>
<td>days</td>
<td>Setup 6</td>
<td>23.48</td>
<td>92.28%</td>
<td>67.48%</td>
</tr>
<tr>
<td></td>
<td>Setup 7</td>
<td>33.70</td>
<td>99.24%</td>
<td>63.96%</td>
</tr>
</tbody>
</table>
Conclusion & Future Work

• Field trials were performed and demonstrated how effectively we detect microbursts in a production network, down to 20 milliseconds, and how efficiently we reduced storage space (above 60%) while preserving good levels of accuracy.

• Similar results can be achieved with port mirroring or fiber taps solutions!

• Future work:
  • Dynamic tuning the parameters
  • Expanding the INT collector to enable additional specialized monitoring functions (e.g., per-flow microbursts detection).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Commercial and OSS Tools</th>
<th>AmLight Innovations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitors Bandwidth Utilization</td>
<td>Full Support</td>
<td>N/A</td>
</tr>
<tr>
<td>Fixed Data Gathering Interval</td>
<td>Full Support</td>
<td>N/A</td>
</tr>
<tr>
<td>Adaptive Data Gathering Interval</td>
<td>Not Supported</td>
<td>N/A</td>
</tr>
<tr>
<td>Detects Bursts</td>
<td>Partial Support *</td>
<td>N/A</td>
</tr>
<tr>
<td>Captures beginning of bursts</td>
<td>Partial Support *</td>
<td>Full Support</td>
</tr>
<tr>
<td>Detects Microbursts</td>
<td>Not Supported</td>
<td>Full Support</td>
</tr>
<tr>
<td>Efficient Disk Space Consumption</td>
<td>Not Supported</td>
<td>N/A</td>
</tr>
</tbody>
</table>
A Scalable Solution to Detection Microbursts

Jeronimo Bezerra, Italo Valcy, Renata Frez, Julio Ibarra

<jbezerra@fiu.edu>, <idasilva@fiu.edu>, <renata.frez@rnp.br>, <julio@fiu.edu>
Recent Presentations/References

• Understanding the impact of network microbursts to science drivers - 07/07/2023
  - https://youtu.be/_wronGw48os
  - CI Engineering Lunch & Learn Series

• Detecting Network Microbursts at AmLight - 04/21/2023
  - https://youtu.be/1x-aVZTyviM
  - CI Engineering Lunch & Learn Series

• In-band Network Telemetry at AmLight - 03/18/2022
  - https://youtu.be/M6n_UZlhBQQ
  - CI Engineering Lunch & Learn Series

• Autonomic Network Architecture at AmLight - 02/25/2022
  - CI Engineering Lunch & Learn Series

• Deploying per-packet telemetry in a long-haul network - 11/21/2021
  - https://www.youtube.com/watch?v=lVtY7dP7UGs&t=2s
  - INDIS Workshop