Distributed Summary Statistics with Bro

Vlad Grigorescu

> whoami

- Member of the Bro development team
- Senior Developer at Broala LLC
- Senior Information Security Engineer at Carnegie Mellon University

https://github.com/grigorescu

@0f010d

Goal

To develop statistics that can efficiently summarize network activity distributed over a large number of sensors, while minimizing memory usage.

Outline

1. Observation examples
2. What types of questions can we answer?
3. SumStats Framework
    1. Overview
    2. Available Reducers
4. Real-world usage

Observation Examples

- 192.168.2.13 received an NXDOMAIN reply for a DNS A query of: host.244.ipoe2.subnets.khb.ttkdv.ru

Observation Examples

- 192.168.2.14 received a 403 Forbidden when performing a POST to: http://sqm.microsoft.com/sqm/Windows/sqmserver.dll

Observation Examples

- 192.168.2.15 sent an e-mail with an application/x-dosexec attachment, with MD5 hash c84a46850de0a29483ed1f7a0b9897ab

What types of questions can we answer?

- Which source/dest IP pairs have the lowest variance in TCP session byte counts?
- Which ASNs have the highest number of connections into your network?
- Which IP source has connected to the highest number of unique destinations?

What types of questions can we answer?

- In the past 24 hours, which clients have sent the most failed DNS queries?
- Which servers have received the most failed DNS queries?
- If we look at each IP's ratio of failed to total DNS queries, which IPs have had over 90% failures?

SumStats Framework

- A set of Bro scripts for generating summary statistics
- Tie into the existing Bro scripts to make observations about events in layers 2-7
- Can threshold values to create notices, which can prompt automated responses
- Can query the current values for more advanced use-cases scripts

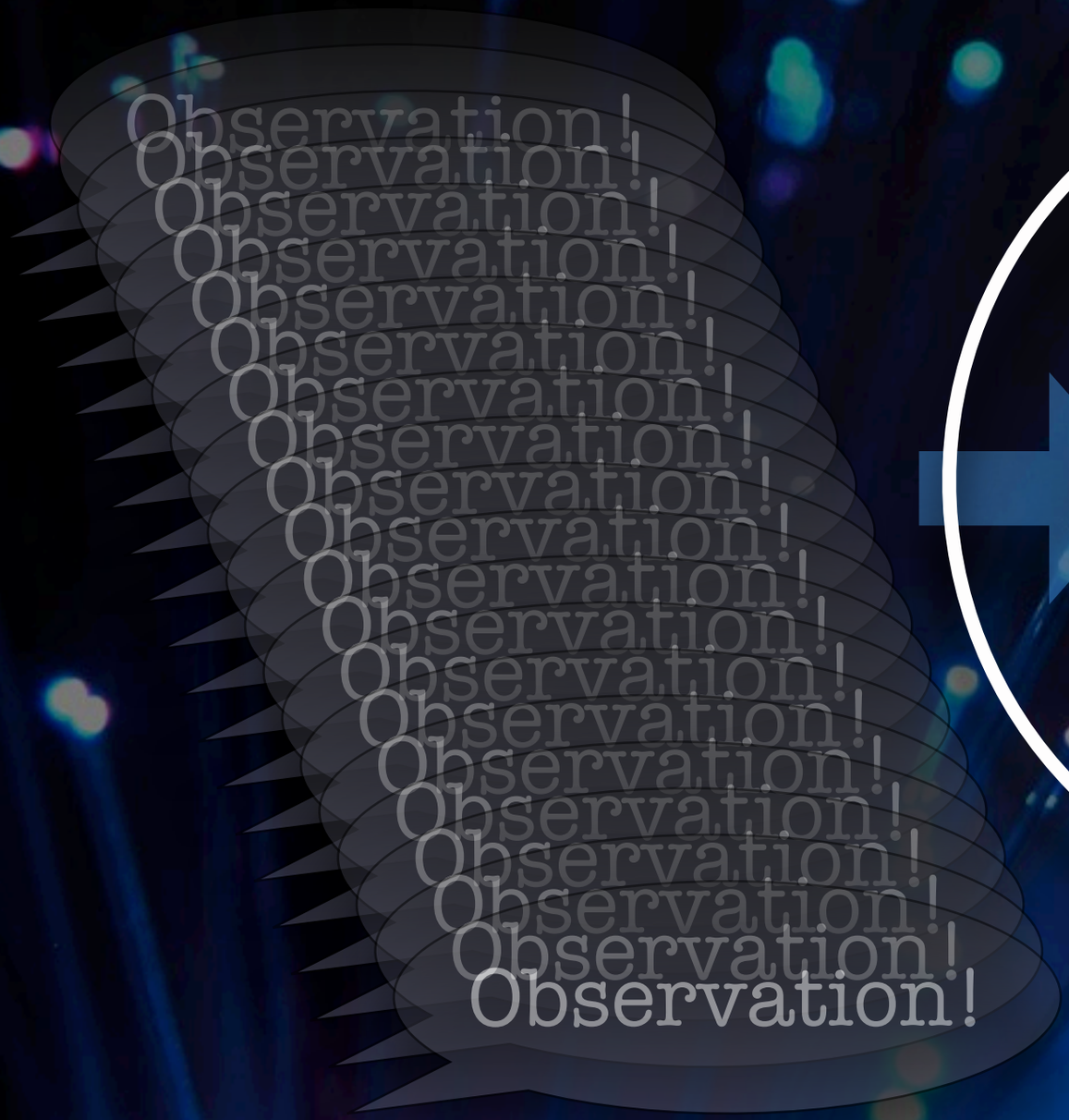
SumStats Framework: Philosphy

All summary statistics must be:

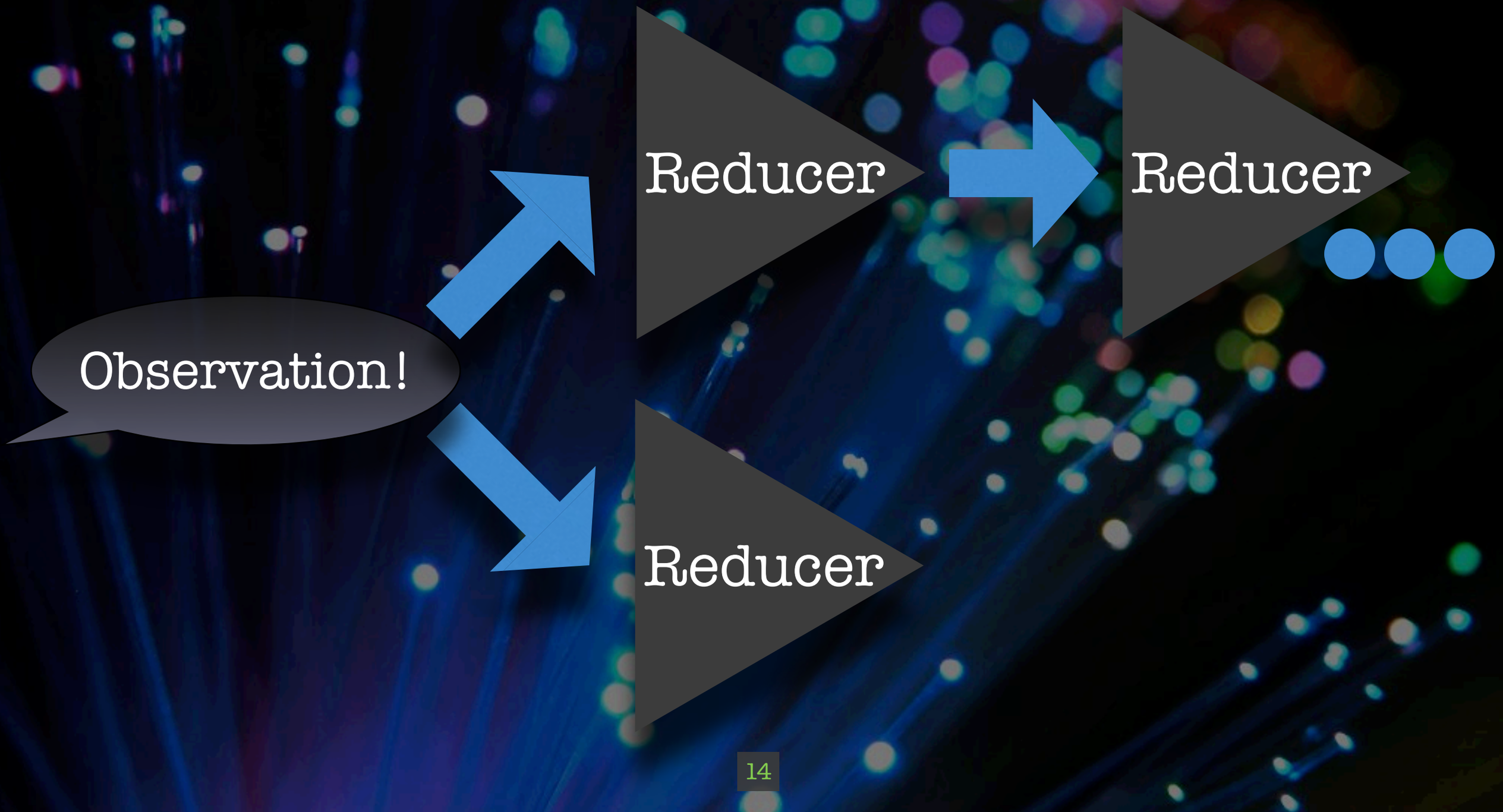
- Highly memory efficient,
- Streaming (the data is only seen once),
- Mergable (distributable across thousands of nodes, each of which see a subset of the total traffic)

SumStats Framework: Design

SumStats Framework: Design



SumStats Framework: Design



SumStats Framework: Reducers

"Classic" Stats:

- Average
- Min
- Max
- Last
- Sum
- Std Dev
- Variance
- Cardinality

"Memory Efficient" Stats:

- HyperLogLog
- Top-k
- Reservoir Sampling

Reducers: HyperLogLog

- Streaming algorithm for calculating cardinality of huge datasets
- Can calculate cardinality of 1 billion elements with a relative accuracy of 2% using 1.5 KB of memory
- Mergeable without any loss in accuracy

Reducers: HyperLogLog

Which IP source has connected to the highest number of unique destinations?

Let's assume that you have a fully populated /8 network (16.5M hosts). We want to know the cardinality of destinations for each host.

16.5M ×1.5 KB ≈ 24 GB of RAM

Reducers: Top-k

- Streaming algorithm for finding the most frequent elements in a dataset, in a space-saving way
- Implementation of: Metwally A, Agrawal D, El Abbadi A (2005) **Efficient computation of frequent and top-k elements in data streams.**

Reducers: Top-k

Which IP source has connected to the highest number of unique destinations?

Connect our HyperLogLog reducer to a Top-k reducer.

Still assuming /8 network and 2% error; top talker connected to 1000 destinations ≈ 6 GB of RAM.

Real-World Usage: Writing a SumStat Script

Which source/dest IP pairs have the lowest variance in TCP session byte counts?

Real-World Usage: Writing a SumStat Script

1. Observation:

```
event connection_state_remove(c: connection)
        {
        SumStats::observe("end_of_conn",
           [$key=cat(c$id$orig_h,c$id$resp_h)],
           [$num=c$orig$size+c$resp$size]);
        }
```

Real-World Usage: Writing a SumStat Script

2. Reducers:

```
local r1 = SumStats::Reducer(
                $stream="end_of_conn",
                $apply=set(SumStats::VARIANCE,
                           SumStats::SUM)
           );
```

Real-World Usage: Writing a SumStat Script

3. SumStat:

```
SumStats::create(
    [$name="variance_of_orig_bytes",
     $epoch=5min, $reducers=set(r1),
     $threshold_val=(1-variance), #See note
     $threshold=0.9,
     $threshold_crossed=doNotice()#See note
    ]);
```

Note: Slightly simplified for brevity where commented.

Real-World Usage: scan.bro

Tracks the number of failed connection attempts ("port scans") by source IP. Generates a notice when:

- A source scans over 25 unique IPs on the same port within 5 minutes, or
- A source scans over 25 unique ports on the same destination IP within 5 minutes.

Real-World Usage: scan.bro

- Carnegie Mellon sees approximately 3000-6000 failed connection attempts per second
- scan.bro uses approx. 150 MB of RAM and has detected 49,500 scans from July-November 2013

Ongoing Work

- Writing more SumStats scripts to detect:
  - DNS amplification attacks
  - Beaconing
  - Behavioral changes