Enabling Event Triggered Monitoring of Traffic Clusters

Gianni Antichi

gianni.antichi@cl.cam.ac.uk

University of Cambridge

in collaboration with:

J.Kucera, D.A.Popescu, J.Korenek and A.W.Moore

The **importance** of finding high-volume traffic clusters has been widely recognized in the past to **improve** network management practices

The **importance** of finding high-volume traffic clusters has been widely recognized in the past to **improve** network management practices

Network event	Management task
(Hierarchical) Heavy Hitters	accounting, traffic engineering
Changes in traffic patterns	anomaly detection, DoS detection
Superspreaders	worm, scan, DDoS detection

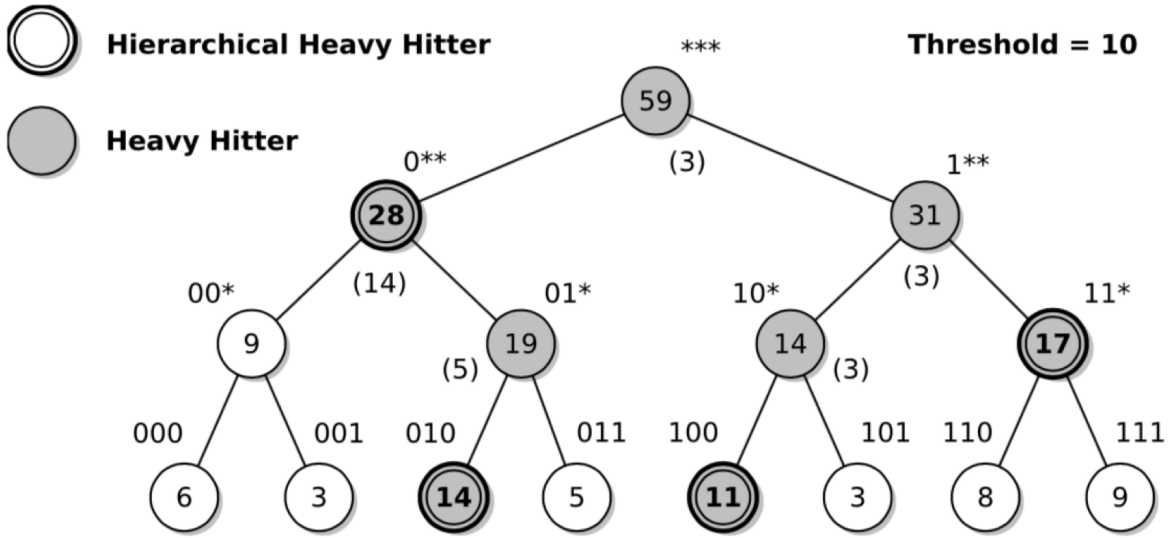
Let's first create a common ground

Network event	Management task
(Hierarchical) Heavy Hitters	accounting, traffic engineering
Changes in traffic patterns	anomaly detection, DoS detection
Superspreaders	worm, scan, DDoS detection

(Hierarchical) Heavy Hitters

Heavy Hitters (HH): a prefix that contribute with a traffic volume larger than a given threshold T during a fixed time interval t .

Hierarchical Heavy Hitter (HHH): a prefix that exceeds a threshold T after excluding the contribution of all its HHH descendants.



Changes in traffic patterns

Identifying the flows that contribute the most for the changes in the traffic patterns over two consecutive time intervals.

Network event	Management task
(Hierarchical) Heavy Hitters	accounting, traffic engineering
Changes in traffic patterns	anomaly detection, DoS detection
Superspreaders	worm, scan, DDoS detection

Superspreaders

A host that contacts at least a given number of distinct destinations over a short time period.

Network event	Management task
(Hierarchical) Heavy Hitters	accounting, traffic engineering
Changes in traffic patterns	anomaly detection, DoS detection
Superspreaders	worm, scan, DDoS detection

All those network events can be seen as a

traffic cluster detection problem

Network event	Management task
(Hierarchical) Heavy Hitters	accounting, traffic engineering
Changes in traffic patterns	anomaly detection, DoS detection
Superspreaders	worm, scan, DDoS detection

HHH and **change detection**: packets or bytes per second.

Superspreaders: flows per second.

Can we leverage **dataplane programmability** to **assist** in the detection of those events?

Can we leverage **dataplane programmability** to **assist** in the detection of those events?

Can we leverage **dataplane programmability** to **assist** in the detection of those events?

HashPipe [1]: calculate within the dataplane the top-k heavy hitters at fixed intervals.

[1] Heavy-Hitter Detection Entirely in the Data Plane, V. sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, J. Rexford. In ACM SOSR 2017.

Can we leverage **dataplane programmability** to **assist** in the detection of those events?

HashPipe [1]: calculate within the dataplane the top-k heavy hitters at fixed intervals.

Focus on heavy hitter only.

[1] Heavy-Hitter Detection Entirely in the Data Plane, V. sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, J. Rexford. In ACM SOSR 2017.

Can we leverage **dataplane programmability** to **assist** in the detection of those events?

HashPipe [1]: calculate within the dataplane the top-k heavy hitters at fixed intervals.

Focus on heavy hitter only.

Univmon [2]: assist the controller by exporting smart representation of aggregated statistics.

[1] Heavy-Hitter Detection Entirely in the Data Plane, V. sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, J. Rexford. In ACM SOSR 2017.
[2] One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon, Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, V. Braverman. In ACM SIGCOMM 2016.

Can we leverage **dataplane programmability** to **assist** in the detection of those events?

HashPipe [1]: calculate within the dataplane the top-k heavy hitters at fixed intervals.

Focus on heavy hitter only.

Univmon [2]: assist the controller by exporting smart representation of aggregated statistics.

The actual detection is performed in the control plane.

[1] Heavy-Hitter Detection Entirely in the Data Plane, V. sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, J. Rexford. In ACM SOSR 2017.

[2] One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon, Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, V. Braverman. In ACM SIGCOMM 2016.

Can we leverage **dataplane programmability** to **assist** in the detection of those events?

HashPipe [1]: calculate within the dataplane the top-k heavy hitters at fixed intervals.

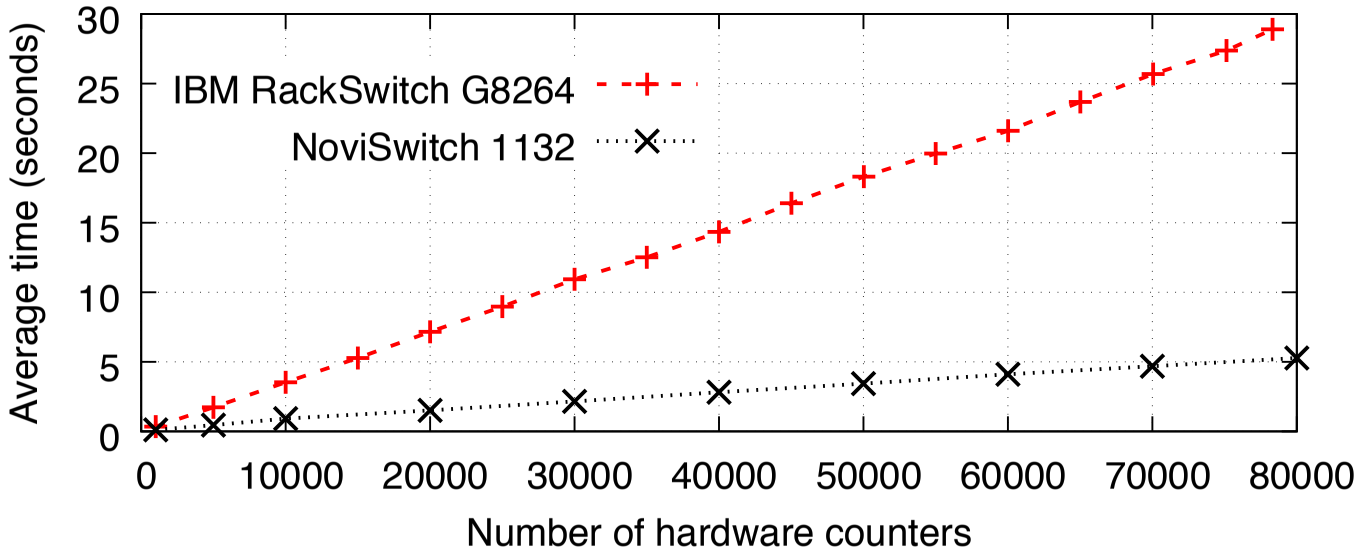
Focus on heavy hitter only.

Univmon [2]: assist the controller by exporting smart representation of aggregated statistics.

The actual detection is performed in the control plane.

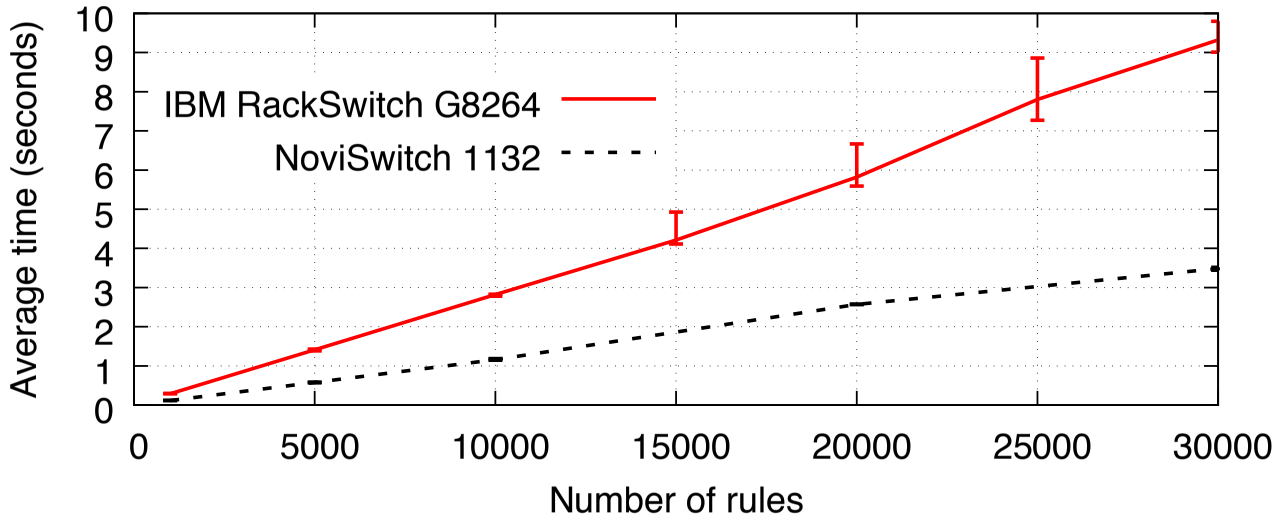
Wait a minute. Is this a problem?

Retrieving a large number of counters from hardware is **time consuming**!!!



Note: probabilistic data structures (i.e., sketches) require large amount of counters to lower false positive ratio.

Updating forwarding state and statistic retrieval are two **competing operations** that are commonly run sequentially



Note: having large chunk of forwarding updates is a pretty common case during blackholing.

Can we leverage **dataplane programmability** to **assist** in the detection of those events?

Can we leverage **dataplane programmability** to ~~**assist**~~ in the detection of those events?

Can we leverage **dataplane programmability** to **enable in-network** detection of those events?

Because if you have in-network detection..

As soon as you detect you can take pre-defined actions.

Good for network reactiveness.

Because if you have in-network detection..

As soon as you detect you can take pre-defined actions.

Good for network reactiveness.

You can directly export the detection result to the control plane.

Control plane does not have to receive lot of data and understand what is going on.

Because if you have in-network detection..

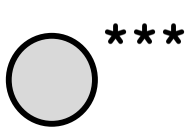
Elastic Trie in a nutshell

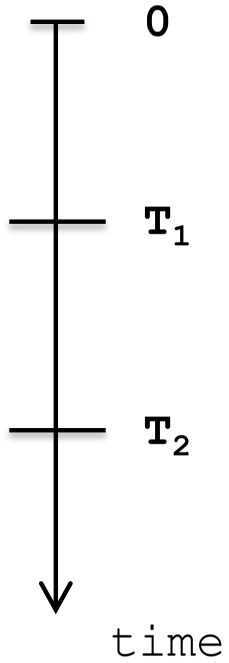
- Prefix tree that **grows** or **collapses:** focus on who account for a *large* share of the traffic.
- **Starting condition:** a single node corresponding with zero-length prefix *.
- Each node consists of **three elements**: (1) left child counter, (2) right child counter, (3) node timestamp.
- Use **timeouts** to detect heavy prefixes and to grow or collapse the trie

(i.e., if in the time interval t, prefix A exceeds a threshold, then refine the prefix)

Elastic Trie in action

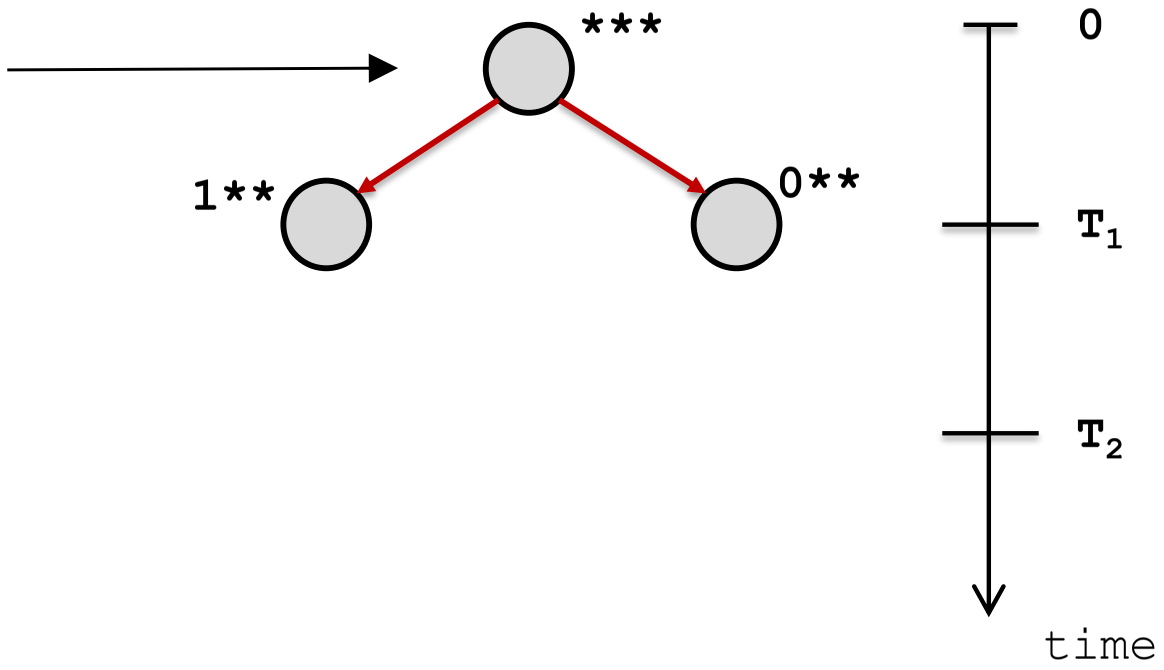
- counter-Left
- timestamp
- counter-Right





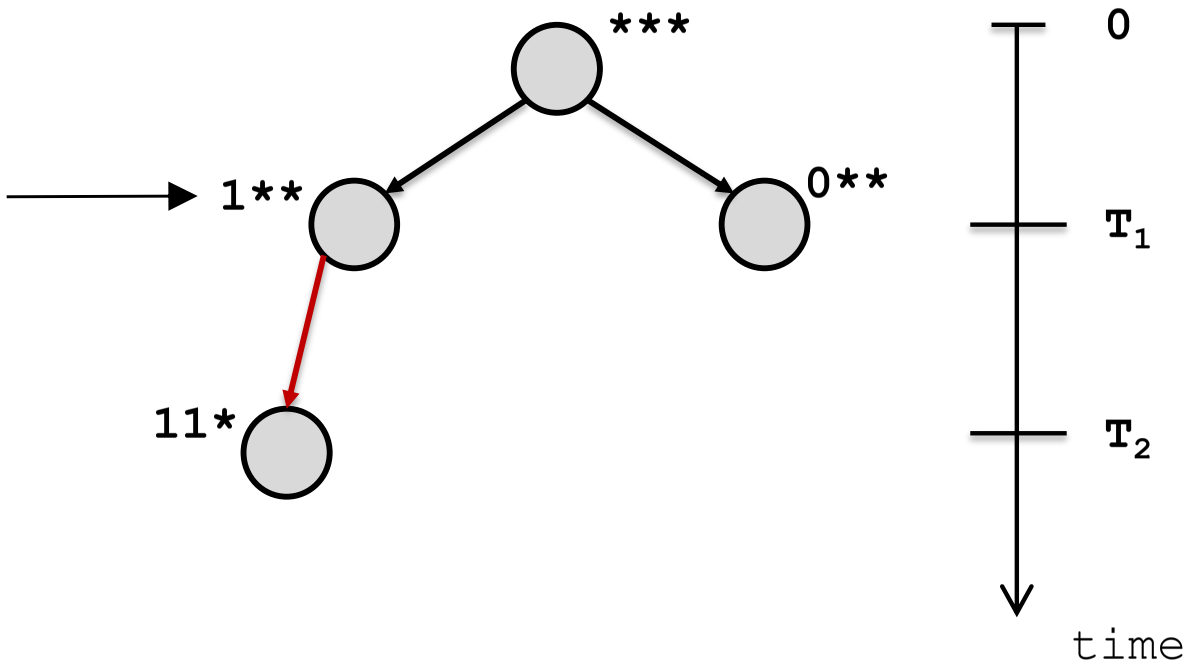
Elastic Trie in action

Both counterL and counterR exceed threshold in T_1

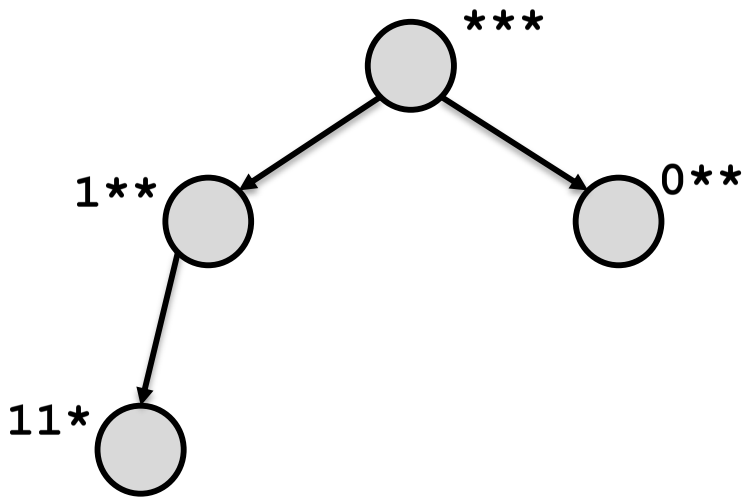


Elastic Trie in action

counterL exceeds threshold in T_2



Elastic Trie in action

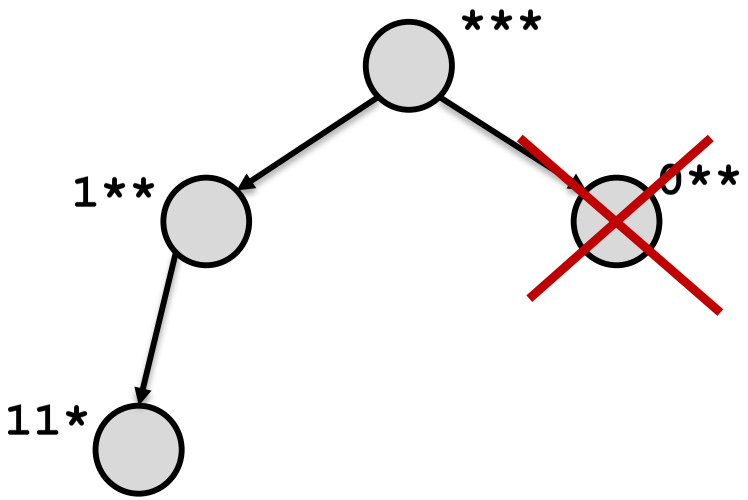


Packet timestamp

>>

Node timestamp

Elastic Trie in action

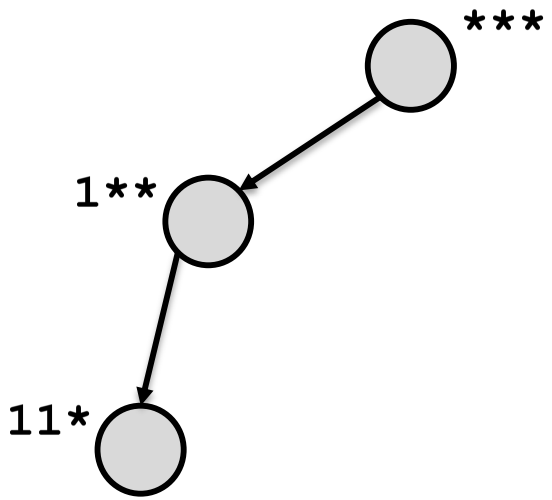


Packet timestamp

>>

Node timestamp

Elastic Trie in action



Elastic Trie implications

- The dataplane iteratively refine the responsible IP prefixes:

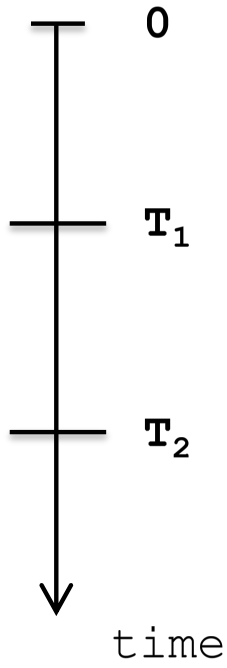
the controller can receive a **flexible granularity information**.

- Each prefix tree layer can have a **different timeout**:

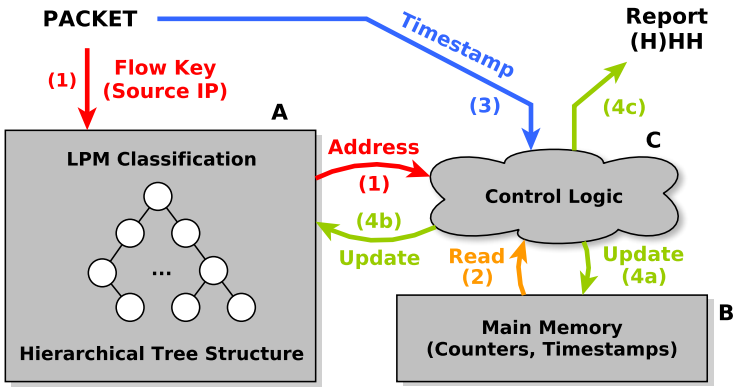
trade-off between trie building process and memory consumption.

- By looking at the **growing rate** of the trie it is possible to:

identify changes in the traffic patterns.

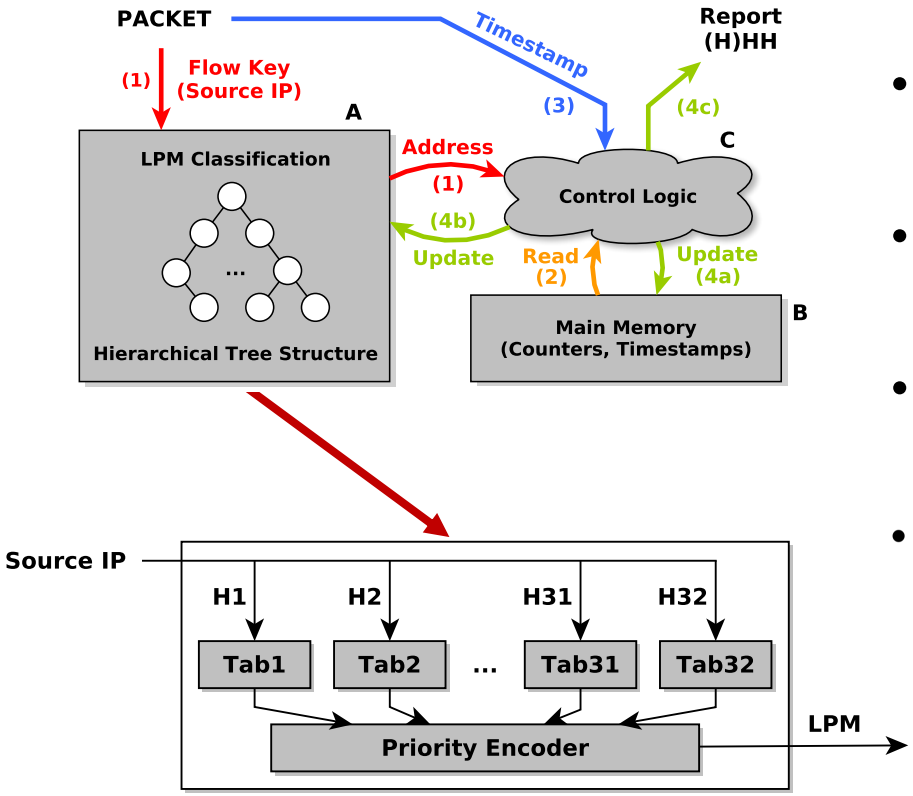


Elastic Trie in P4



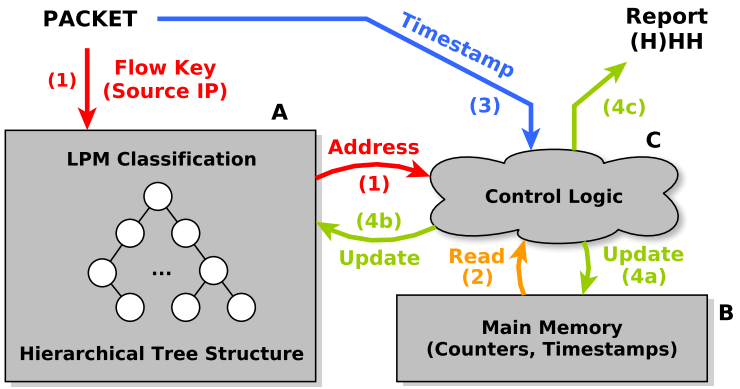
- **LPM classification:** the prefix tree
- **Control logic:** the brain
- **Main memory:** where all the per-node information are stored

Elastic Trie in P4: LPM classification



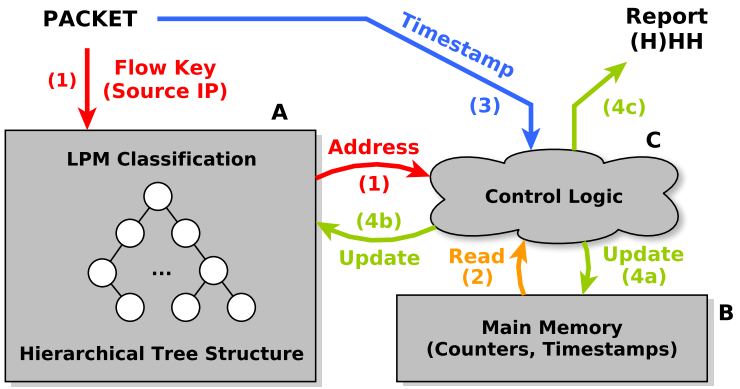
- We cannot modify entries in the dataplane itself
- A hash table for each prefix length
- Each hash table implemented as register array
- Hash extern API with CRC32

Elastic Trie in P4: main memory



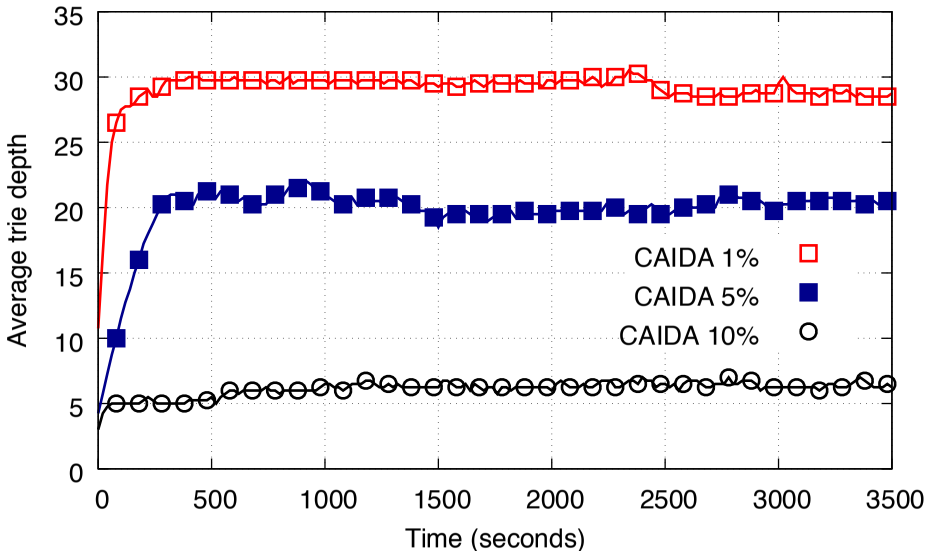
- The hash value of the LPM is the address to access a register that stores the node information

Elastic Trie in P4: control logic

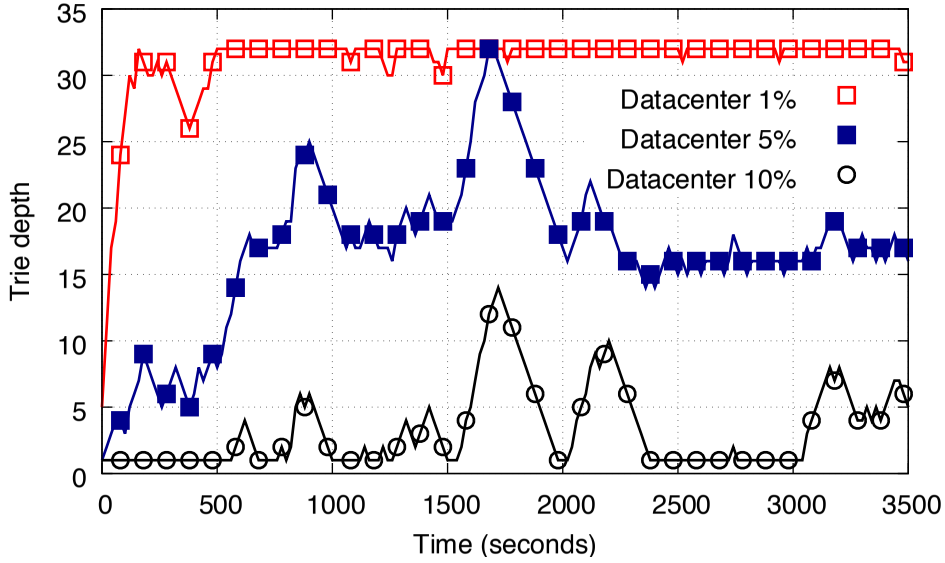


- We compare node timestamp and packet timestamp
- It implements the node update logic, and the push-based mechanic with a *digest message*

Elastic Trie in action



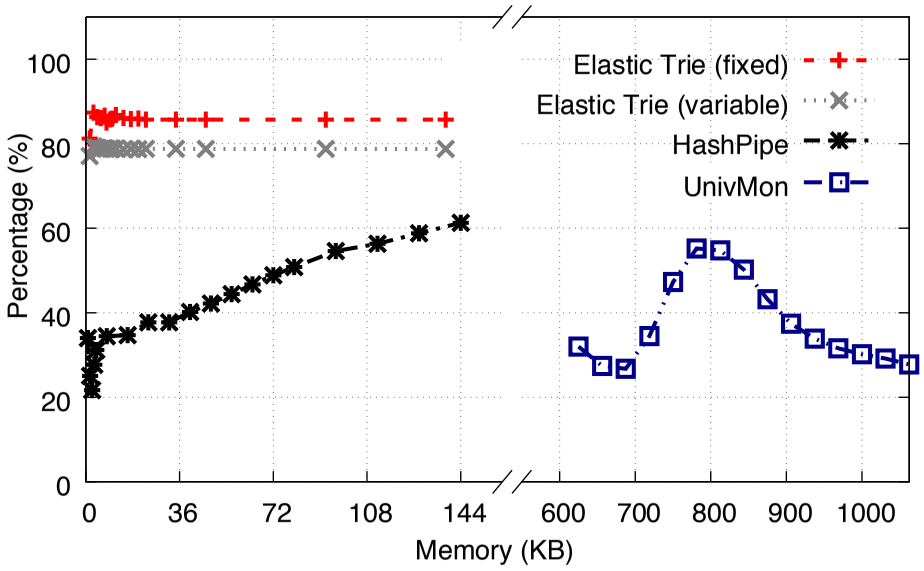
ISP



Datacenter

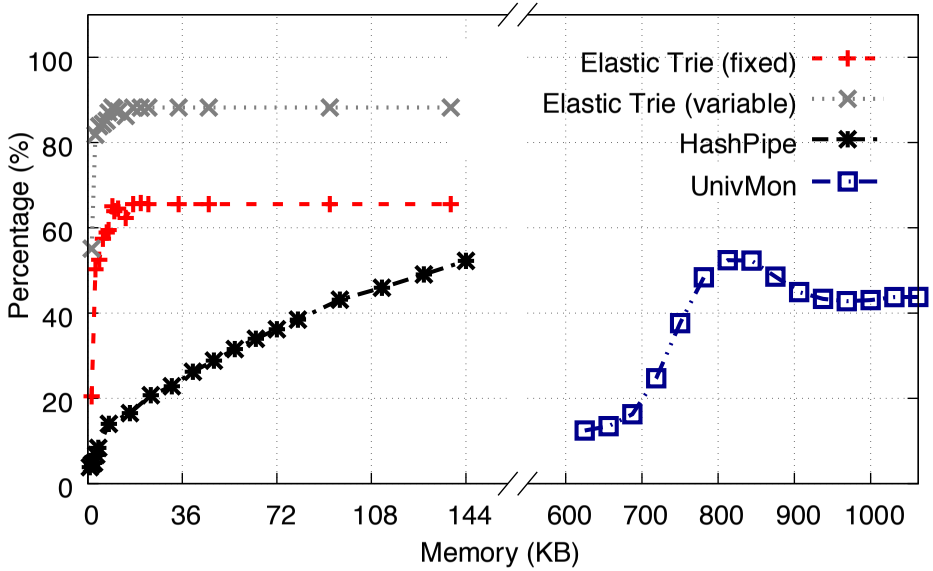
Elastic Trie in action

Precision



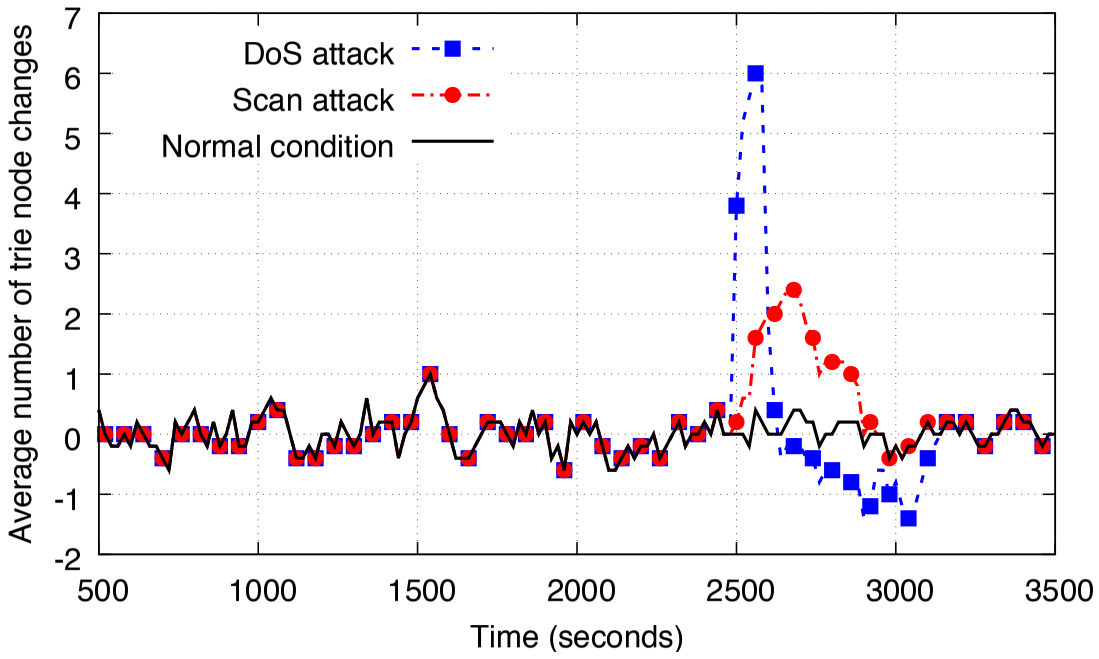
$$\frac{true_{positives}}{true_{positives} + false_{positives}}$$

Recall



$$\frac{true_{positives}}{true_{positives} + false_{negatives}}$$

Elastic Trie in action



Changes can be spotted!!!

Conclusions

- Elastic Trie enables **in-network detection** of traffic aggregates
- **Push-based monitoring** approach
- Suitable for **HH**, **HHH**, **Superspreader** and **Change detection**.
- Low memory footprint!