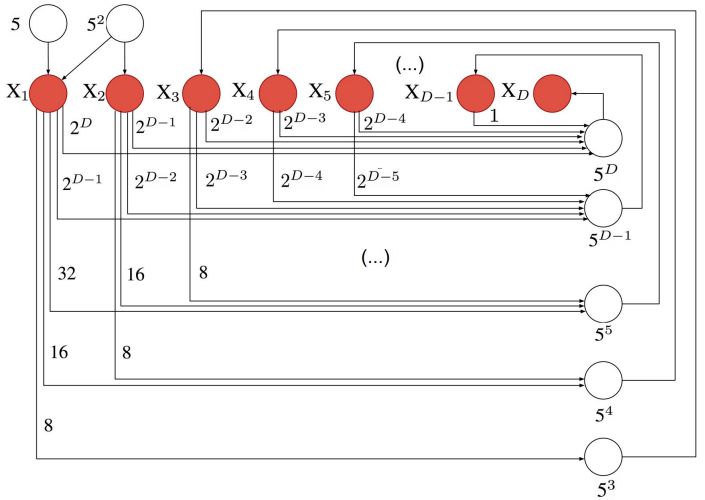
GradientGraph Analytics: Identifying Small Yet High Impact Flows Using Zeek to Optimize Network Performance

ZeekWeek 2019

Reservoir Labs

Jordi Ros-Giralt, Sruthi Yellamraju, Peter Cullen, Troy Hanson, James Ezick, Alison Ryan, Erik Mogus, Richard Lethin

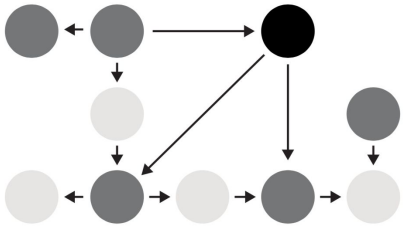


GradientGraph Analytics

- Tackles critical network operations objective: Bringing today's network utilization from below 30% to above 90%.
- Leverages Zeek for a (1) seamless and (2) scalable integration with (3) full visibility.

  [spoiler alert - open sourcing a new Zeek analyzer]

GradientGraph Analytics



Are all Elephant Flows Heavy Hitters?

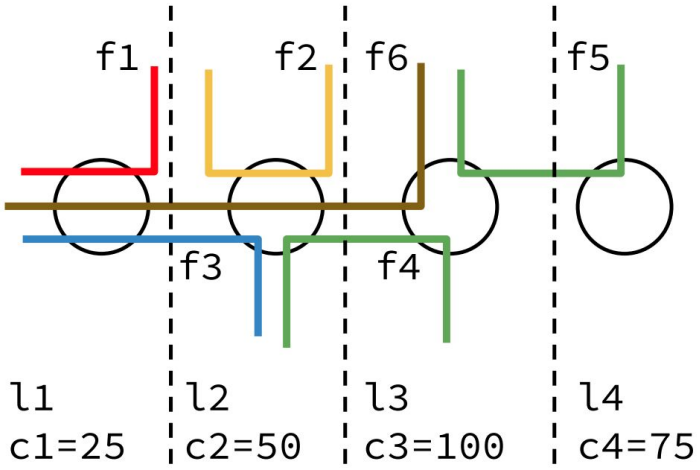
- Suppose N is a network with 6 TCP flows that receive this rate allocation vector: $\mathbf{r} = [8.3, 16.6, 8.3, 16.6, 75, 8.3]$ Mbps
- Which is the largest (elephant) flow?

Are all Elephant Flows Heavy Hitters?

- Suppose N is a network with 6 TCP flows that receive this rate allocation vector: $\mathbf{r} = [\underset{f_1}{8.3}, \underset{f_2}{16.6}, \underset{f_3}{8.3}, \underset{f_4}{16.6}, \underset{f_5}{75}, \underset{f_6}{8.3}]$ Mbps
- Which is the largest (elephant) flow?

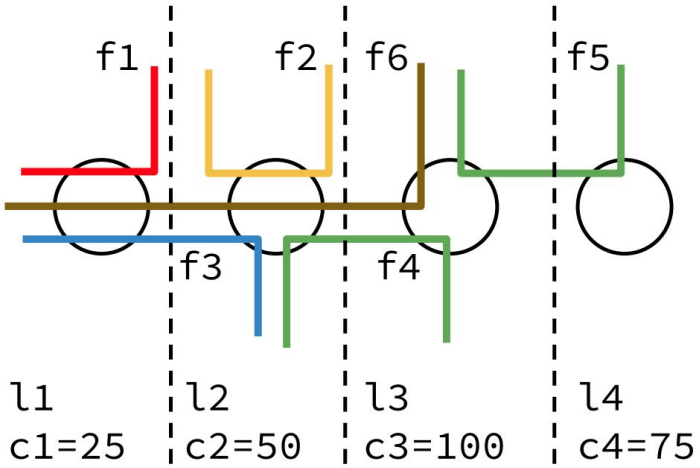
Are all Elephant Flows Heavy Hitters?

- Suppose N is a network with 6 TCP flows that receive this rate allocation vector: $\mathbf{r} = [\underset{f_1}{8.3}, \underset{f_2}{16.6}, \underset{f_3}{8.3}, \underset{f_4}{16.6}, \underset{f_5}{75}, \underset{f_6}{8.3}]$ Mbps
- Which is the largest (elephant) flow?

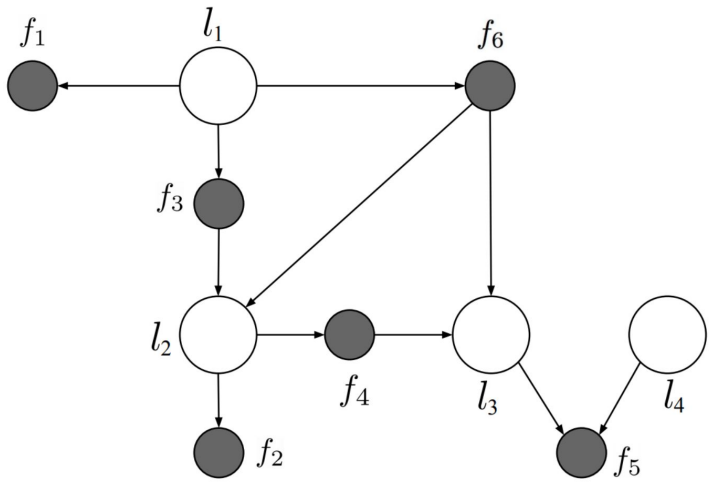


Are all Elephant Flows Heavy Hitters?

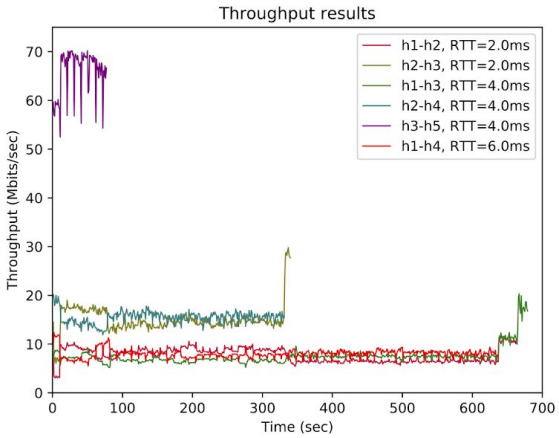
- Suppose N is a network with 6 TCP flows that receive this rate allocation vector: $\mathbf{r} = [\underset{f_1}{8.3}, \underset{f_2}{16.6}, \underset{f_3}{8.3}, \underset{f_4}{16.6}, \underset{f_5}{75}, \underset{f_6}{8.3}]$ Mbps
- Which is the largest (elephant) flow?



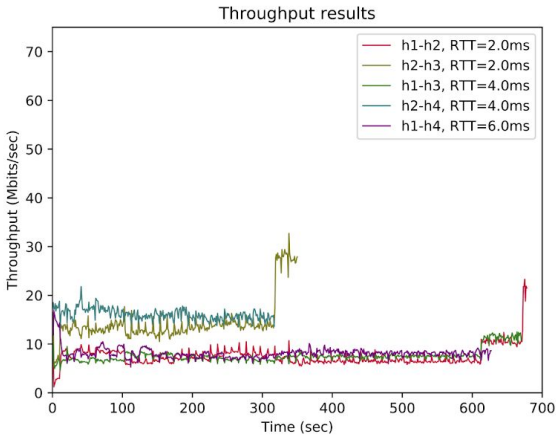
Flow Gradient Graph:



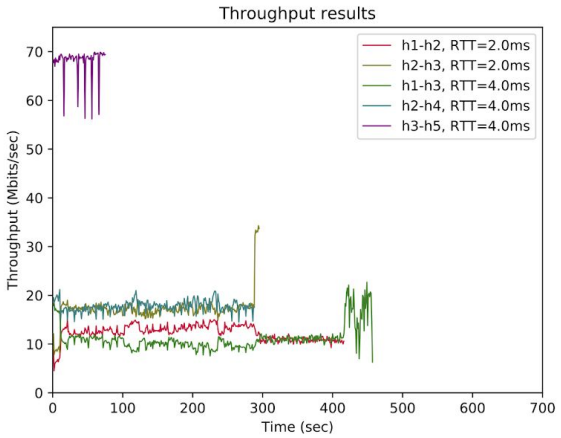
Are all Elephant Flows Heavy Hitters?



(a) Without removing any flow.



(b) Removing the heavy-hitter flow f_5.

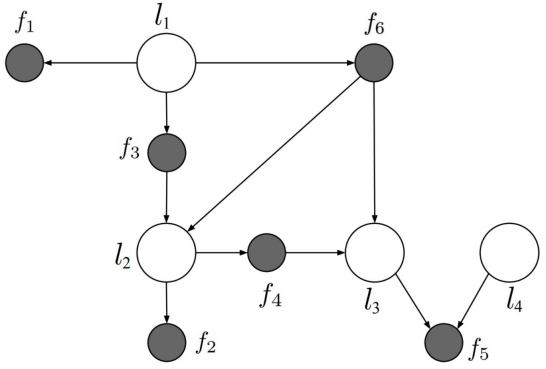


(c) Removing a low-hitter flow f_6.

Table 3: As predicted by the theory of bottleneck ordering, flow f_6 is a significantly higher impact flow than flow f_5.

Comp. time (secs)	f_1	f_2	f_3	f_4	f_5	f_6	Slowest
With all flows	664	340	679	331	77	636	679
Without flow f_5	678	350	671	317	—	611	678
Without flow f_6	416	295	457	288	75	—	457
Avg rate (Mbps)	f_1	f_2	f_3	f_4	f_5	f_6	Total
With all flows	7.7	15.1	7.5	15.4	65.8	8.1	119.6
Without flow f_5	7.5	14.5	7.6	16.1	—	8.3	54
Without flow f_6	12.2	17.2	11.1	17.7	68.1	—	126.3

Flow Gradient Graph:



Problem Scope and Relevance

[Slide taken from Bill Johnston's talk at ASCAC19: "ESnet: Advanced Networking for Data-Intensive Science"]

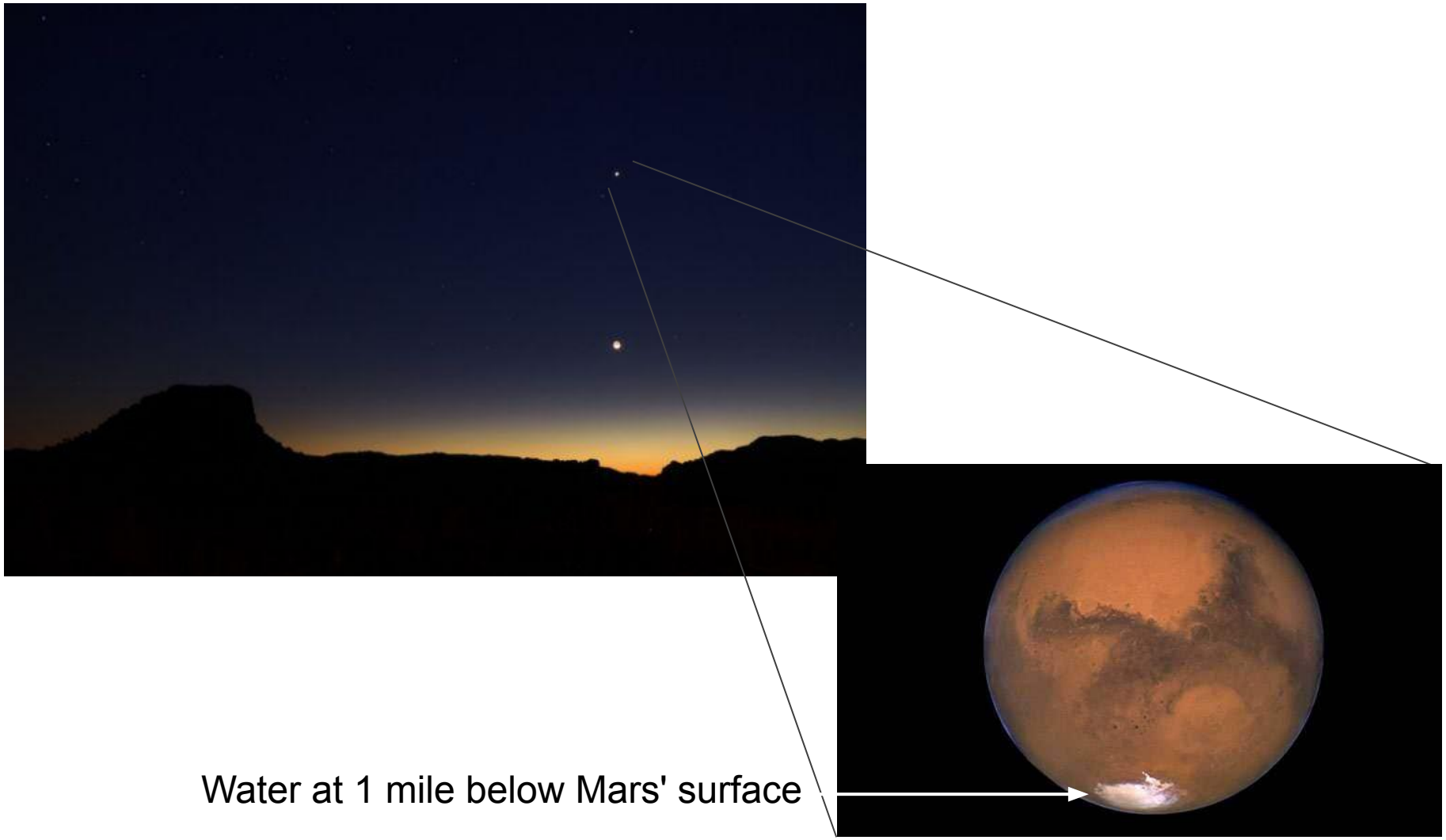
LHCONE: Not all big data traffic is suitable for the general Internet

- As the LHC ramped up to first production operation, ESnet monitoring detected several transatlantic network paths serving the R&E community were being congested
- Finding the cause was not trivial because it turned out to be LHC data analysis groups moving data with GridFTP using dozens of parallel data transfers, so no one end system stood out in the monitoring
- ESnet engaged CERN on how to deal with this, and CERN set up a study group to characterize the problem
- CERN, ESnet, and Internet2 to set up a working group to make recommendations on how to address this issue
  - ESnet engineers proposed a **network overlay approach where the paths used by the overlay were explicitly under control of network operators**
    - In other words, the paths could be easily configured by network engineers not to interfere with general R&E traffic in their domain
    - Access to the overlay was limited to high energy physics projects, which also provided a modicum of security
- The result is called **LHCONE** and carries most of the LHC data worldwide
  - [13] See http://lhcone.web.cern.ch

ESnet

Naked eye view is nice…
But insufficient to understand bottlenecks and flows

Towards an intimate understanding of bottlenecks and flows



Water at 1 mile below Mars' surface

Bottleneck Structure of Google's SDN WAN B4 Network

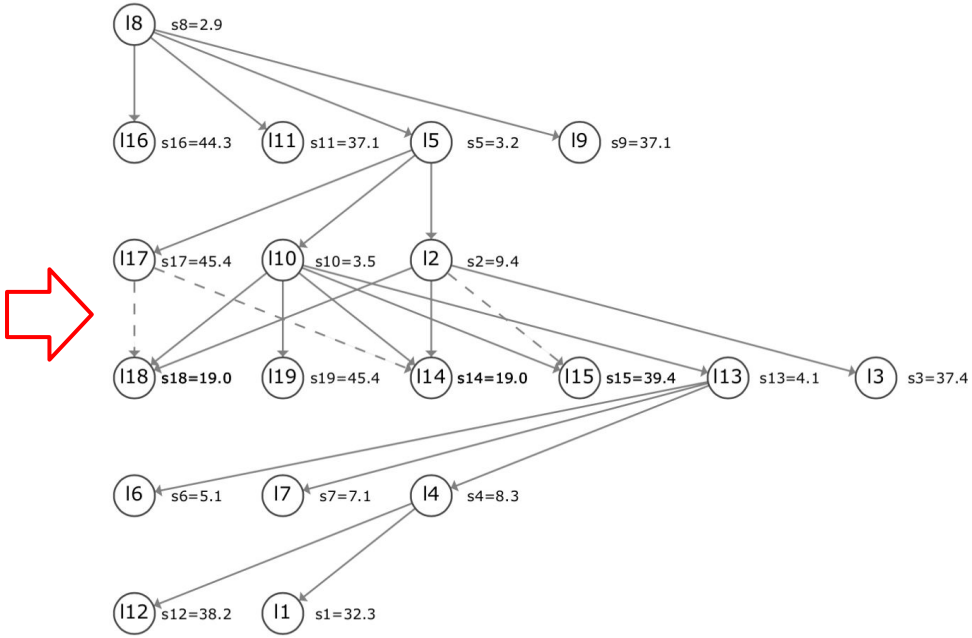
- Google's B4 Network:
  (from ACM SIGCOMM paper)

Bottleneck Structure of Google's SDN WAN B4 Network

- Google's B4 Network:
  (from ACM SIGCOMM paper)

- Bottleneck Structure of B4

  (shortest path full mesh configuration):



Theory of Bottleneck Ordering

Theory of Bottleneck Ordering

LEMMA 2.4. *Bottleneck influence. A bottleneck l can influence the performance of another bottleneck l', i.e., $\partial s_{l'}/\partial c_l \neq 0$, if and only if there exists a set of bottlenecks $\{l_1, l_2, \ldots, l_n\}$ such that l_i is a direct precedent of l_{i+1}, for $1 \leq i \leq n-1$, $l_1 = l$ and $l_n = l'$.*

LEMMA 2.12. *Flow influence. A flow f can influence the performance of another flow f', i.e., $\partial s_{f'}/\partial s_f \neq 0$, if and only if there exists a set of bottlenecks $\{l_1, l_2, \ldots, l_n\}$ such that (1) l_i is a direct precedent of l_{i+1}, for $1 \leq i \leq n-1$, (2) flow f' is bottlenecked at link l_n and (3) flow f goes through l_1.*

LEMMA 2.1. *Minimum convergence time of a distributed congestion control algorithm. Let $\tau(l_i, l_j)$ be a weight assigned to each edge (l_i, l_j) of the BPG graph as follows: (1) If l_i is a direct precedent of l_j, then $\tau(l_i, l_j)$ is the time that it takes for a message to be sent from l_i to l_j; (2) If l_i is an indirect precedent of l_j, then $\tau(l_i, l_j) = \max\{\tau(l_i, l_r) + \tau(l_r, l_j) \mid \text{for any relay link } l_r \text{ between } l_i \text{ and } l_j\}$. Let $l_1 - l_2 - \ldots - l_n$ be a longest path terminating at link l_n according to these weights. Then the minimum convergence time for link l_n is $\sum_{1 \leq i \leq n-1} \tau(l_i, l_{i+1})$.*

Mathematics?
Let's not go there today...

Full details will be presented at ACM SIGMETRICS 2020

ACM SIGMETRICS 2020 for full Math and Algorithms

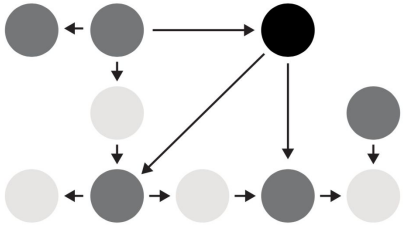
On the Bottleneck Structure of Congestion-Controlled Networks

Jordi Ros-Giralt[1], Sruthi Yellamraju[1], Atul Bohara[1], Harper Langston[1], Richard Lethin[1], Yuang Jiang[2], Leandros Tassiulas[2], Josie Li[3], Malathi Veeraraghavan[3]

[1] Reservoir Labs, 632 Broadway, Suite 803 New York, New York 10012
[2] Yale Institute of Network Science
[3] University of Virginia
{giralt,yellamraju,bohara,langston,lethin}@reservoir.com
{yuang.jiang,leandros.tassiulas}@yale.edu
{jl9gf,mv5g}@virginia.edu

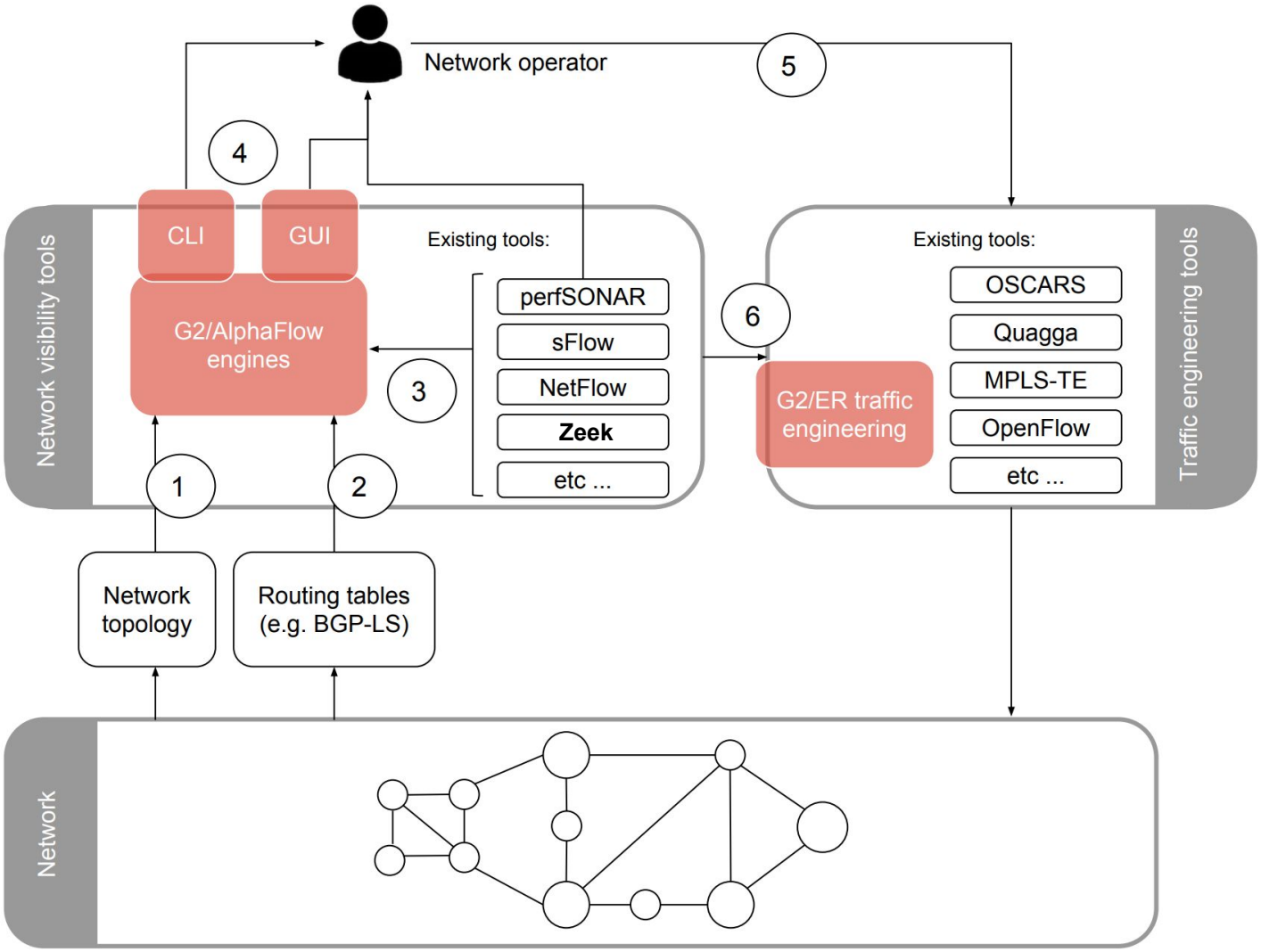
GradientGraph Analytics: Features and Functions

GradientGraph Analytics

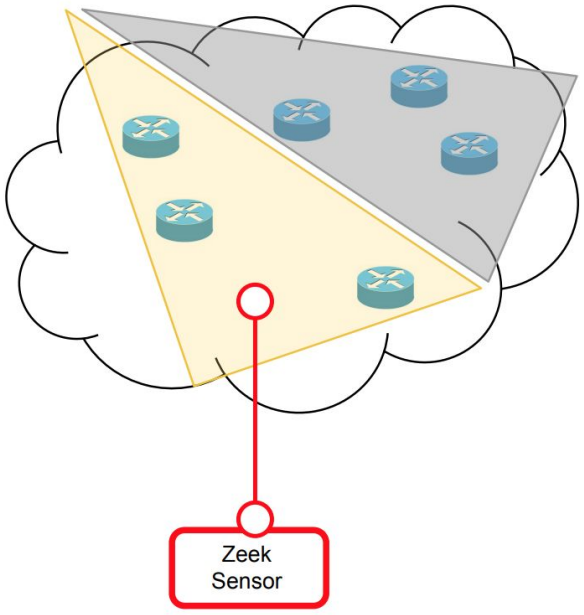


- Interactive analytical dashboards
- Computation of bottleneck structures
- Real-time traffic engineering recommendations
- Offline capacity planning suggestions
- Network performance troubleshooting congestion analysis
- Locating routing misconfigurations
- Replay bottleneck structures

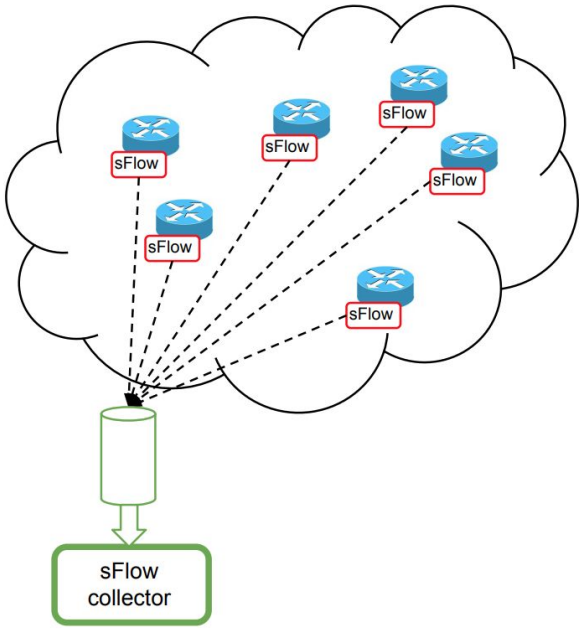
GradientGraph Analytics: Operational Workflow



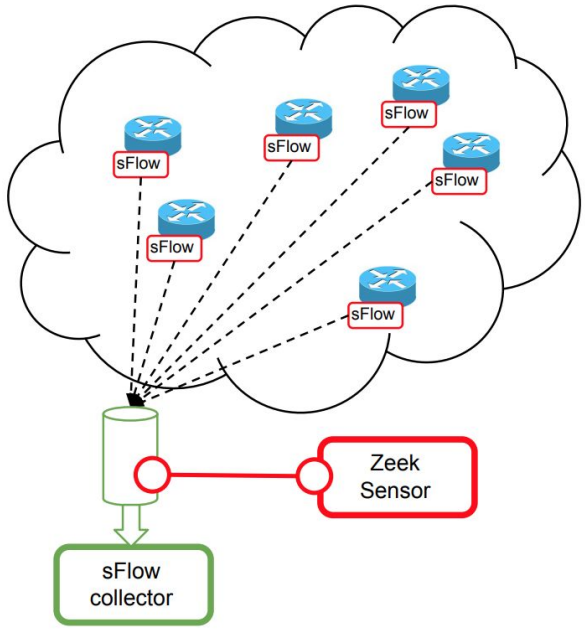
Using Zeek for a (1) Seamless and (2) Scalable Integration with (3) Full Visibility



- Shadow regions
- Scalability challenges



- Disrupts existing operations



- No shadow regions
- Scales well
- No disruption

New sFlow Analyzer

- BinPAC: Really cool domain specific language:
  - Almost like writing an IETF RFC.
  - Full base parser up and running in a few days of work.
  - Great to leverage the 80/20 rule.
- Basic functionality:
  - Populates 'service' field in conn.log with 'sflow' tag.
  - Two new events:
    - sflow_event: Issued for each sFlow datagram
    - sflow_pkt_sample: Issued for each sFlow sample
  - Two new logs:
    - sflow.log: A record for each sFlow datagram
    - sflow_sample.log: A record for each sFlow sample
- We just open sourced it!

  https://github.com/zeek/packages/blob/master/reservoirlabs/bro-pkg.index

New sFlow Analyzer: events.bif

```
## Generated for each sFlow datagram
##
## c: The sFlow connection
## version: sFlow version
## ip_version: agent's IP version (v4 or v6) --
##         see https://www.ietf.org/rfc/rfc3176.txt
## agent_addr: agent's IP address
## subagent_id: Sub-agent ID
## seq_num: This datagram's sequence number
## sys_uptime: System up time
## num_samples: Number of sFlow samples in this datagram
##
event sflow_event%(c: connection,
                   version: count,
                   ip_version: count,
                   agent_addr: count,
                   subagent_id: count,
                   seq_num: count,
                   sys_uptime: count,
                   num_samples: count%);
```

```
## Reports one or more samples from a connection
##
## c: The sFlow connection reporting the samples
## addr_src: Source IP address of the sampled connection
## addr_dst: destination IP address of the sampled connection
## port_src: source port number of the sampled connection
## port_dst: destination port number of the sampled connection
## proto: Transport protocol
## srate: Current sampling rate
## num_samples: number of samples reported for this connection
##
event sflow_pkt_sample%(c: connection,
                        addr_src: count,
                        addr_dst: count,
                        port_src: count,
                        port_dst: count,
                        proto: count,
                        srate: count,
                        num_samples: count%);
```

New sFlow Analyzer: sflow-analyzer.pac

```
refine flow SFLOW_Flow += {
        function proc_sflow_message(msg: SFLOW_PDU): bool
                %{
                // Report first the general sflow event
                BifEvent::generate_sflow_event(connection()->bro_analyzer(),
                                        connection()->bro_analyzer()->Conn(),
                                        msg->header()->version(),
                                        msg->header()->ip_version(),
                                        msg->header()->agent_addr(),
                                        msg->header()->subagent_id(),
                                        msg->header()->seq_num(),
                                        msg->header()->sys_uptime(),
                                        msg->header()->num_samples());
(...)
                for (int i = 0; i < msg->samples()->size(); i++) {
(...)
                                BifEvent::generate_sflow_pkt_sample(
                                            connection()->bro_analyzer(),
                                        connection()->bro_analyzer()->Conn(),
                                        addr_src,
                                        addr_dst,
                                        port_src,
                                        port_dst,
                                        ip_pkt->ip_hdr()->proto(),
                                            fsample->srate(),
                                        1);
```

New sFlow Analyzer: sflow-protocol.pac

```
(...)

type FLOW_SAMPLE = record {
        sample_seqnum: uint32;
        src_type_idx: uint32;
        srate: uint32;
        spool: uint32;
        pkt_drops: uint32;
        snmp_if_in: uint32;
        snmp_if_out: uint32;
        num_flow_rec: uint32;
        flow_recs: FLOW_RECORDS(num_flow_rec);
};

(...)

type SFLOW_SAMPLE = record {
        enterprise_format: uint32;
        sample_len: uint32;
        # The last 12 bits of enterprise_format determine the format
        sample_body: case enterprise_format & 0xfff of {
                1 -> flow_sample: FLOW_SAMPLE;
                2 -> count_sample: UNSUPPORTED_SAMPLE(sample_len);
                default -> unsupported_sample: UNSUPPORTED_SAMPLE(sample_len);
        };
};

type SFLOW_SAMPLES(nsamples: uint32) = SFLOW_SAMPLE[nsamples];

type SFLOW_PDU(is_orig: bool) = record {
        header: SFLOW_HEADER;
        samples: SFLOW_SAMPLES(header.nsamples);
} &byteorder=bigendian;
```

New sFlow Analyzer: conn.log proto field

```
[rscope-logs] /rscope_logs/logs# column -t ./current/conn.log | less -S

#separator         \x09
#set_separator     ,
#empty_field       (empty)
#unset_field       -
#path              conn
#open              2019-10-09-02-52-16
#fields            ts                  uid                 id.orig_h  id.orig_p  id.resp_h  id.resp_p  proto  service  duration  orig_bytes  resp_bytes  conn_state  local_orig  local_resp
#types             time                string              addr       port       addr       port       enum   string   interval  count       count       string      bool        bool
1570614672.117856  CMgQscmWr0fCGtQu    fe80::f87b:bff:fe5e:c79d  133        ff02::2    134        icmp   -        3.999985  16          0           OTH         F           F           0
1570614717.632911  CV4KU5zkYFHuyThv9   10.1.1.250                 41224      10.1.1.224 6343       udp    sflow    6.999701  26304       0           S0          T           T           0
```

New sFlow Analyzer: sflow.log

```
[rscope-logs] /rscope_logs/logs# column -t ./current/sflow.log | less -S

#separator         \x09
#set_separator     ,
#empty_field       (empty)
#unset_field       -
#path              sflow
#open              2019-10-09-02-51-59
#fields            ts        uid          id.orig_h   id.orig_p  id.resp_h   id.resp_p  version  ip_version  agent_addr    subagent_id  seq_num  sys_uptime  num_samples
#types             time      string       addr        port       addr        port       count    count       addr          count        count    count       count
1570614719.632824  CV4KU5zkYFHuyThv9  10.1.1.250  41224      10.1.1.224  6343       5        1           10.1.1.250    0            1384     1521000     1
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250  41224      10.1.1.224  6343       5        1           10.1.1.250    0            1385     1521000     8
1570614719.981697  CV4KU5zkYFHuyThv9  10.1.1.250  41224      10.1.1.224  6343       5        1           10.1.1.250    0            1386     1521000     8
1570614720.010899  CV4KU5zkYFHuyThv9  10.1.1.250  41224      10.1.1.224  6343       5        1           10.1.1.250    0            1387     1521000     8
1570614720.097272  CV4KU5zkYFHuyThv9  10.1.1.250  41224      10.1.1.224  6343       5        1           10.1.1.250    0            1388     1521000     8
1570614720.154397  CV4KU5zkYFHuyThv9  10.1.1.250  41224      10.1.1.224  6343       5        1           10.1.1.250    0            1389     1521000     8
1570614720.210440  CV4KU5zkYFHuyThv9  10.1.1.250  41224      10.1.1.224  6343       5        1           10.1.1.250    0            1390     1521000     8
```

New sFlow Analyzer: sflow_sample.log

```
[rscope-logs] /rscope_logs/logs# column -t ./current/sflow_sample.log | less -S

#separator         \x09
#set_separator     ,
#empty_field       (empty)
#unset_field       -
#path              sflow_sample
#open              2019-10-09-02-51-59
#fields            ts                 uid          id.orig_h   id.orig_p  id.resp_h   id.resp_p  addr_src    addr_dst    port_src  port_dst  proto  srate  num_samples
#types             time               string       addr        port       addr        port       addr        addr        count     count     count  count  count
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.251  10.1.1.223  80        1024      6      16     1
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.223  10.1.1.251  1027      80        6      16     1
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.251  10.1.1.223  80        1027      6      16     1
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.223  10.1.1.251  1030      80        6      16     1
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.223  10.1.1.251  1030      80        6      16     1
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.251  10.1.1.223  80        1033      6      16     1
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.223  10.1.1.251  1030      80        6      16     1
1570614719.879784  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.251  10.1.1.223  80        1033      6      16     1
1570614719.981697  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.223  10.1.1.251  1033      80        6      16     1
1570614719.981697  CV4KU5zkYFHuyThv9  10.1.1.250   41224       10.1.1.224  6343       10.1.1.223  10.1.1.251  1033      80        6      16     1
```

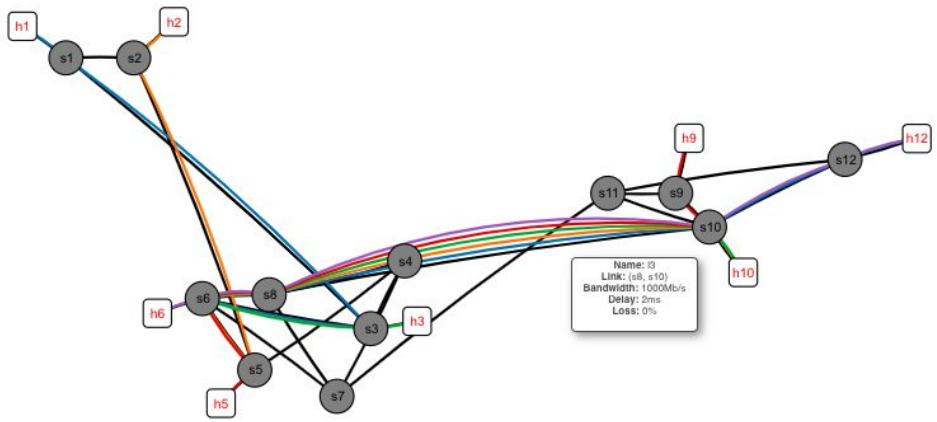
GradientGraph Analytics Platform

Reservoir Labs, Inc.

Network Topology Dashboard

Dashboard

Topology Panel



Toggle Filter | Toggle Flows | Reload Flows | Reload All

Switches

Switches | Hosts | Links | Flows

Type in element to search

Show 10 entries

ID	Links Attached
s1	(s1 , s3) (s1 , s2)
s10	(s9 , s10) (s8 , s10) (s10 , s12) (s10 , s11)
s11	(s9 , s11) (s10 , s11) (s7 , s11) (s11 , s12)
s12	(s10 , s12) (s11 , s12)
s2	(s2 , s5) (s1 , s2)
s3	(s3 , s6) (s3 , s4) (s1 , s3)
s4	(s3 , s4) (s4 , s8) (s4 , s7) (s4 , s5)
s5	(s2 , s5) (s5 , s6) (s4 , s5)
s6	(s3 , s6) (s6 , s8) (s6 , s7) (s5 , s6)
s7	(s7 , s11) (s7 , s8) (s6 , s7) (s4 , s7)

Showing 1 to 10 of 12 entries

Previous | 1 | 2 | Next

Dashboard Configuration

Automated Refresh

Off

Refresh Rate (s)

10

Time to refresh (positive integer)

Overlap Flows

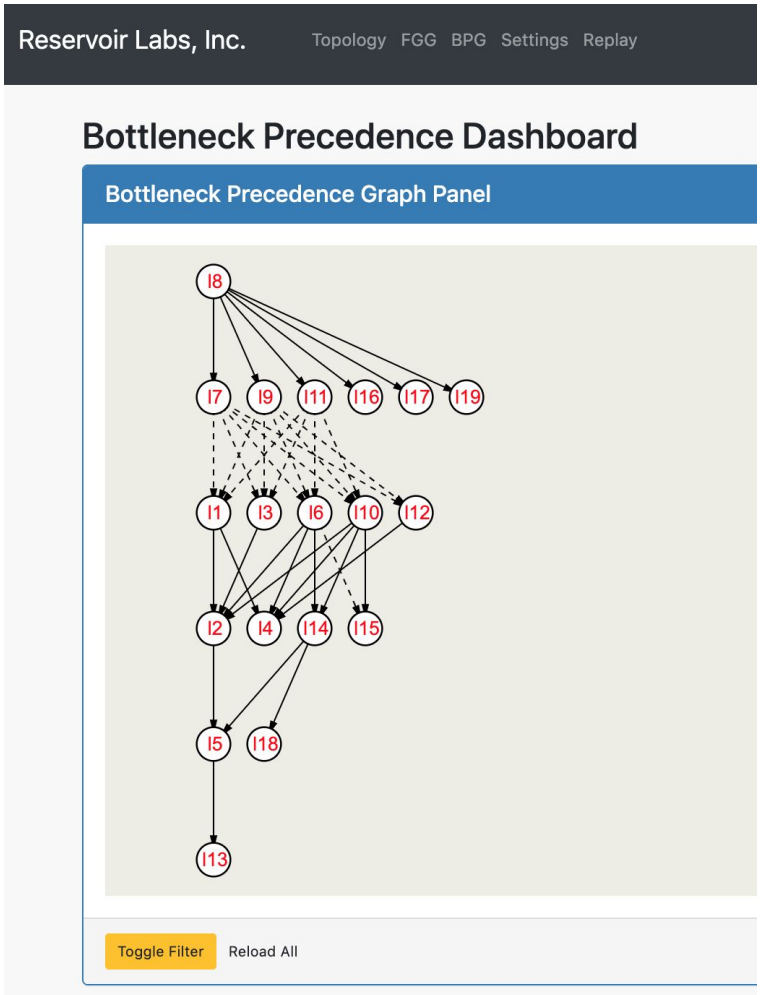
No

Export Configuration

Apply

Automated Refresh Stops

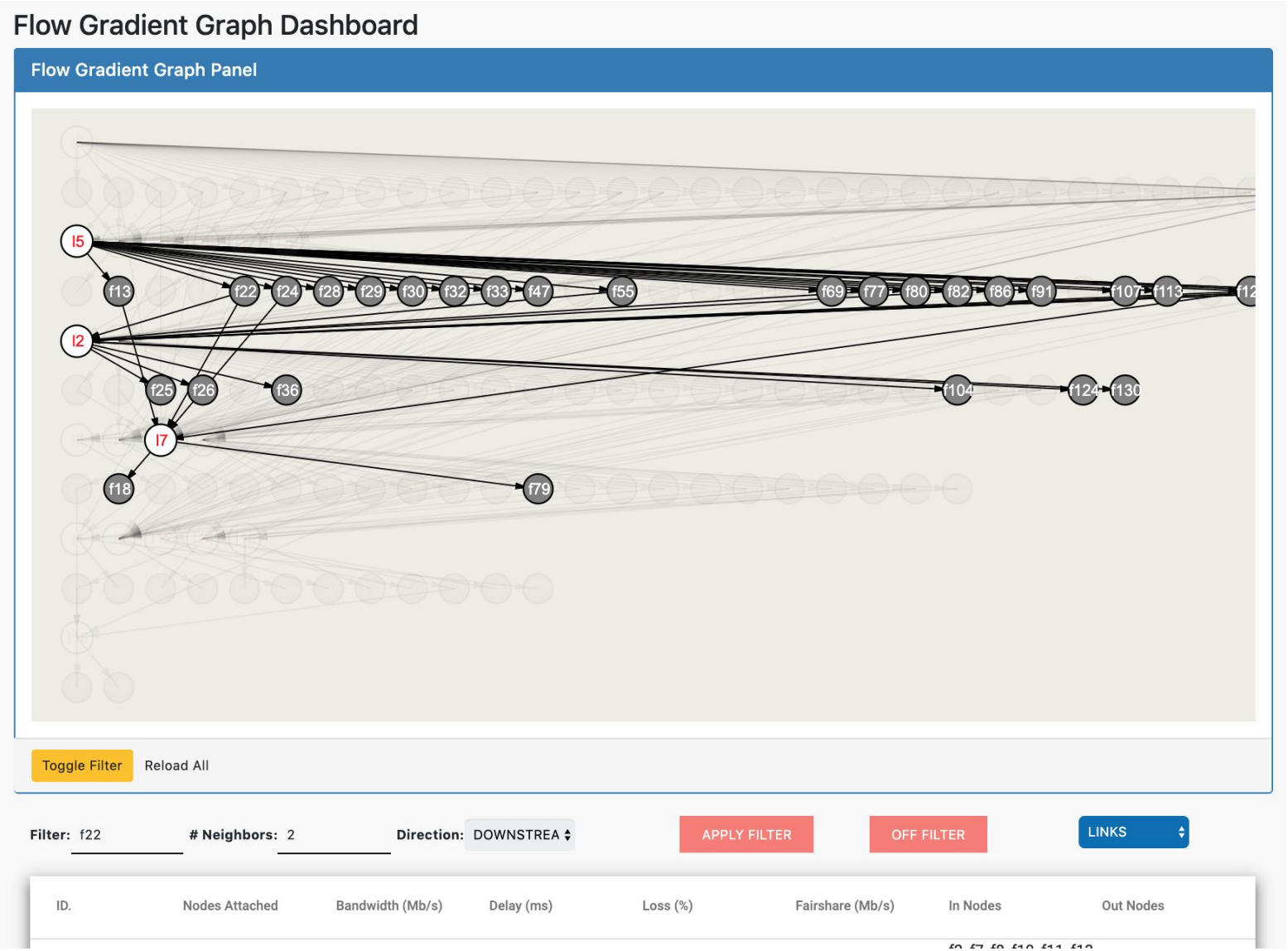
GradientGraph Analytics Platform



Filter: I1 # Neighbors: 2 APPLY FILTER OFF FILTER

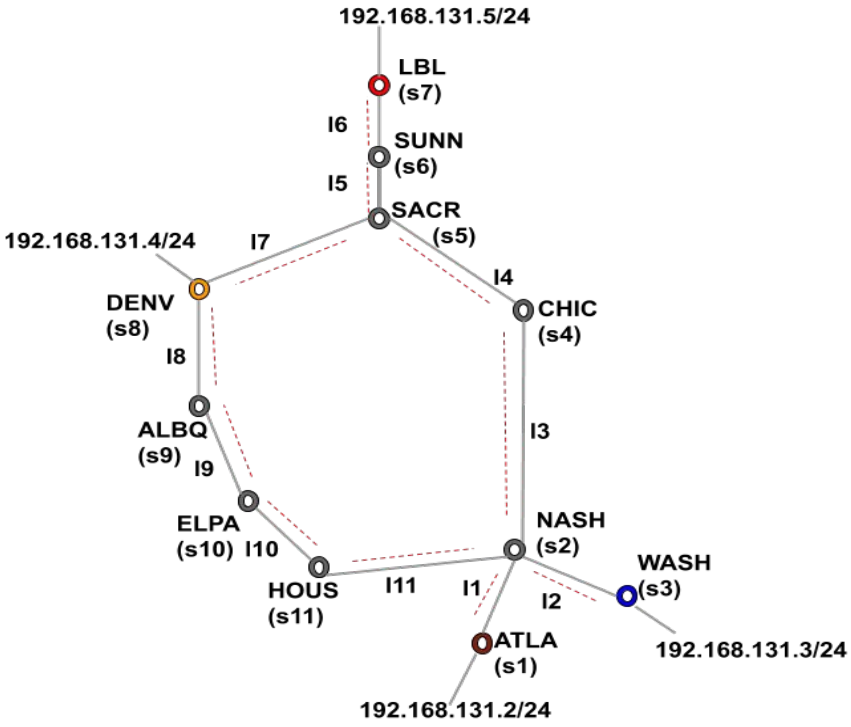
ID.	Nodes Attached	Bandwidth (Mb/s)	Delay (ms)	Loss (%)	Fairshare (Mb/s)	In Nodes	Out Nodes
I1	(s1 , s3)		2ms	0	31.54	I7(i), I9(i), I11(i)	I4, I2
I2	(s1 , s2)		2ms	0	9.57	I1, I3, I6, I10	I5
I3	(s6 , s8)		10ms	1	37.08	I7(i), I9(i), I11(i)	I2
I4	(s4 , s8)		2ms	0	8.89	I1, I6, I10, I12	None
I5	(s2 , s5)		2ms	0	3.35	I2, I14	I13
I6	(s4 , s5)		2ms	0	7.33	I7(i), I9(i), I11(i)	I14, I4, I2, I15(i)
I7	(s5 , s6)		2ms	0	4.50	I8	I6(i), I10(i), I3(i), I12(i), I1(i)
I8	(s3 , s6)		2ms	0	2.63	None	I19, I16, I17, I11, I7, I9

GradientGraph Analytics Platform



Thank you!

Come see GradientGraph running live from the ESnet Testbed (outside at the Reservoir Labs table)



ESnet
100Gbps Testbed Network