AmLight ExP
Americas Lightpaths Express & Protect

CI Lunch and Learn / April 21st, 2023

Handling Microbursts @ AmLight – Part 1 of 2

Jeronimo Bezerra jab@amlight.net

Italo Valcy italo@amlight.net

Outline

- Part 1 – The challenge of detecting microbursts
  - What is a burst? When is a burst *micro*?
  - Detecting microbursts: what's the challenge?
  - A step back: some fundamentals about tools and protocols
  - The AmLight INT Collector 2.0: Our adaptive approach
- Part 2 – Why and when is detecting microbursts important?
  - When is a microburst an issue?
  - Use cases
  - Future

Introduction

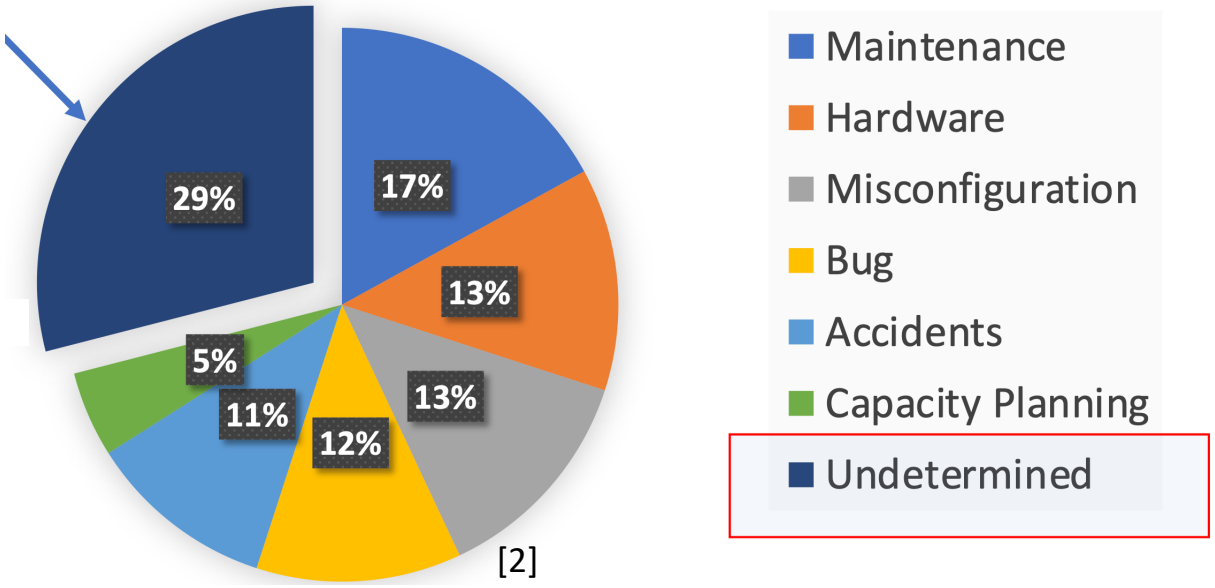
Paper [1], a study performed by Facebook, reported that 29% of incidents from 2011 to 2018 affecting its network infrastructure was considered "Undetermined"

Root-cause could not be determined:
1) Transient
2) Inability to correlate events

Examples of transient events: bursts and microbursts.

Let's talk about microbursts!

Data center incidents at Facebook from '11 to '18 [IMC '18]



[2]

[1] Meza, Justin, et al. "A large-scale study of data center network reliability." *Proceedings of the Internet Measurement Conference 2018*. 2018.
[2] Kannan, Pravein Govindan, et al. "Debugging Transient Faults in Data Centers using Synchronized Network-wide Packet Histories." *NSDI*. 2021.

What is a network burst?

A condition in which network activity rises suddenly for a short period of time. A burst is a transient elevation in network activity. [1]

Research and Education Networks (REN) tend to be *bursty* and bandwidth overprovisioning is the standard practice to handle bursts.

The duration and scale of a burst is based on one's experience: milliseconds vs seconds, above CIR, more than 50% of the interface bandwidth, 4-5x growth over previous Y seconds, etc.

[1] - The Network Encyclopedia - http://www.thenetworkencyclopedia.com/entry/burst/

What is a network burst? [2]

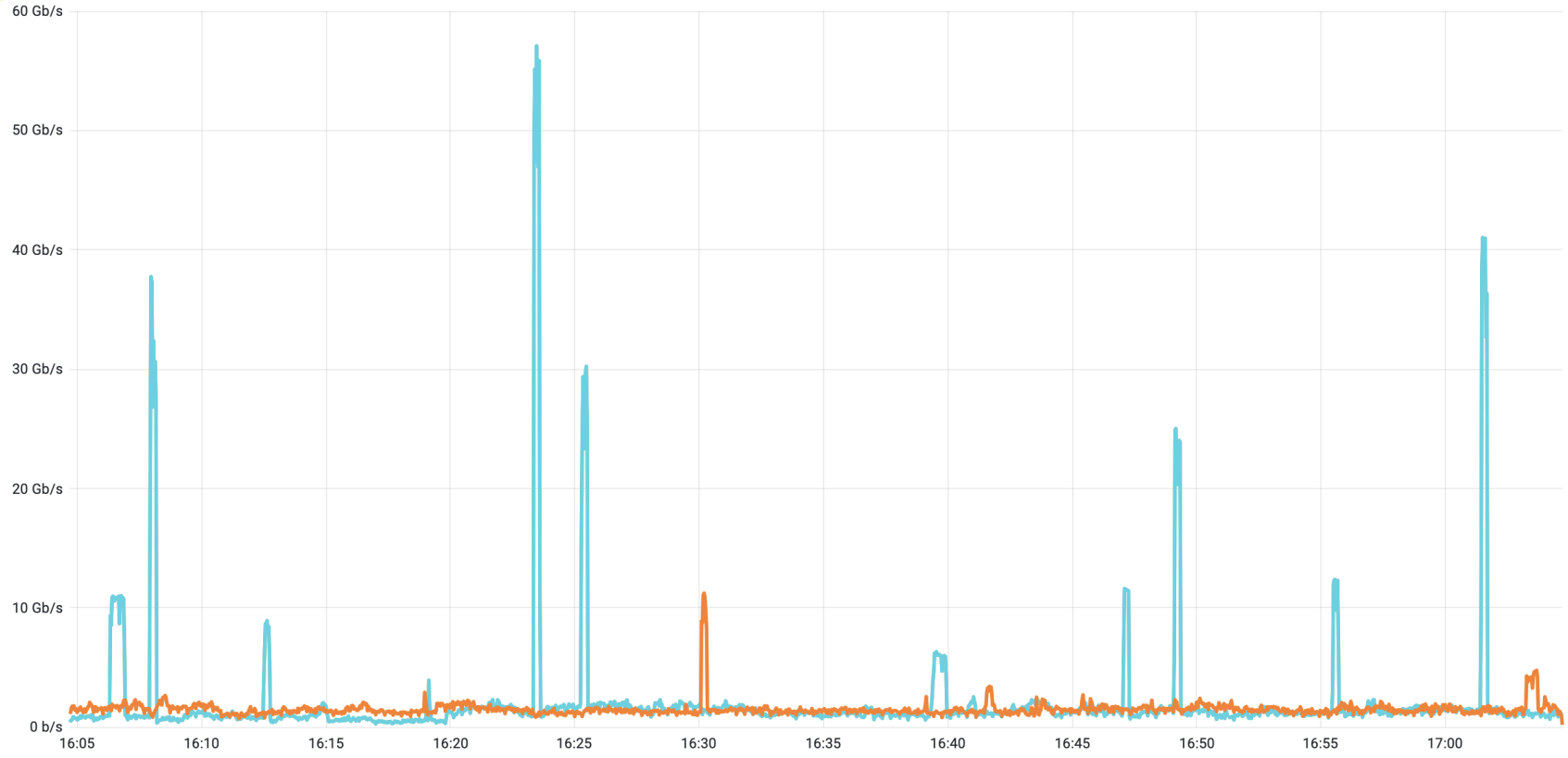
Why should we monitor bursts:

- Bursts can quickly deplete bandwidth
- Bursts can fill out queues and buffers
- And introduce delay, jitter, and drops

- Where to look for bursts:
  - Interface utilization
  - Queue utilization
  - Flow utilization

- Usually, network operators monitor counters/utilization every 15 to 300 seconds.
  - Lower the number, more granular the measurements -> higher CPU and disk space consumption

Examples of bursts



What is a network [micro]burst?

Network microbursts are sporadic bursts of traffic that occurs in *very short* timescales

"very short" varies per vendor and per author:

- Cisco and Facebook: In a **microsecond** time-scale
- Huawei, Arista, and most authors: In a **millisecond** time-scale
- Mine: In a time-scale my network monitoring system can't detect

Where to look for microbursts:

- Interface utilization
- Queue utilization
- Flow utilization

Some applications behave as microbursts, for example, IPTV, and distributed sensors.

Why is detecting a microburst complex?

- Control Plane and Data Plane synchronization timers
- How can it be done? Which technology to use?
- The ideal measurement gathering/query interval
- Handling disk space consumption

A step back: accessing network counters

- A little bit of background:
  - How are packets counted by network devices?
  - How fast can those counters be accessed?

How does a network device work?

Control Plane: Create/Maintain routing and switching tables

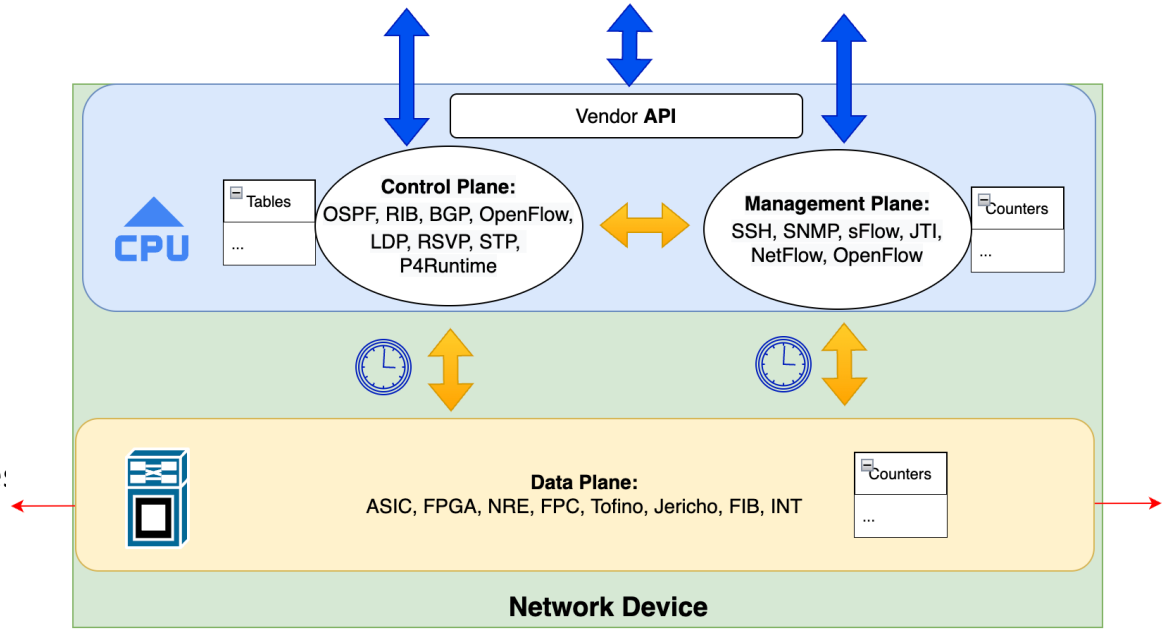
Management Plane: Operation, Administration, and Maintenance (OAM)

Data Plane: Forward packets at line rate

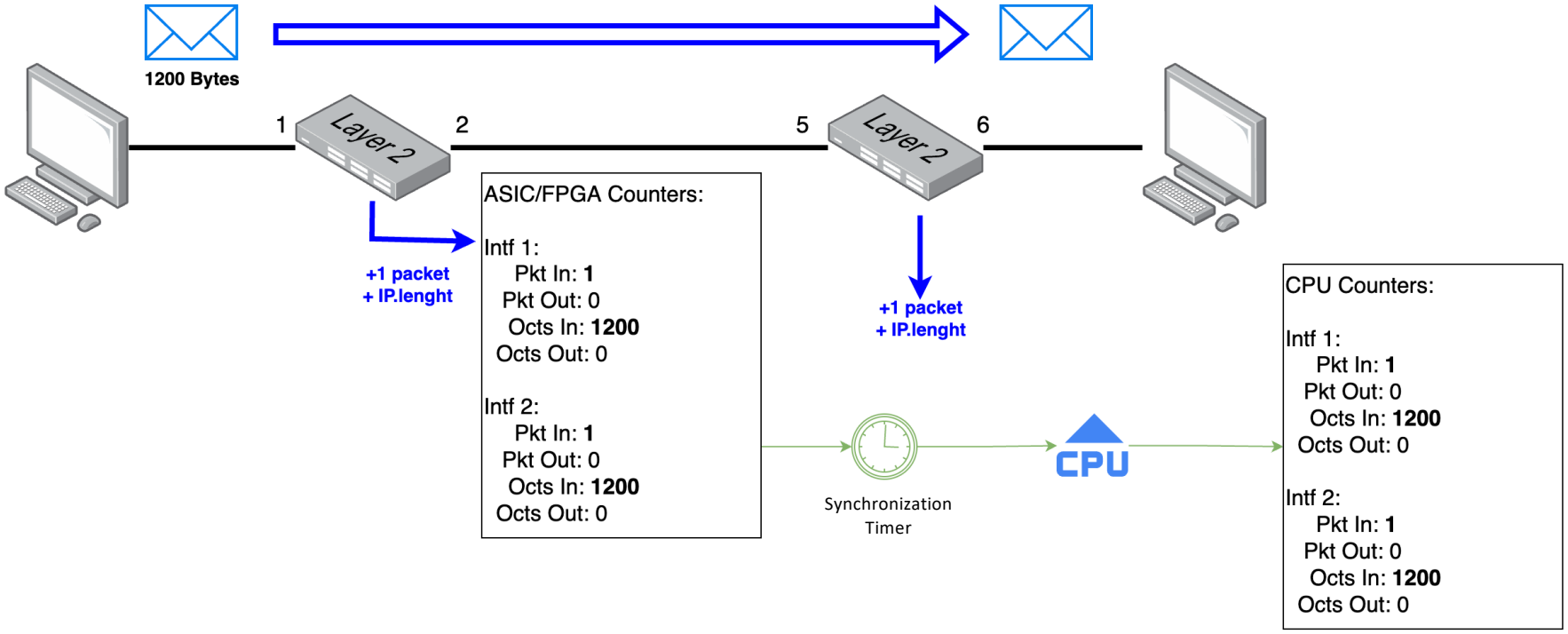
Interfaces/APIs for external communication systems (blue)

Internal channels for communication between planes (orange)

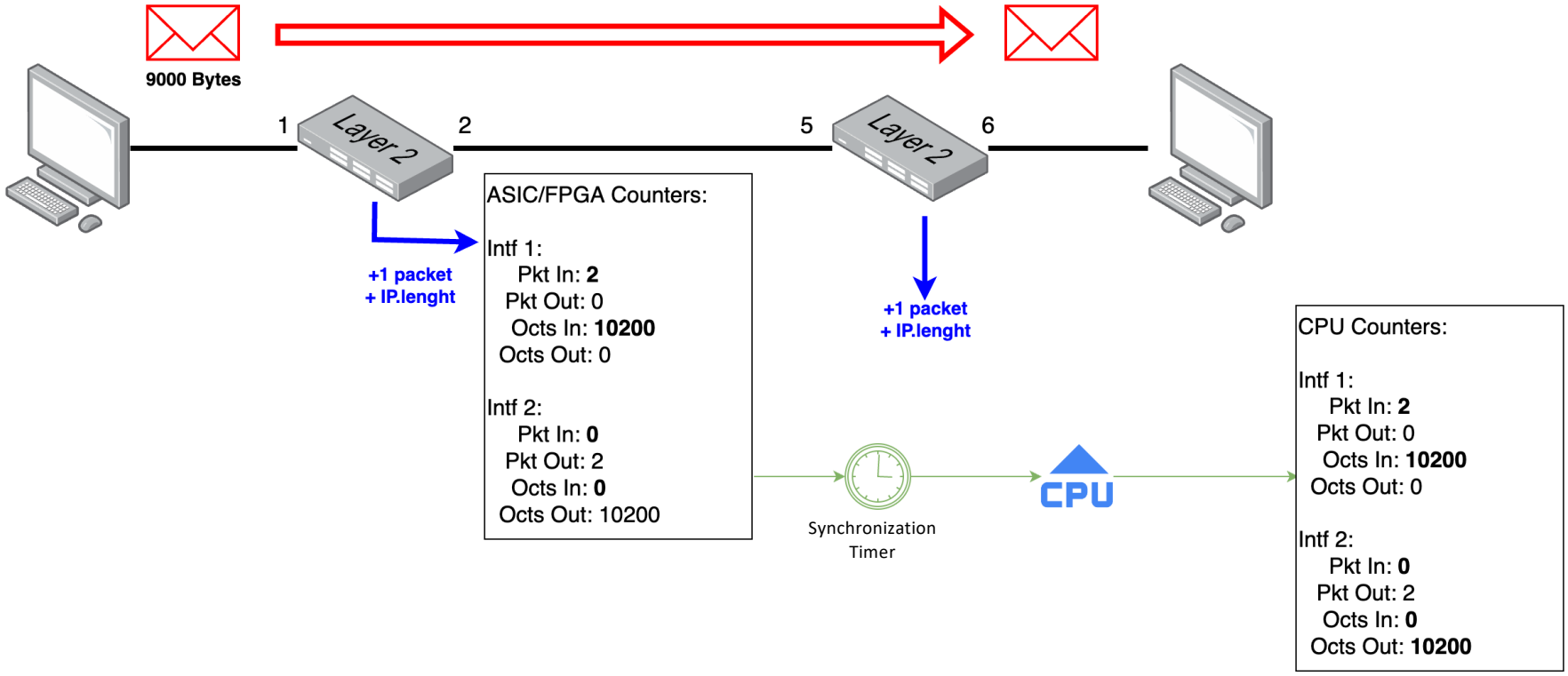
Timers for internal updates between planes



Counting packets

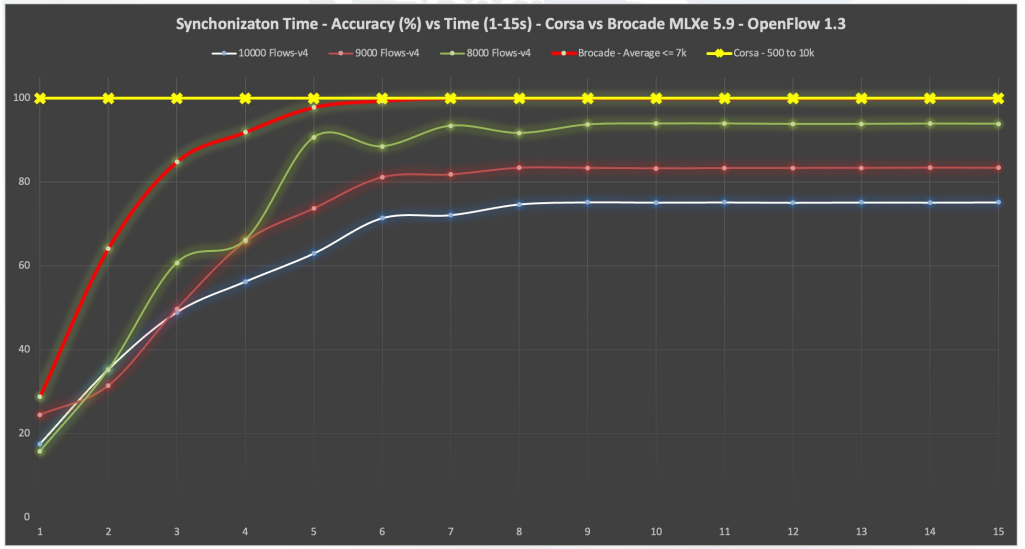


Counting packets [2]



Synchronization Timers

- How fast can that data be accessed?
  - Internal Timers control how often tables in the Control and Management Plane are synchronized with tables in the Data Plane
  - With smaller intervals more accurate measurements are possible but with possible impact to the device's CPU
- The synchronization timers usually goes from 1 to 20 seconds depending on the vendor (HP, Dell, Extreme, Juniper) and part number
  - Disclaimer: we never tested sub-second.
- Conclusion: shorter gathering intervals only makes sense if they are greater than the device's synchronization timers
  - Otherwise, you will have duplicated values
  - Meaning, no gains but with pains



Retrieving network counters

- Network Monitoring can leverage two approaches:

- Passive:
  - Operator REQUESTS counters from the network device
  - Gathering Interval defines the granularity/quality of the measurement
  - Gathering Interval is customized by the network operator (from 1 to many seconds)
  - Operator can request only the counters of interest

- Active/Streaming
  - Operator RECEIVES counters from the network device per trigger
  - Trigger can be a timer or an event (packet seen, drop because of an ACL)
  - The network monitoring system must be ready to RECEIVE network counters
  - Operator can receive multiple counters per message
  - *Ideal for microbursts detection*

Retrieving network counters [2]

- SNMP / Passive monitoring:
  - Well consolidated, used by most network operators
  - Several counters available via OIDs defined in MIB
  - Vendors can create their own MIBs
  - Used primarily to monitor incoming and outgoing interface utilization
  - IETF standard

- sFlow / Streaming monitoring:
  - Sampling approach, usually 1 every 4096 packets
  - Network device sends a sFlow record to a remote sFlow collector
  - Exports packet and device's metadata
  - Used primary to generate flow reports, similarly to NetFlow
  - IETF standard

Retrieving network counters [3]

- Juniper Telemetry Interface (JTI) / Streaming monitoring:
  - Each group of sensors export JTI reports to a JTI collector
  - JTI reports are sent every 1-**2** seconds
  - Offers consolidated data for each interval, including queue utilization
  - Great replacement for SNMP in Juniper environments
  - Proprietary
  - All JTI reports are stored in a database

- In-band Network Telemetry (INT) / Streaming monitoring
  - Every single packet triggers an INT report
  - INT reports are exported right away
  - Offers highest granular data possible
  - INT reports provide data and device's metadata
  - P4.org standard
  - Our implementation: Counters are gathered every 100 milliseconds to then be stored.

Retrieving network counters [4]

Tool/Framework	Accuracy depends on:	Challenges:	Used for:
SNMP	➢ Data Plane counters collection interval. ➢ SNMP collector polling.	➢ Low interval ➔ higher CPU utilization. ➢ High interval ➔ lower accuracy.	➢ General monitoring.
sFlow	➢ Sampling rate.	➢ Low sampling rate ➔ more storage required ➔ higher CPU utilization. ➢ High sampling rate ➔ lower accuracy.	➢ Troubleshooting unusual events. ➢ TOP N reports.
Juniper Telemetry Interface (JTI)	➢ Data sending interval.	➢ Low interval ➔ more storage required. ➢ High interval ➔ lower accuracy.	➢ Environments that require more granular information.
In-band Network Telemetry (INT)	➢ Real time. Complete visibility.	➢ Processing all data collected in real time.	➢ Troubleshooting short-time events.

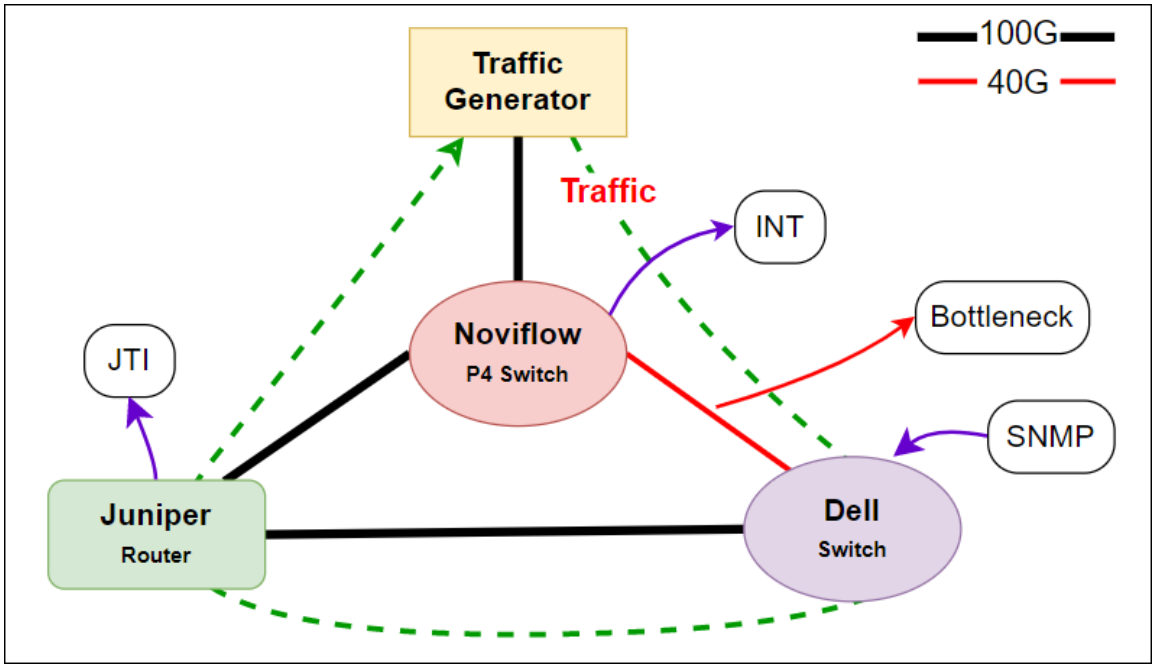
[2]

[2]- Evaluating INT, JTI, and sFlow @ AmLight – 2022 Internet2 TechEx

Retrieving network counters [5]

- Port Mirror/Fiber Taps
  - Not a protocol but an approach
  - All packets flowing through an interface or fiber can be captured by the operator
  - Packets captured can become SNMP counters, sFlow reports, and just locally inspected
  - Useful for troubleshooting and intrusion-detection systems (IDS)
  - Hard to scale, high CAPEX, and high OPEX

Comparing tools and protocols to detect microbursts

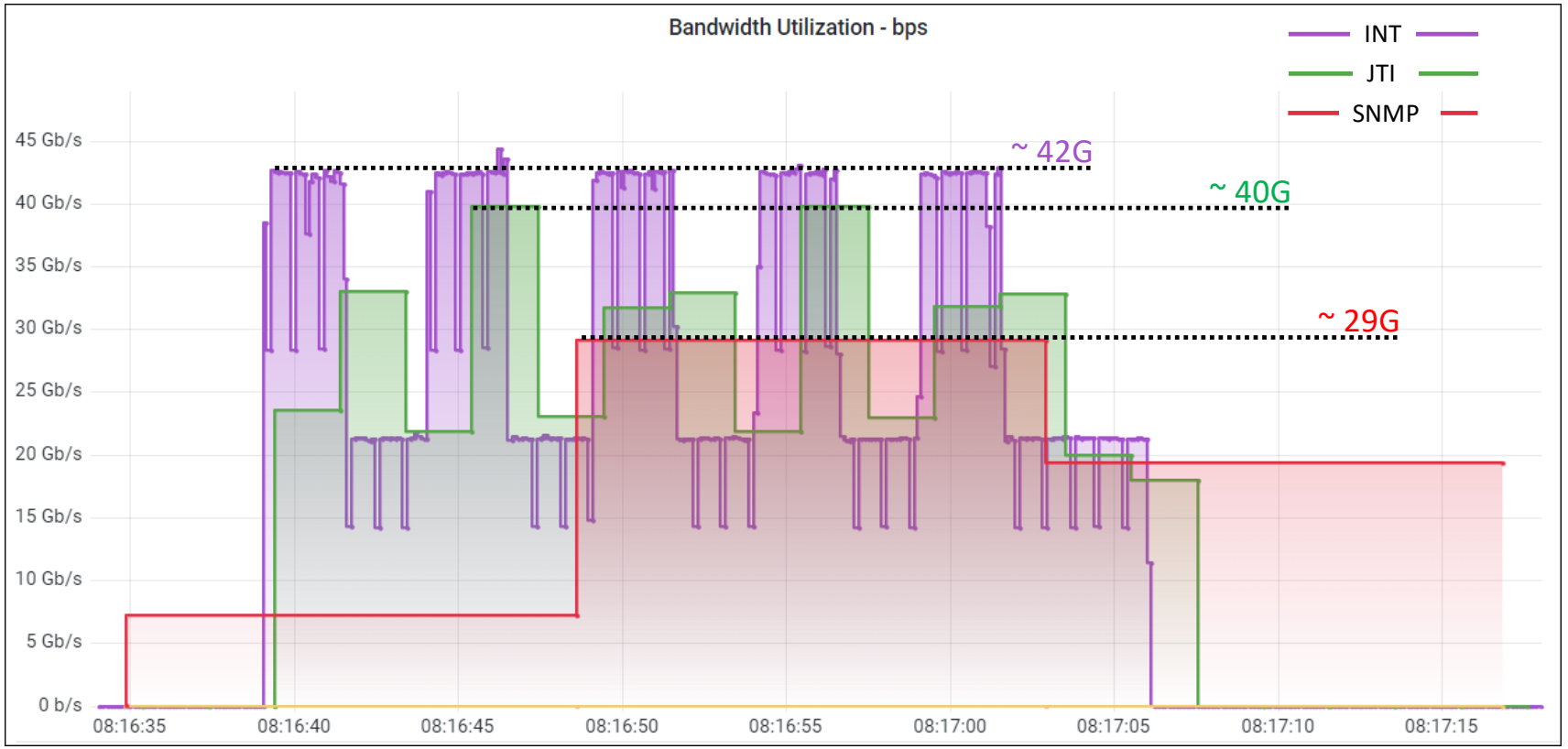


- EXFO Packet Generator
- Dell Z9100 = SNMP polling every **14s** (lowest possible value).
- Juniper MX204 = JTI enabled sending telemetry every **2s**
- Edgecore/Noviflow (P4 Switch) = INT enabled for all packets, i.e., **real-time**. Counters gathered and saved every **100ms** (experimentation purpose. We use 500 ms in production). Used AmLight INT Collector 1.1b

Slides from Evaluating INT, JTI, and sFlow @ AmLight – 2022 Internet2 TechEx

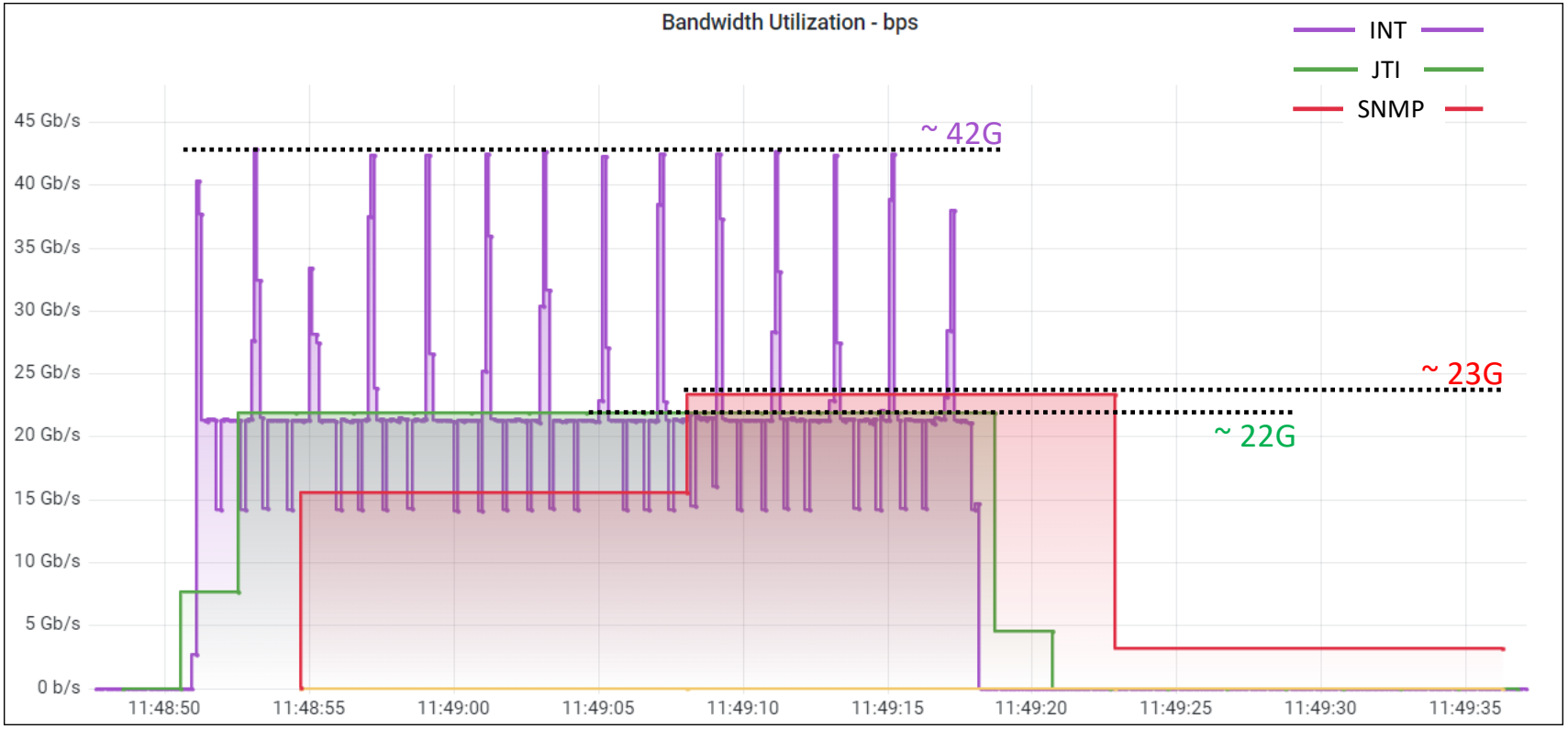
Identifying Bursts: SNMP x JTI x INT [Test 1]

- Interval: 30s.
- 2 flows: Continuous and Burst.
- Continuous Traffic: 20G.
- Burst: 5 x 30Gbps
- Burst duration: 2.5s.
- Interval between bursts: 2.5s.



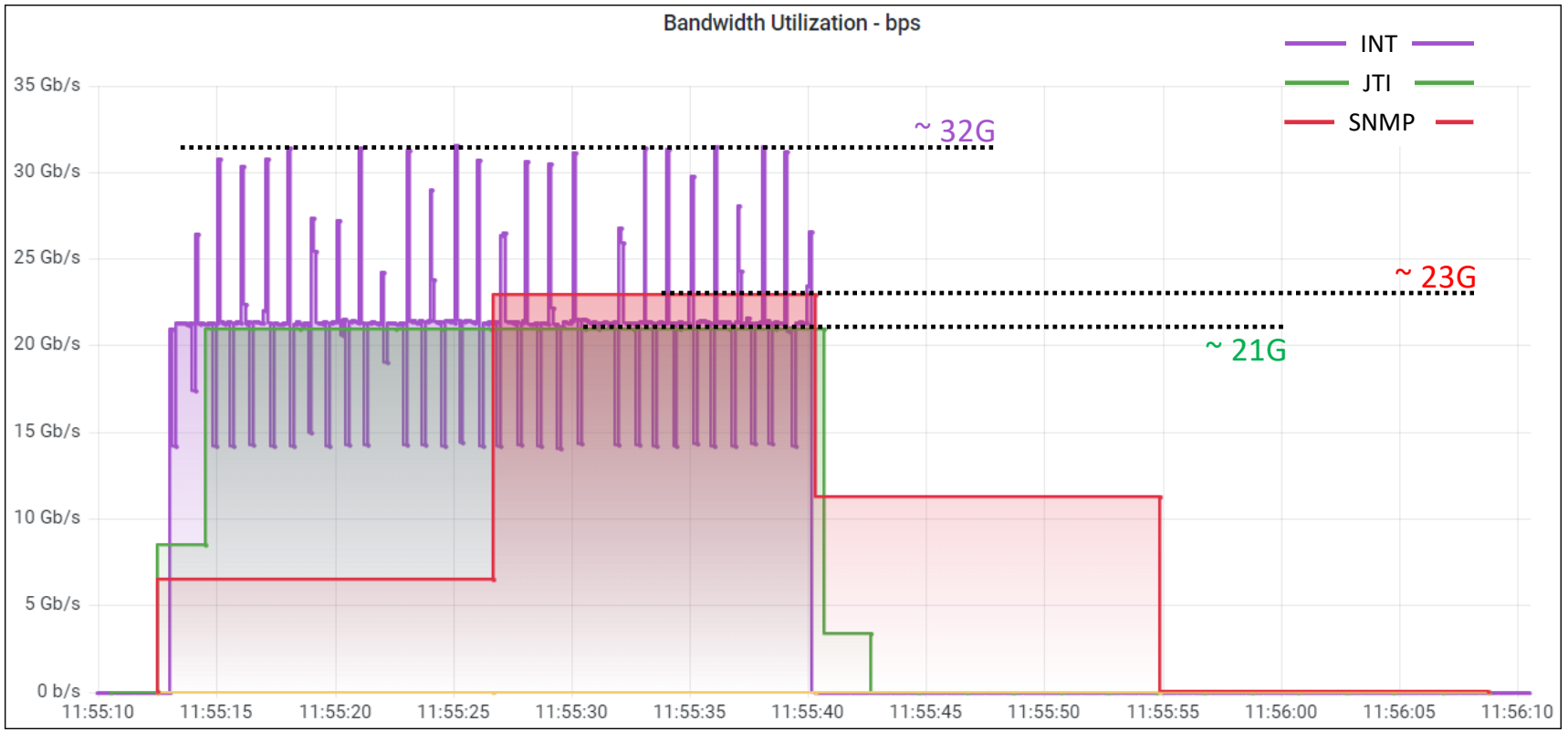
Identifying Bursts: SNMP x JTI x INT [Test 2]

- Interval: 30s.
- 2 Streams: Continuous and Burst.
- Continuous Traffic: 20G.
- Burst: **15x** 30G.
- Burst duration: **200ms**.
- Interval between bursts: **1.8s**.



Identifying Bursts: SNMP x JTI x INT [Test 3]

- Interval: 30s.
- 2 Streams: Continuous and Burst.
- Continuous Traffic: 20G.
- Burst: **30x** 30G.
- Burst duration: **50ms**.
- Interval between bursts: **0.95s**.



Why not lowering the gathering interval on the INT Collector?

Disk space is a concern!

AmLight in Numbers: 20 P4 switches x ~14 interfaces per switch in use x ~20 VLANs per interface

- 5,600 monitoring items (ignoring queues)
- Each measurement: Octets: 8 bytes + Packets: 8 bytes + Timestamp (nanosecond): 8 bytes = 24 bytes

AMOUNT OF MEASUREMENT DATA TO BE STORED IN GBYTES UNDER DIFFERENT INTERVALS

Intervals	25 ms	50 ms	100 ms	500 ms	1000 ms
Per second	0.00526	0.00263	0.00131	0.00026	0.00013
Per day	454.2	227.1	113.6	22.7	11.4

1

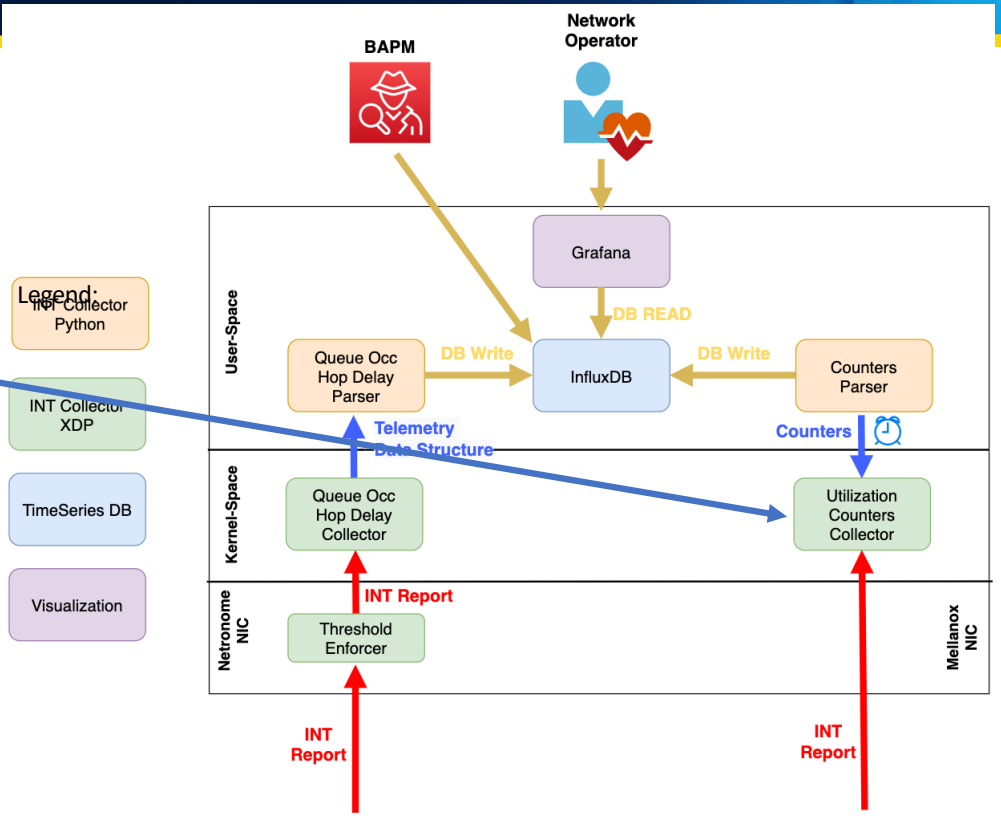
Due to the amount of data to store and the low frequency of microbursts, AmLight decided to modify the AmLight INT Collector to monitoring for microbursts every 20 ms **but** only storing data when a microburst is detected.

[1] NOMS 2022, An adaptive and efficient approach to detect microbursts leveraging per-packet telemetry in a production networks

The AmLight INT Collector 2.0: Our adaptive approach

Let's review the AmLight INT Collector 1.x

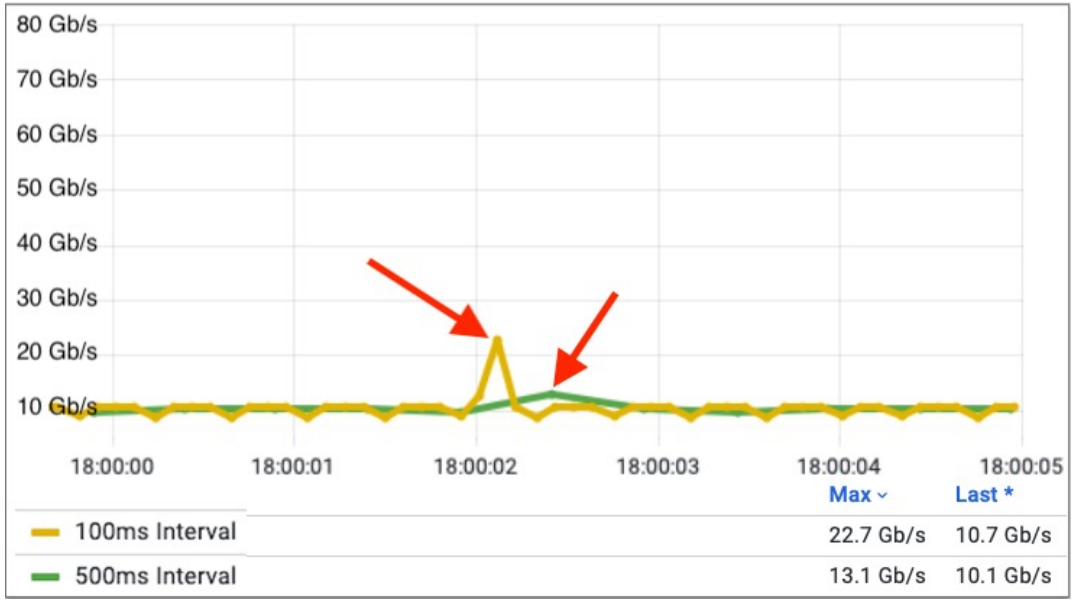
- eBPF/XDP/BCC + Influxdb + Grafana
- The Utilization Counters Collector
  - reads counters saved in memory via eBPF every **FIXED** interval (user-defined)
  - and pushes to Influxdb
  - **Default interval: 500ms**
- For version 2.x, the goal is to make intervals dynamic and adjusted at runtime.



https://docs.google.com/presentation/d/1CFpj-ZA1dbeJqeCHjN6OA8uMvhsgJRf2/edit#slide=id.p18

Our motivation to pursue microbursts @ AmLight

- Playing with the traffic generator:
  - A microburst traffic of 70Gbps for 40ms
  - And a constant 10Gbps flow
- Two instances of the AmLight INT Collector 1.1 running:
  - One gathering counters every 500 ms (default)
  - One gathering counters every 100 ms
- The results are provided in the figure.
  - None of them came even close to 80 Gbps
  - *We can't have gaps in our monitoring!*



AmLight INT Collector 2.0 – Microburst edition

- Goal 1: Leveraging INT to detect microbursts as is
- Goal 2: No more static gathering interval (every X milliseconds) for microbursts
- Goal 3: Monitoring bandwidth utilization in a very short time interval but saving bandwidth utilization in case of *pattern changes* (user-defined)

- Implementation:
  - Monitor bandwidth utilization every N milliseconds (10-20) and saving the last Δ bandwidth values in an array
  - Component 1: Every α milliseconds, evaluate the current bandwidth value against the last Δ values and a microburst factor
  - Component 2: Monitor bandwidth utilization using an adaptive approach
  - All metrics and values should be customizable and dynamic.
  - Our implementation consider microbursts events shorter than 500 ms.

Component 1: Detecting microbursts

Algorithm 1 searches for *pattern changes*: when current bandwidth grows above a predefined β factor using the last Δ values as reference.

α = loop interval to collect counters
γ = minimal bandwidth
β = growing factor
Δ = number of previous measurements
θ = microburst margin

Algorithm 1: Detecting Microbursts

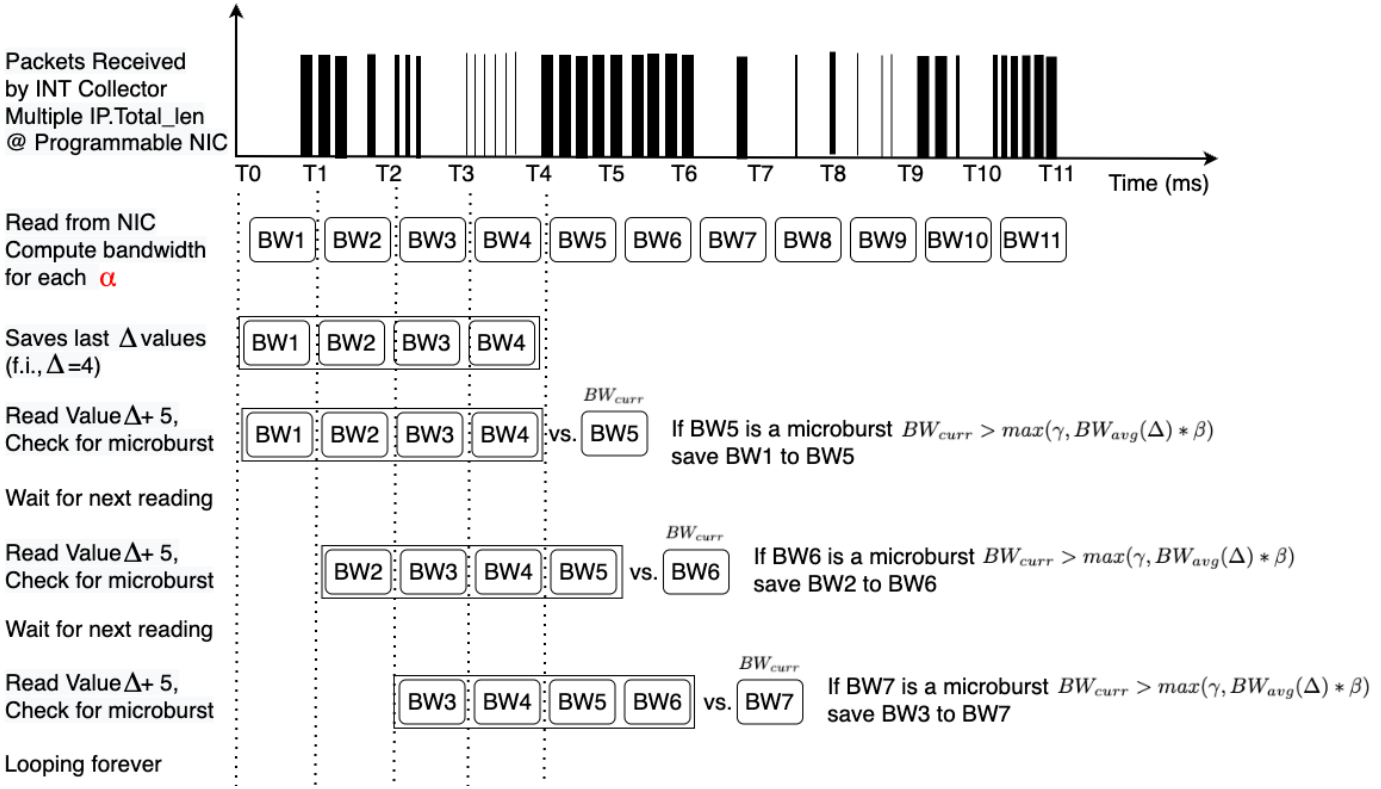
```
1  isBurst ← False;
2  while True do
3      BW_curr ← GetCurrBW();
4      if isBurst = False then
5          if BW_curr > max(γ, BW_avg(Δ) * β) then
6              isBurst ← True;
7              SaveLastValues(Δ);
8              SaveCurrBW(BW_curr);
9              BW_target ← BW_curr;
10     else
11         SaveCurrBW(BW_curr);
12         if not BW_curr > BW_target * θ then
13             isBurst = False;
14             SendToDatabase();
15     Sleep(α);
16 end
```

pattern changes

Saves state pre-microburst

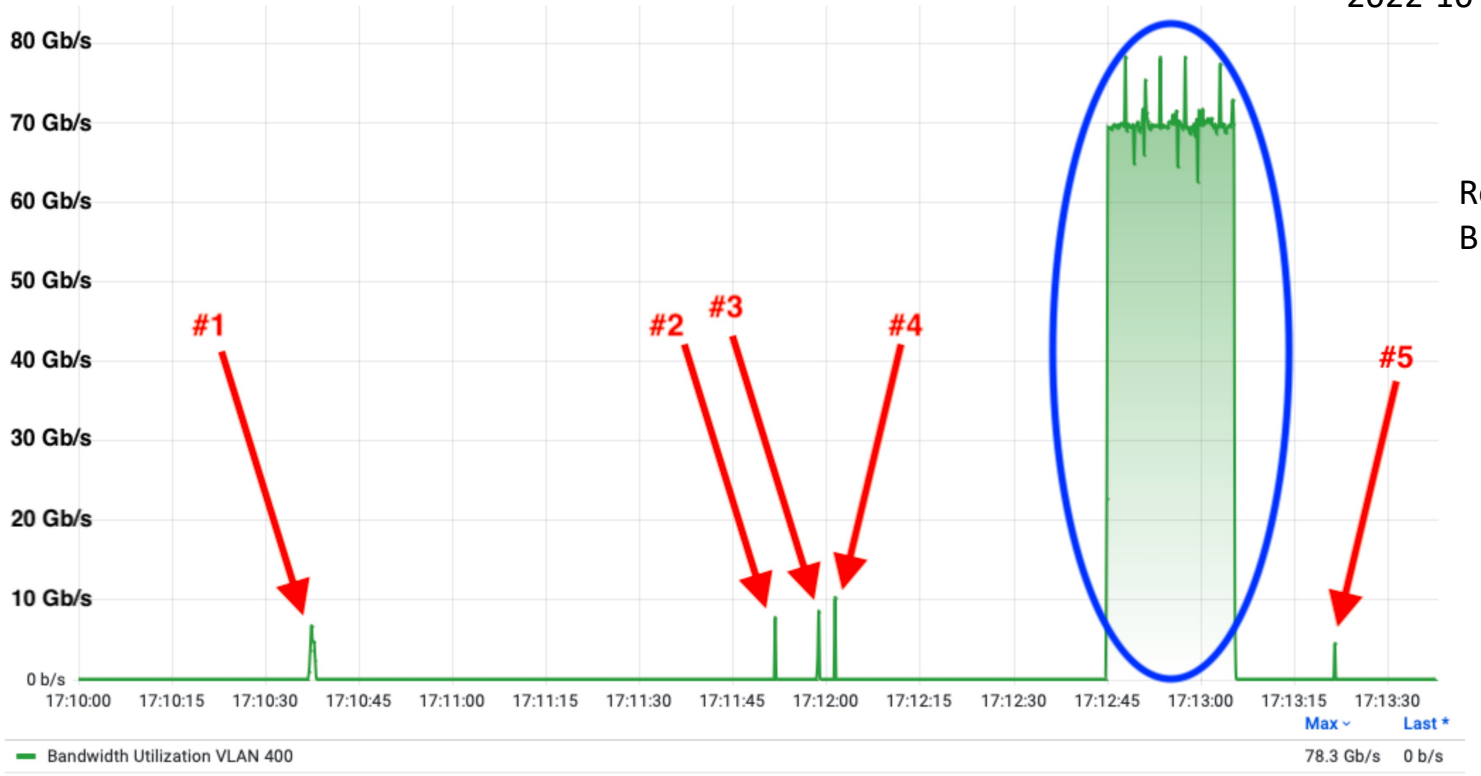
Once microburst is over, store it in disk

Component 1: Detecting microbursts [2]



SC22 warming up – Microbursts detected

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



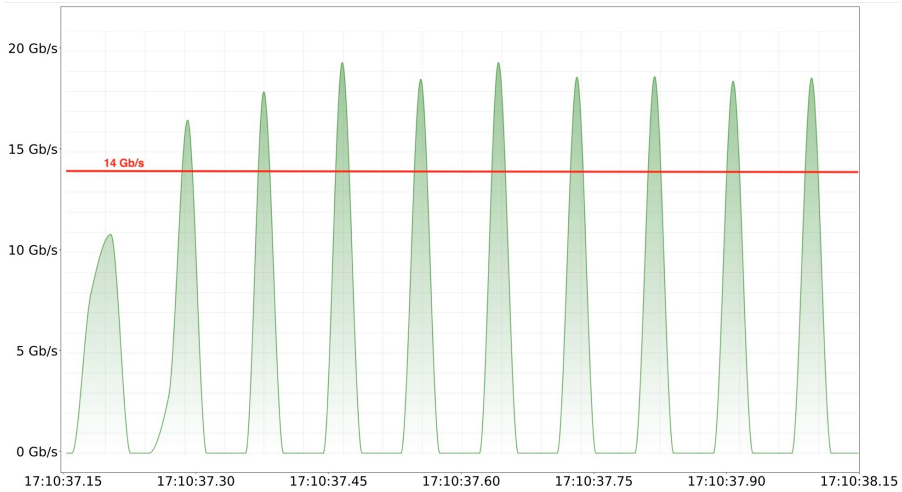
Red Arrows: Microburst
Blue Ellipse: Burst

$\alpha = 20$ ms
$\gamma = 14$ Gbps
$\beta = 4$
$\Delta = 3$
$\theta = 60\%$

AmLight ExP
Americas Lightpaths Express & Protect

SC22 warming up – Microbursts detected [2]

Zoom in on Microburst #1:



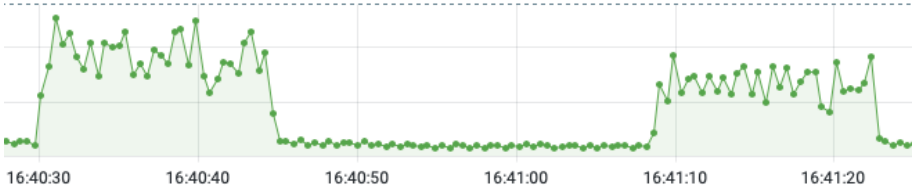
Start Time (UTC)	Duration (s)	Max BW (Gbps)
2022-10-09T13:10:37.304385768	0.02	16.35
2022-10-09T13:10:37.400937960	0.02	17.44
2022-10-09T13:10:37.499991784	0.02	18.88
2022-10-09T13:10:37.598316288	0.02	19.01
2022-10-09T13:10:37.696891136	0.02	18.97
2022-10-09T13:10:37.795097088	0.02	18.91
2022-10-09T13:10:37.893028608	0.02	19.09
2022-10-09T13:10:37.992322792	0.02	18.66
2022-10-09T13:10:38.091456256	0.02	18.91
2022-10-09T13:11:51.925128192	0.04	37.87
2022-10-09T13:11:58.794430952	0.06	53.41
2022-10-09T13:12:01.507265768	0.04	41.48
2022-10-09T13:13:21.666561768	0.04	20.83

Component 2: Adaptive Monitoring

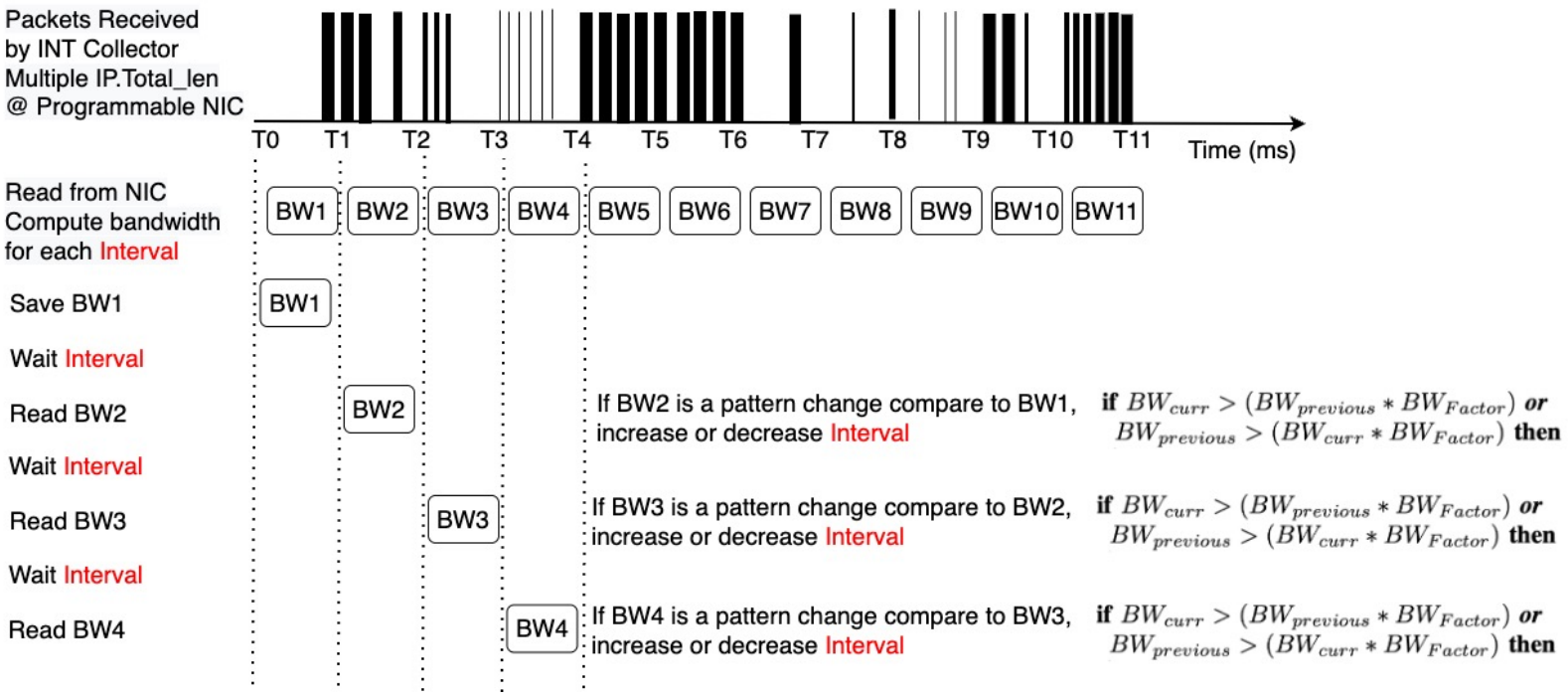
- The adaptive approach is keep processing the bandwidth counters and dynamically deciding whether to save them.
- Our strategy compares the observed current bandwidth **BWcurr** to previous value **BWprevious** to understand if bandwidth has varied “significantly” using a user-defined margin **BWFactor**.
- If bandwidth has not varied significantly, the strategy decreases or increases the interval via user-defined **DecreaseFactor** and **IncreaseFactor**.
- The network operator also controls the minimum **MinInterval** and maximum **MaxInterval** gathering and storage intervals.
- We consider that bandwidth varies when it increases or decreases more than a user-defined threshold, for instance, more than 15%.

To evaluate our solution, we consider efficiency the percentage of saved storage and accuracy by comparing how similar charts are.

Motivation:



Component 2: Adaptive Monitoring [2]



Component 2: Adaptive Monitoring [2]

- Complete version of Algorithm 2.
- DecreaseFactor and IncreaseFactor are key variables since they define how fast you react to bandwidth variation.

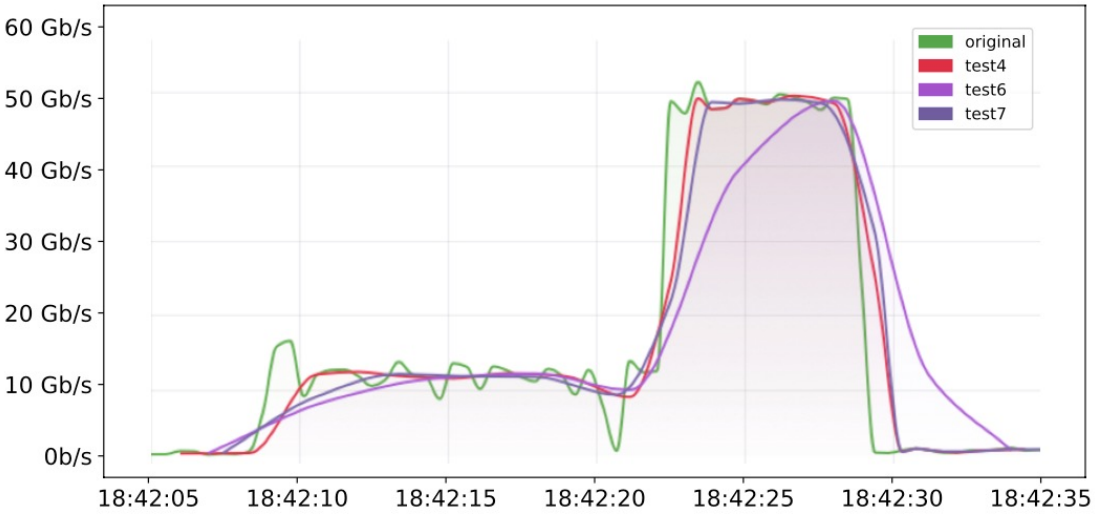
Algorithm 2: Adaptive Measurement Interval

```
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
```

Component 2: Adaptive Monitoring [2]

- 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)

Time Frame	Parameters	Similarity		Efficiency
		PSNR-B	SSIM	
Last 30 seconds	Setup 4	26.03	81.36%	61.64%
	Setup 6	26.23	81.48%	64.38%
	Setup 7	26.10	81.28%	64.38%
Last 30 days	Setup 4	34.88	97.92%	53.90%
	Setup 6	23.48	92.28%	67.48%
	Setup 7	33.70	99.24%	63.96%



Conclusion

- The AmLight INT Collect 2.0 was developed to support many user variables to define what microburst and adaptive mean to the network operator.

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

- In Part 2, answering “why and when is detecting microbursts important” is the main goal. We modeled some science drivers to showcase our microburst detection solution.

AmLight ExP
Americas Lightpaths Express & Protect

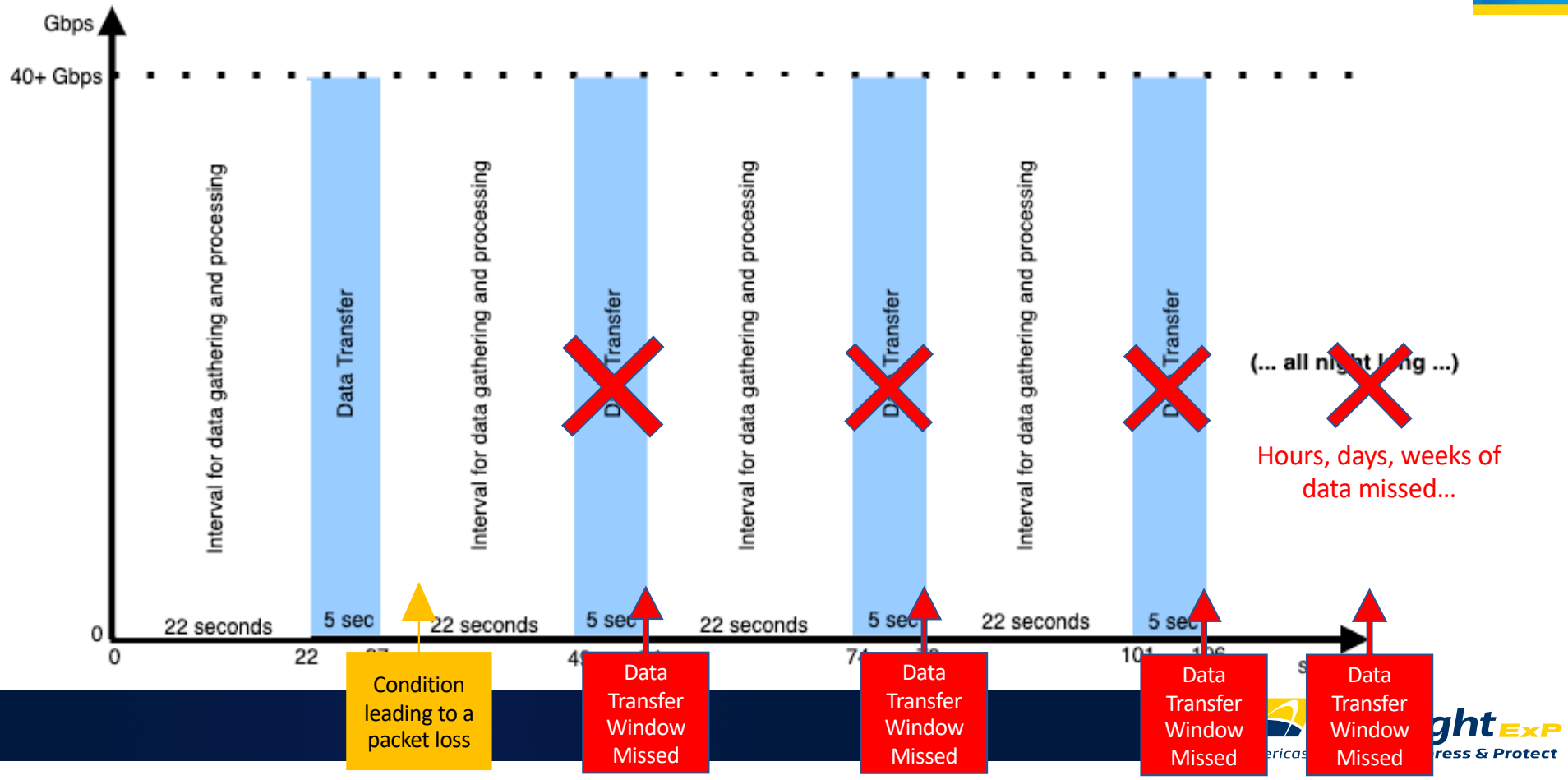
CI Lunch and Learn / April 21st, 2023

Thank you! Questions?

Handling Microbursts @ AmLight – Part 1 of 2

Jeronimo Bezerra <jab@amlight.net>

The Use Case: Vera Rubin Obs's operation



How does In-band Network Telemetry (INT) work?

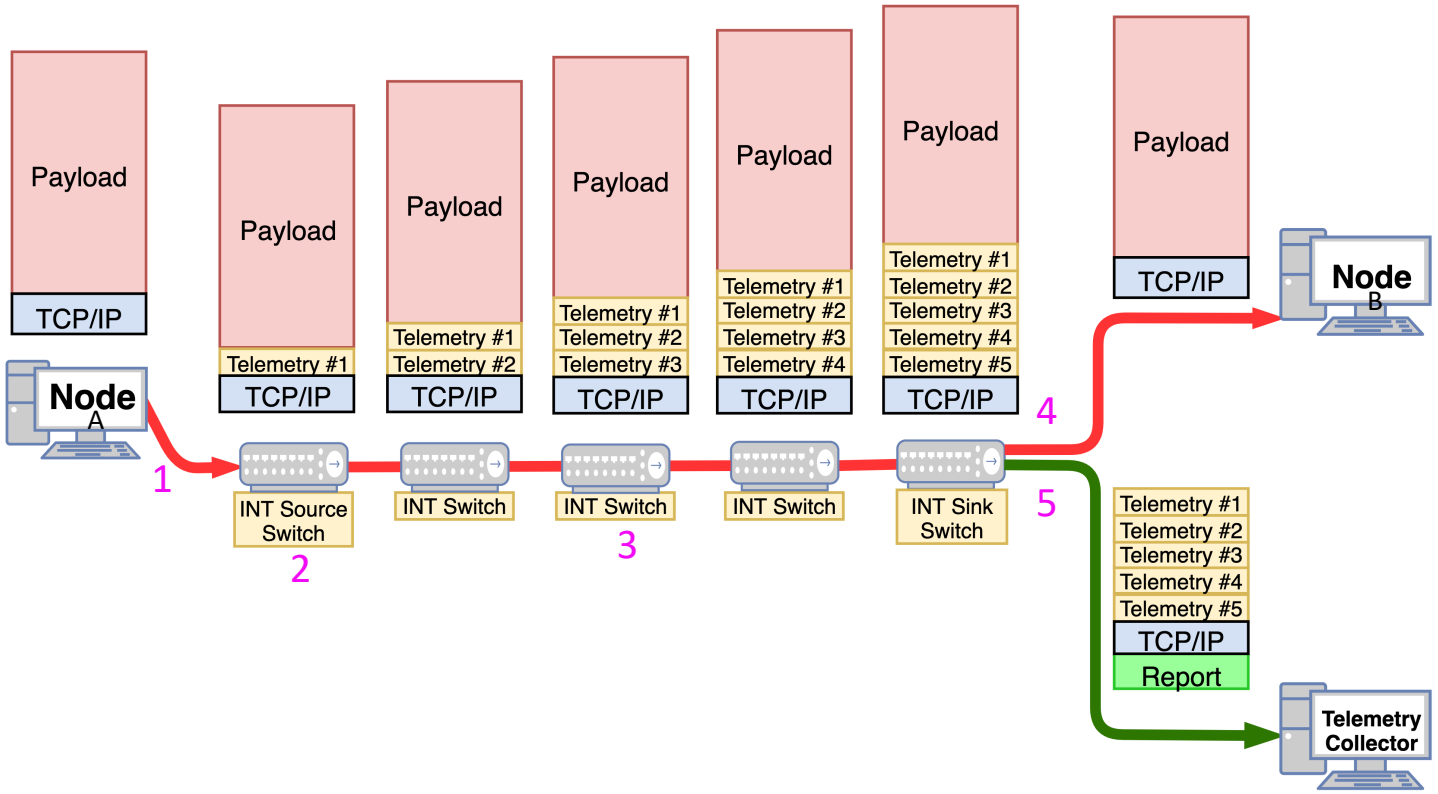
1 – User sends a TCP or UDP packet unaware of INT

2 – First switch (INT Source Switch) pushes an INT header + metadata

3 – Every INT switch pushes its metadata. Non-INT switches just ignore INT content

4 – Last switch (INT Sink Switch) extracts the telemetry and forwards original packet to destination

5 – Last switch (INT Sink Switch) forwards the 1:1 telemetry report to the Telemetry Collector



AmLight ExP
Americas Lightpaths Express & Protect

What INT metadata is being used and how?

- Instantaneous Ingress and Egress Interface utilization
  - Telemetry Collector monitors and reports egress interface utilization every customized interval (100-500ms)
  - Bandwidth monitored per interface & queue & VLAN
- Instantaneous Egress Interface Queue utilization (or buffer)
  - Useful for evaluating QoS policies
  - Useful for detecting sources of packet drops
- Per-Node Hop Delay
  - Useful for evaluating sources of jitter along the path
  - Useful for mitigating traffic engineering issues (under and over provisioned links)
- L1/L2 Path Tracing
  - EVERY packet and recording changes
  - Useful for detecting LAG or ECMP hash errors/mismatches and detecting unstable links
  - Path taken even reports the egress queue ID

Some nice outputs – more @ https://youtu.be/M6n_UZlhBQQ