sFlow Tutorial 6NRP Jan 28 2025

Agenda:

1. Introduction to sFlow
2. Hands on deployment of open source host-sflow agent
3. Passive TCP delay, loss and jitter measurements
4. Network-wide packet drop analysis

sFlow Tutorial 6NRP Jan 28 2025

Agenda:

1. Introduction to sFlow
2. Hands-on deployment of sFlow tool-chain
3. Passive TCP delay, loss and jitter measurements
4. Network-wide packet drop analysis

sFlow Data Model

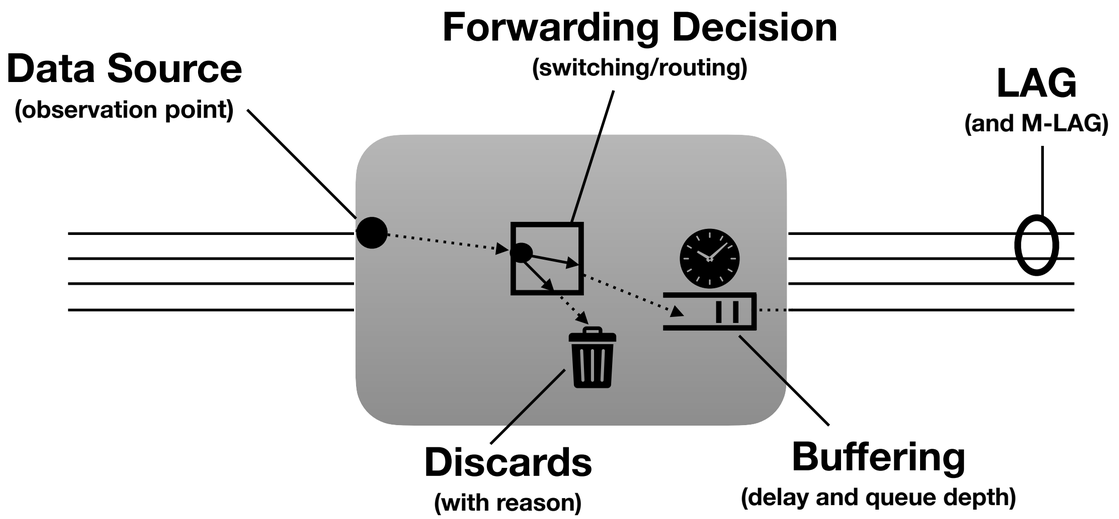
Observability and Controlability

1. **Model the system.**
2. **Decide on key metrics. No hidden state.**
3. **Detail: measurements must be actionable.**
4. **Latency: measurements must be timely.**
5. **Accuracy: measurements must be accurate (enough).**
6. **Use efficient, extensible protocol.**

sFlow designed by control engineer, not computer scientist!

sFlow Data Model

sFlow models a node in a packet-switched network as a "black box" with ports, forwarding and buffering.

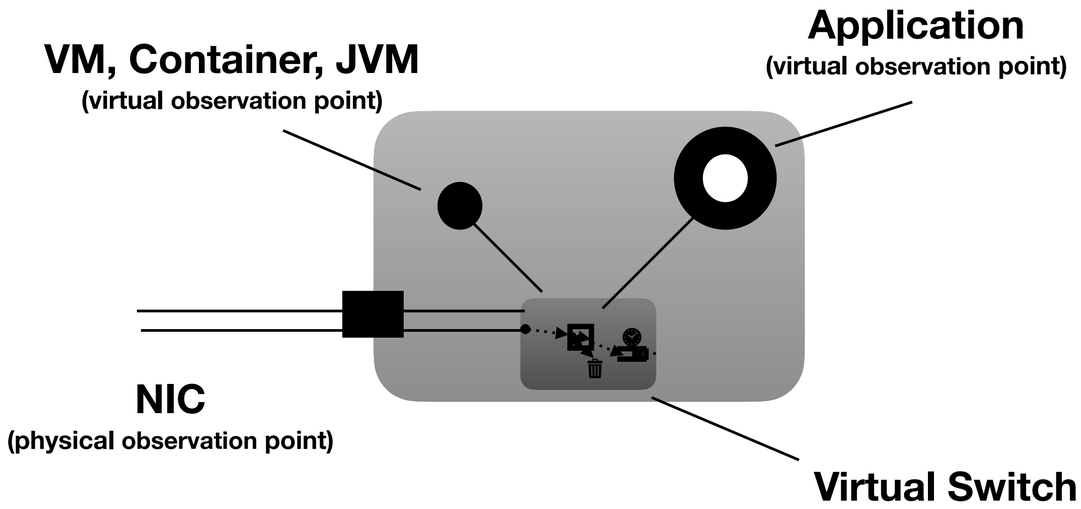


An sFlow data source sends:

1. Counters (periodic) - interface counters, CPU/RAM counters, LAG info, ...
2. Packet headers (1:N random sampled) - with in/out ports + forwarding + buffering
3. Dropped packets (rate-limted) - with drop-reason

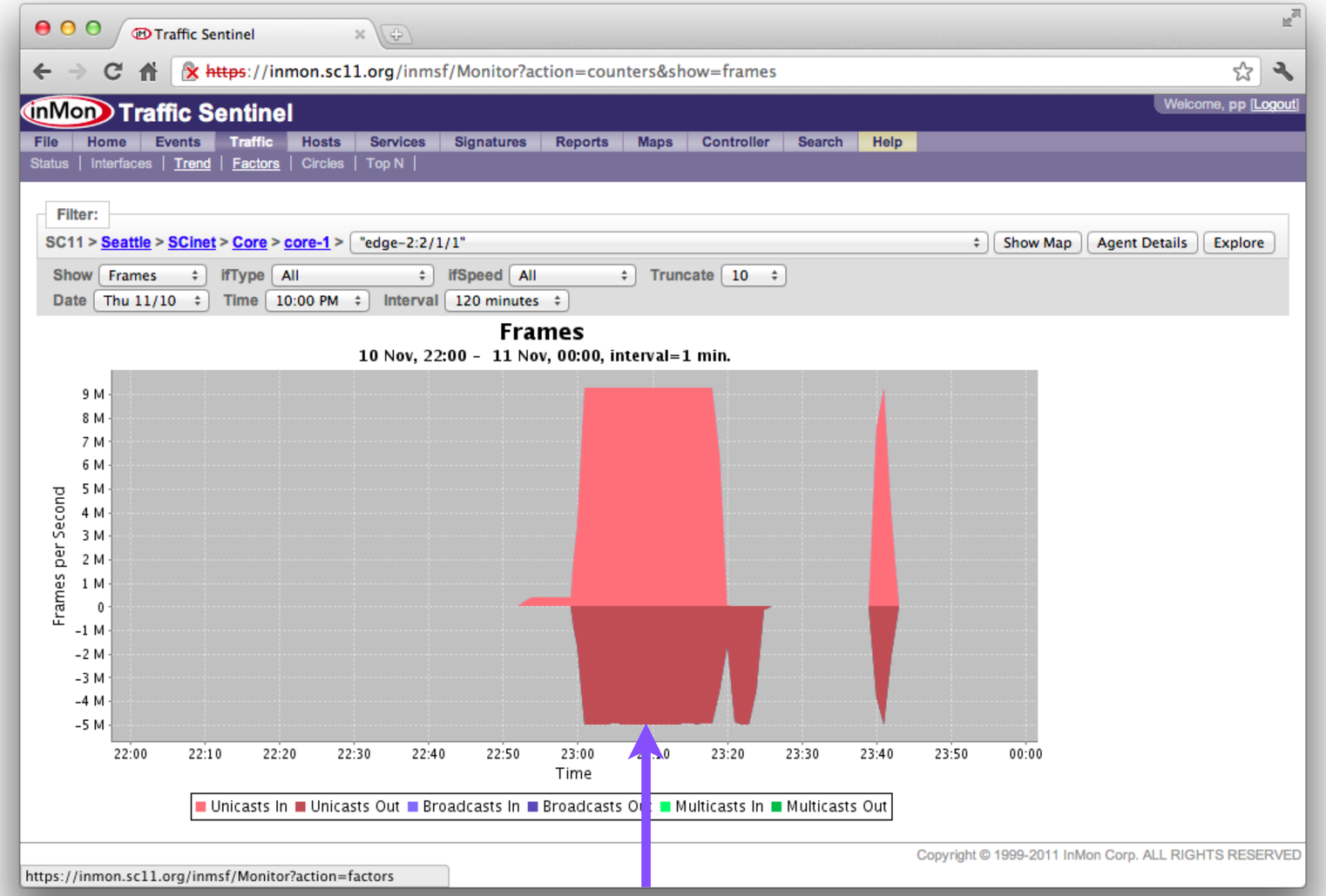
sFlow Data Model

Within a node, virtual data-sources (containers, VMs, JVMs, applications) can also be modeled:



A VM data source can send CPU/mem/IO counters (e.g. Kubernetes Pod).
An application data source can send 1:N client-server transaction samples (e.g. Web Server).
Packet-samples can be annotated with the identity of their virtual source or destination.

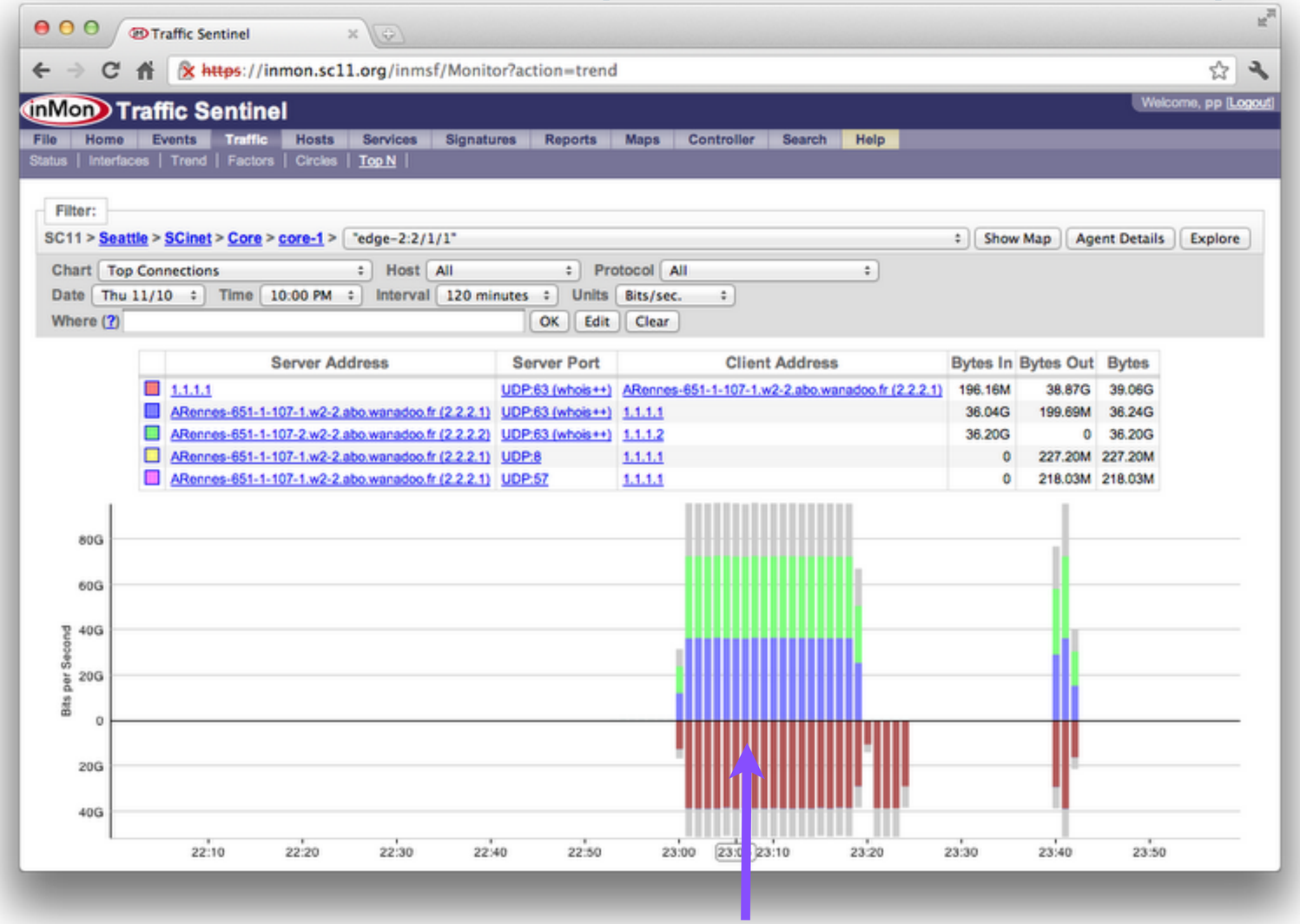
Counters are not enough



Why the spike in traffic?

(100Gbit link carrying 14,000,000 packets/second)

sFlow also exports random samples



Break out traffic by client, server and port

(graph based on samples from100Gbit link carrying 14,000,000 packets/second)

Important properties

Important properties:

1. Scalable
2. Simple
3. Real time

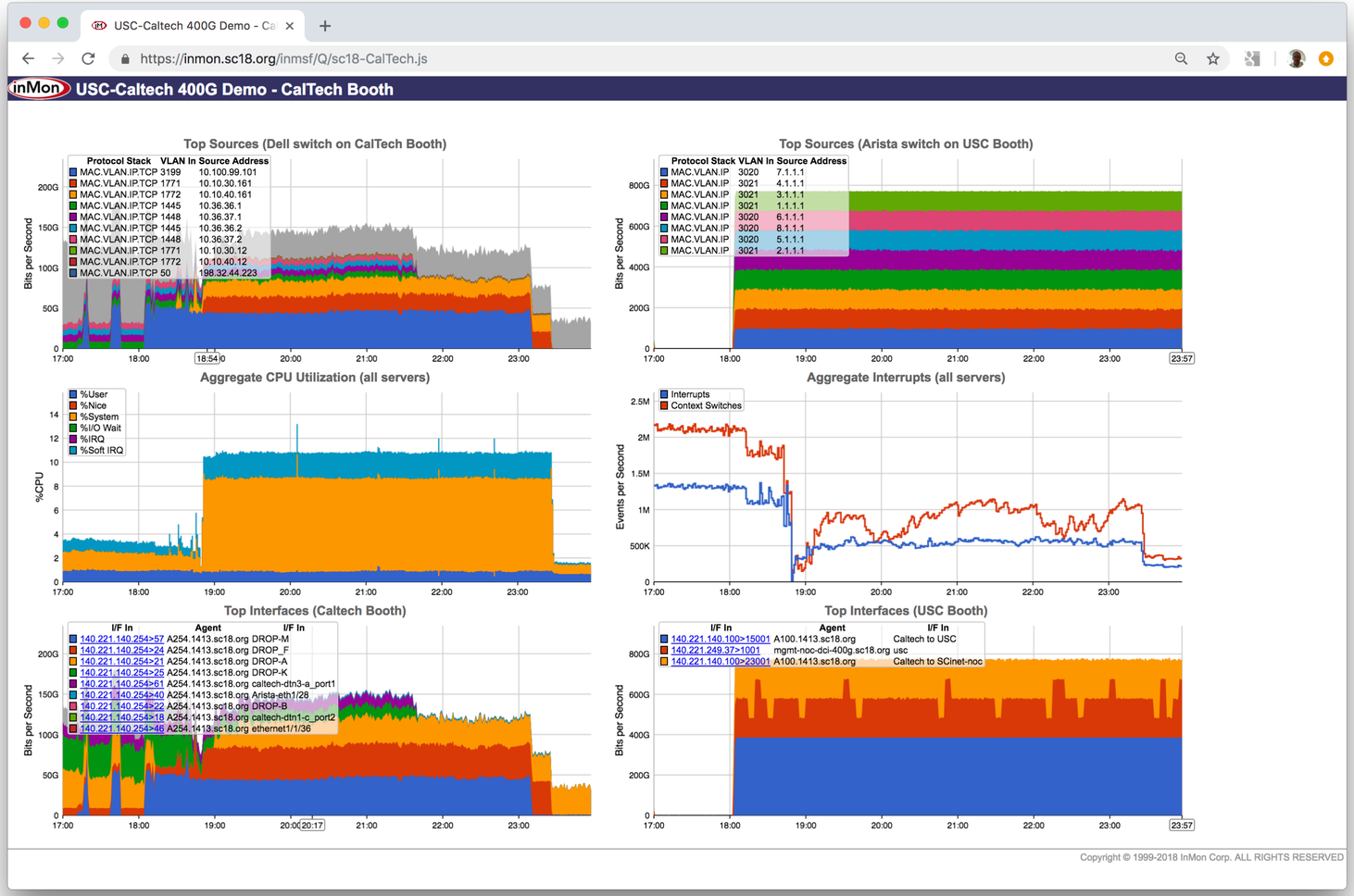
Important properties

Important properties:

1. Scalable
2. Simple
3. Real time

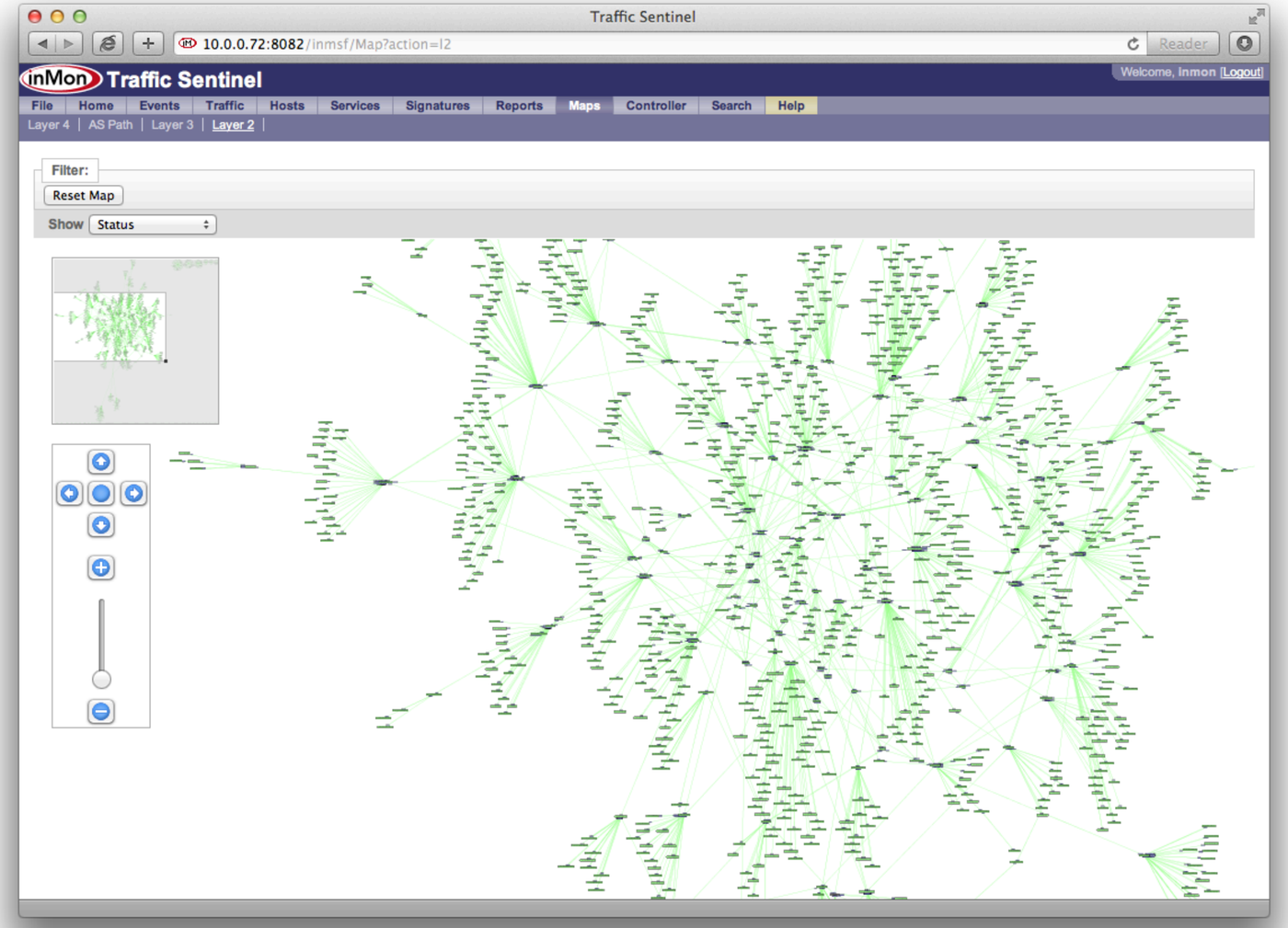
Scalability - link speed

800G?
No problem



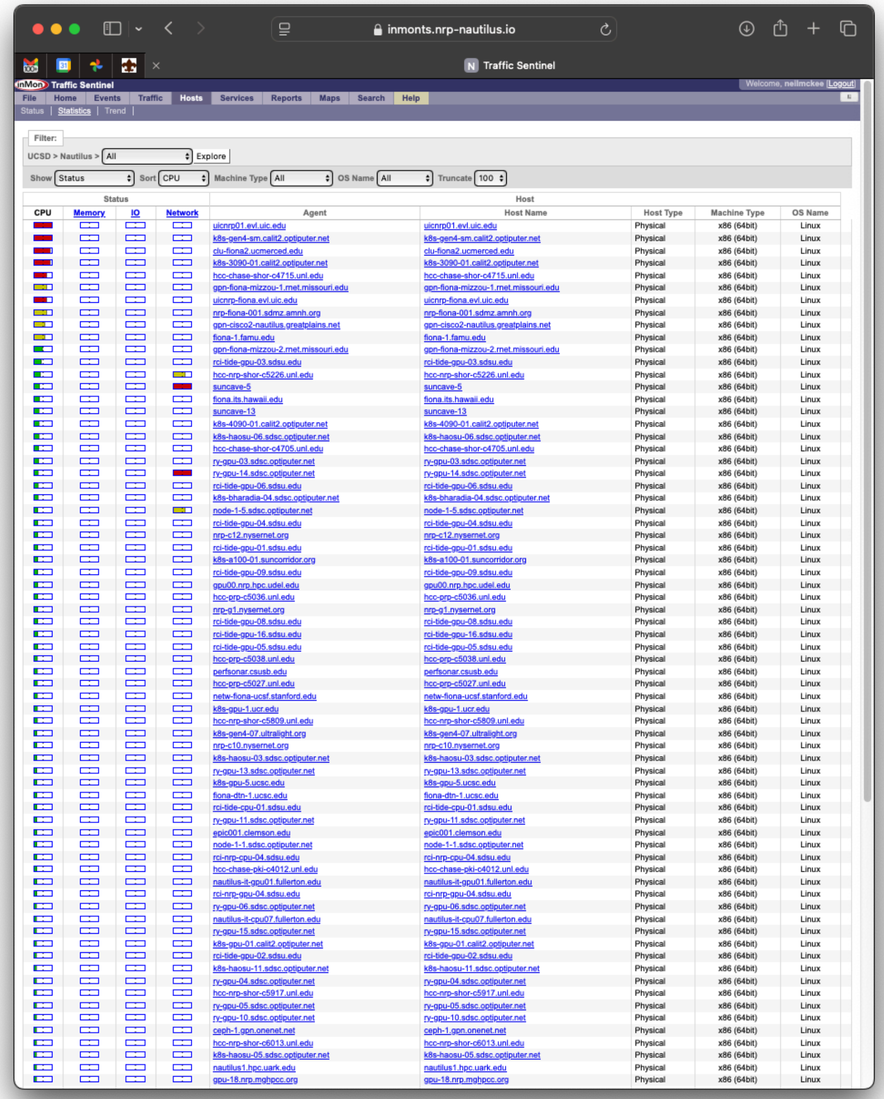
Scalability - Network

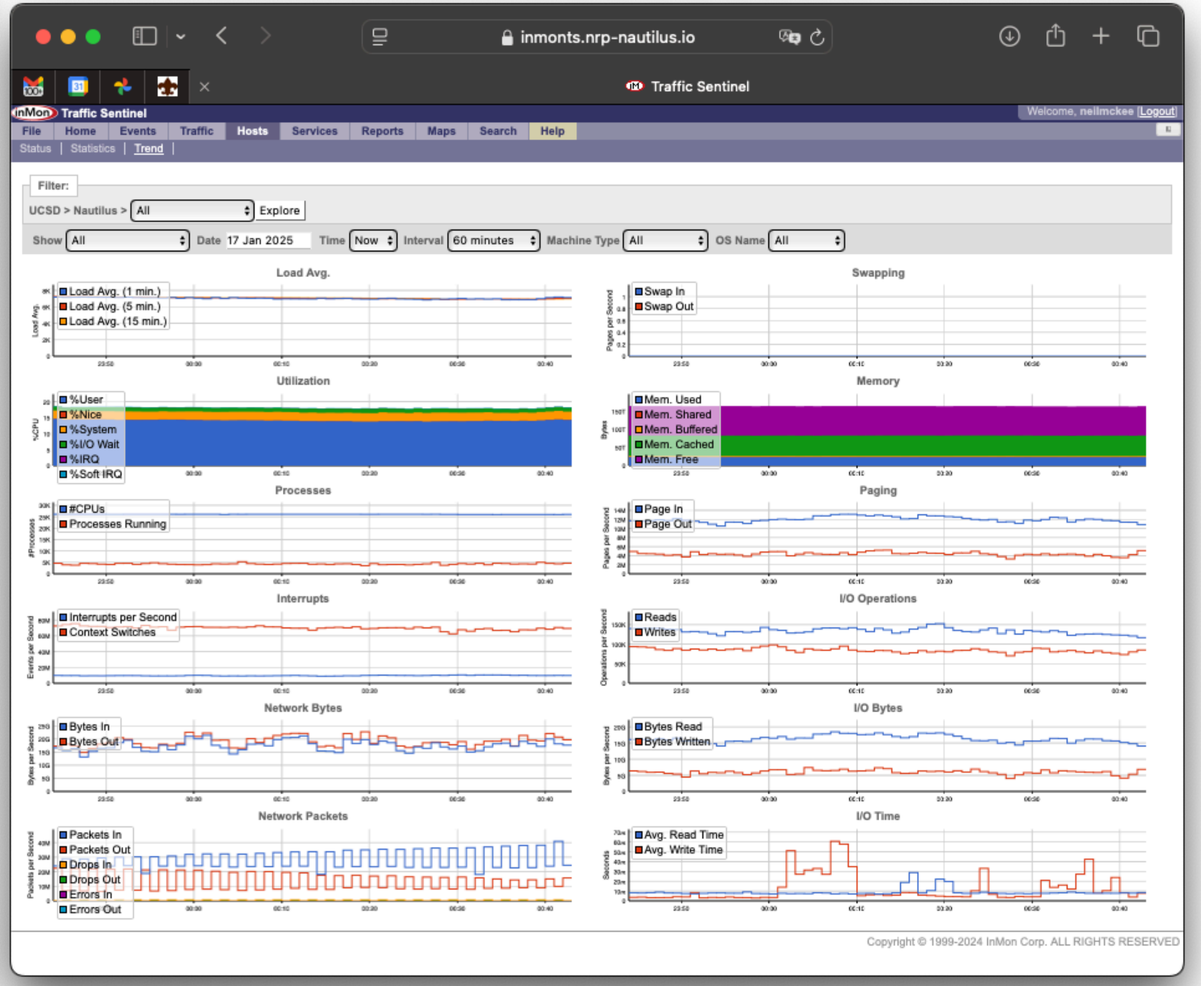
4000 switches?
No problem



Scalability - Servers

10000 servers?
No problem





Important properties

Important properties:

1. Scalable
2. Simple
3. Real time

Config - Network

SONiC

```
sudo config feature state sflow enabled
sudo config sflow collector add my-collector 10.1.2.3
```

Arista

```
sflow sample 10000
sflow polling-interval 20
sflow vrf mgmt destination 10.1.2.3
sflow vrf mgmt source-interface Management0
sflow sample rewrite dscp
sflow run
sflow extension bgp
```

For more examples, see https://github.com/sflow/config

Config - Server

Nautilus hsflowd.conf

```
sflow {
  agent.cidr = !172.16.0.0/12
  agent.cidr = !192.168.0.0/16
  agent.cidr = !10.0.0.0/8
  agent.cidr = !::/0
  collector {ip = 10.111.42.41}
  collector {ip = 10.109.4.240}
  collector {ip = 10.97.26.74}
  sampling.1G = 2000
  sampling.10G = 5000
  sampling.40G = 10000
  sampling.100G = 20000
  sampling.200G = 50000
  sampling.400G = 80000
  sampling.800G = 100000
  sampling.1200G = 100000
 headerBytes=256
  pcap {speed=1G-}
  tcp { tunnel=on }
  k8s { markTraffic=on eof=on }
  nvml {}
  dropmon { limit=50 max=30000 hide=dev_kfree_skb_any }
}
```

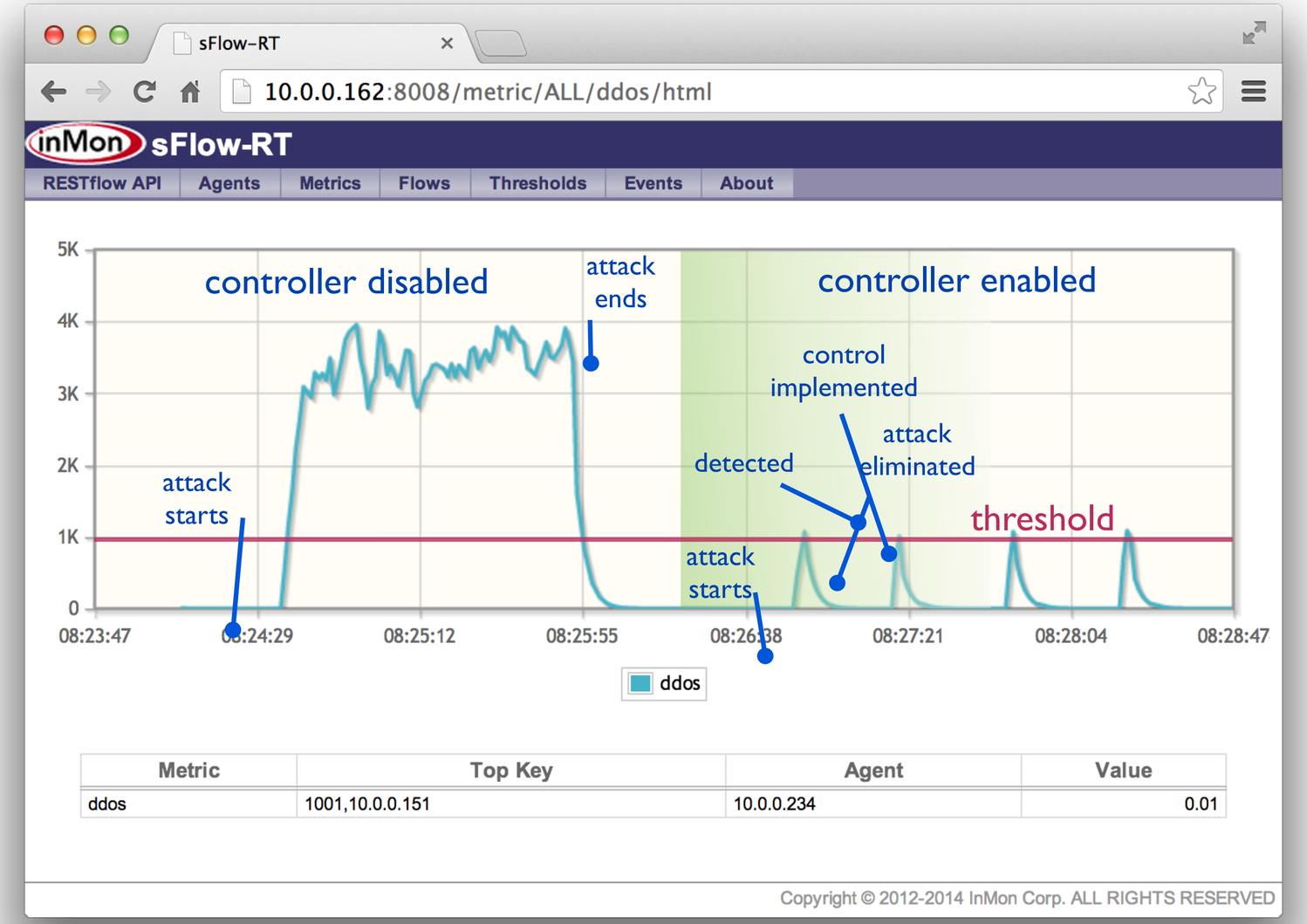
Same config applies to over 400 agents

Important properties

Important properties:

1. Scalable
2. Simple
3. Real time

Real-time control with sFlow-RT

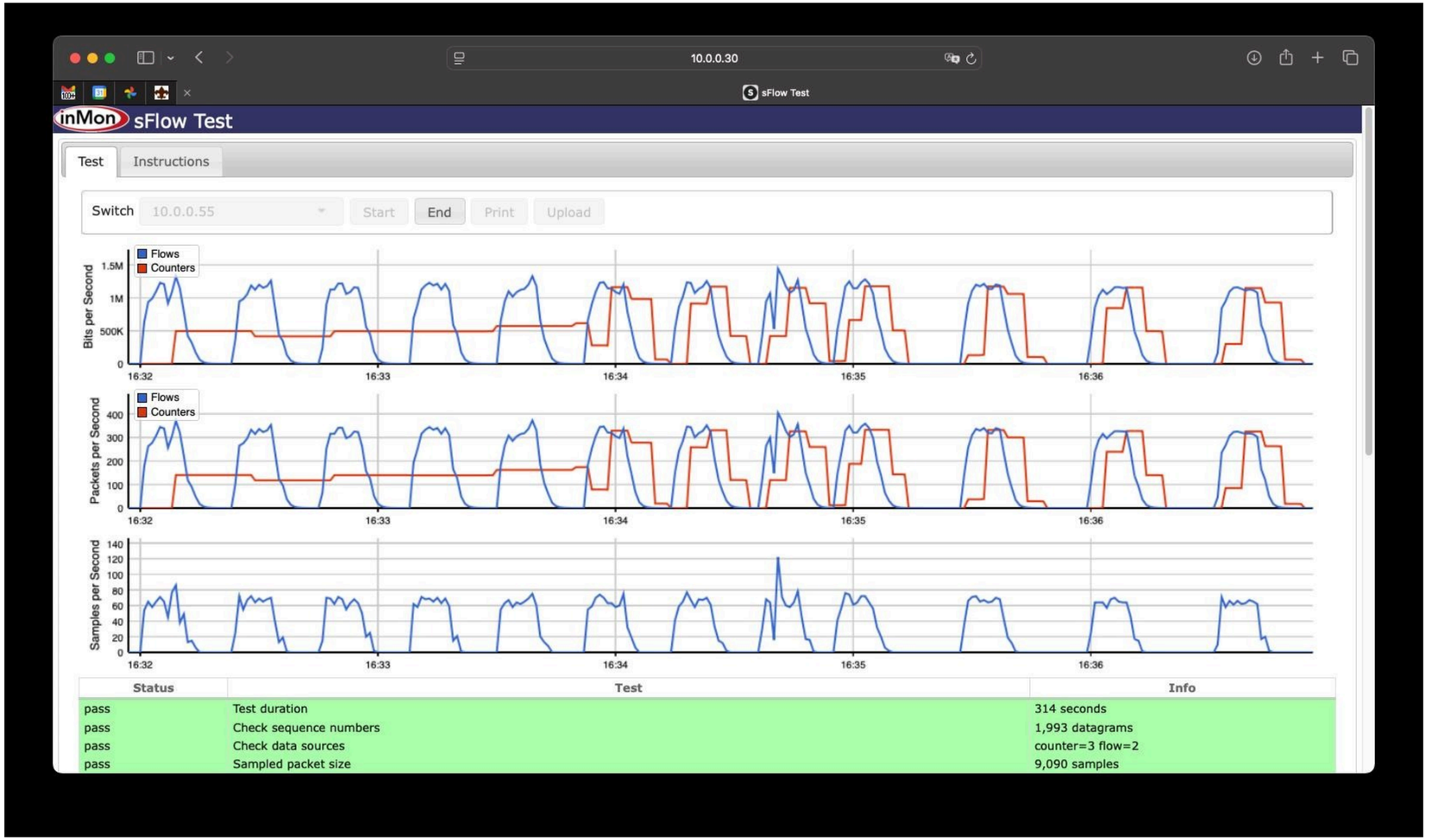


Real-time control with sFlow-RT

15 second oscillation in 400G RoCE2 stream (when competing with other traffic)

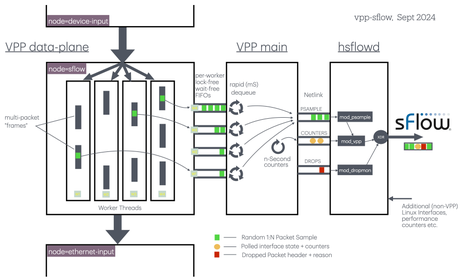
Real-time control with sFlow-RT

sFlow-RT = scriptable real-time analytics engine
(sFlow, BGP, DNS, OpenFlow, REST, FlowSpec)



10-second traffic bursts.
5-second counter polling (red),
but packet sampling (blue)
still much lower latency.

(example is vpp-sflow)



Optional Extensions

Options:

1. Headers of dropped packets
2. TCP delay, loss and jitter
3. Optical link metrics
4. Transit delay and queue depth

Optional Extensions

Options:

1. Headers of dropped packets
2. TCP delay, loss and jitter
3. Optical link metrics
4. Transit delay and queue depth

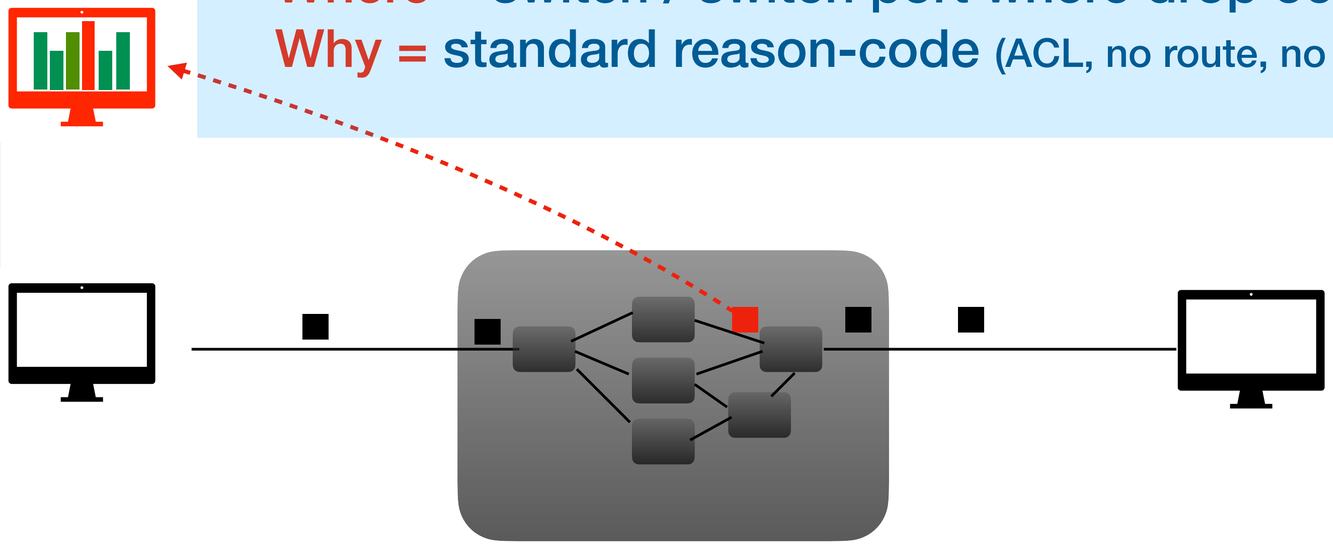
Headers of Dropped Packets

sFlow packet drop notification

What = header of dropped packet (with MAC, VLAN, IP...)

Where = switch / switch port where drop occurred

Why = standard reason-code (ACL, no route, no buffer, MTU,...)



Linux servers running hsflowd report same standard measurement (packets dropped in kernel stack)

Headers of Dropped Packets

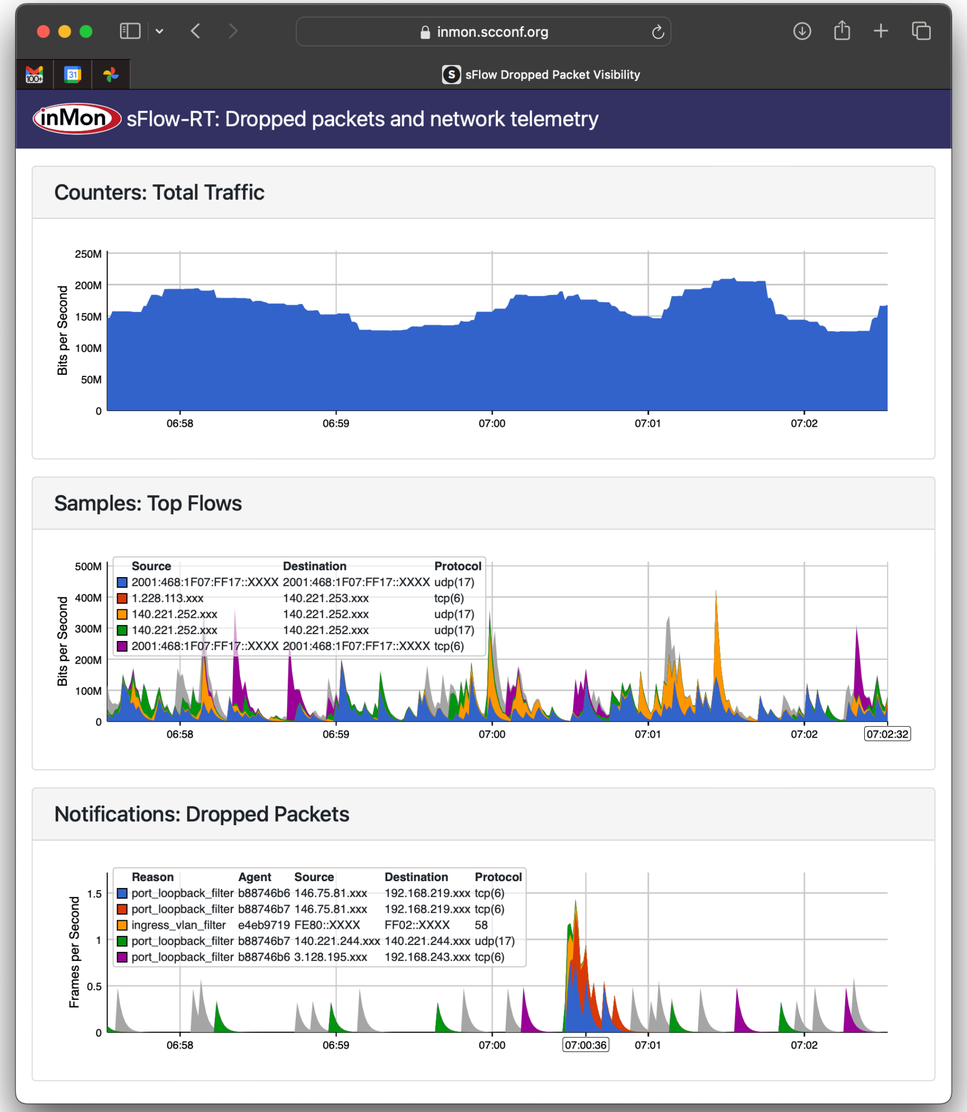
sflowtool output:

```
startSample ----------------------
sampleType_tag 0:5
sampleType DISCARD
sampleSequenceNo 1179686
sourceId 0:2
dropEvents 60445424
inputPort 2
outputPort 0
discardCode 256
discardReason unknown
discarded_flowBlock_tag 0:1038
discarded_extendedType function
discarded_symbol stp_pdu_rcv+0x90/0xa0 [stp]
discarded_flowBlock_tag 0:1
discarded_flowSampleType HEADER
discarded_headerProtocol 1
discarded_sampledPacketSize 53
discarded_strippedBytes 4
discarded_headerLen 53
discarded_headerBytes 01-80-C2-00-00-00-5C-5E-
AB-72-93-24-00-27-42-42-03-00-00-02-02-0E-12-0A-00-01-02-03-04-05-00
-00-07-D0-F2-0A-5C-5E-AB-72-93-01-82-22-02-00-14-00-02-00-0F-00-00
discarded_dstMAC 0180c2000000
discarded_srcMAC 5c5eab729324
endSample   ----------------------
```

packet header decoded at collector

Headers of Dropped Packets

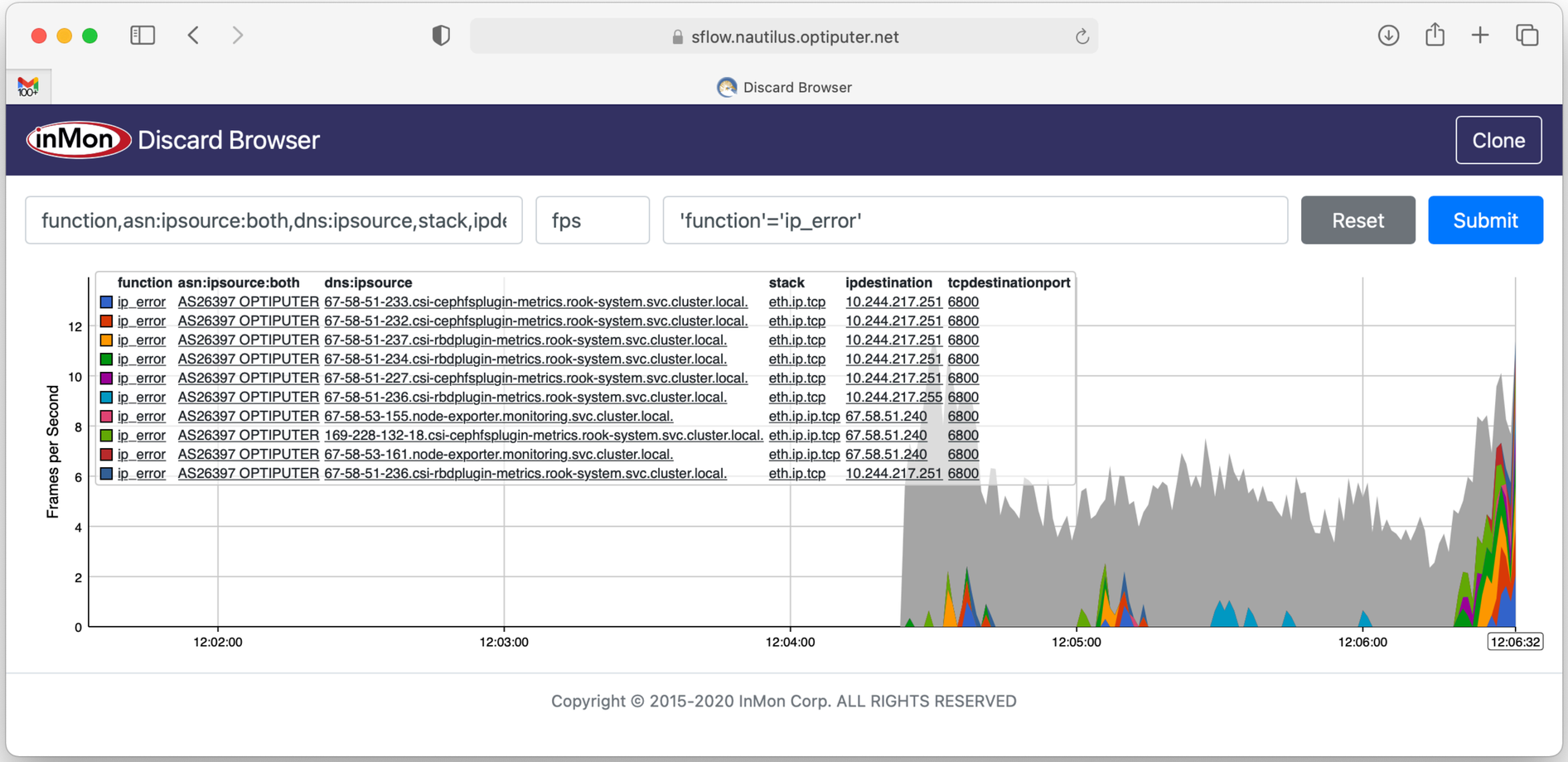
Hardware switch:



Arista switch (Broadcom ASIC) at SC23 in Denver

Headers of Dropped Packets

Linux server:



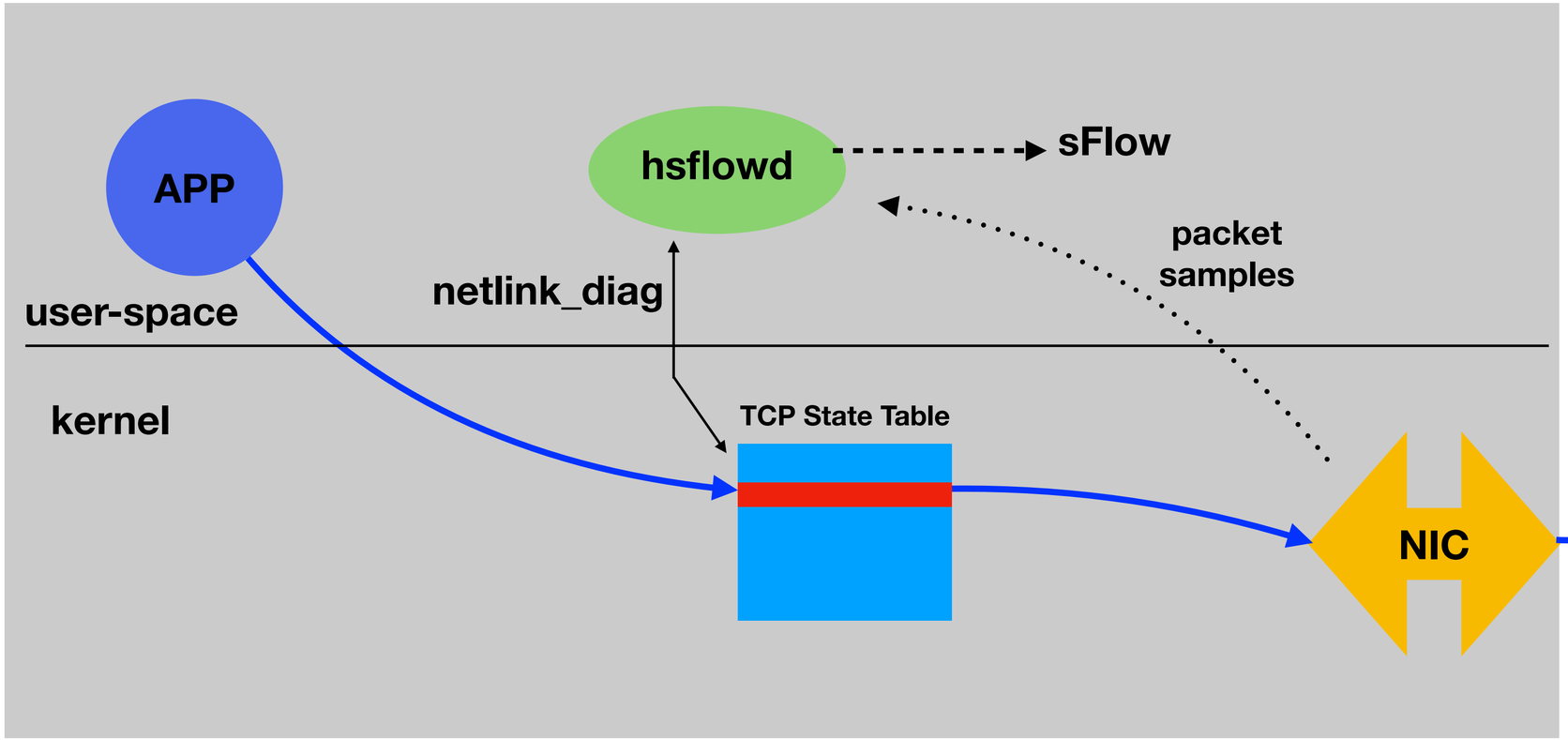
Optional Extensions

Options:

1. Headers of dropped packets
2. TCP delay, loss and jitter
3. Optical link metrics
4. Transit delay and queue depth

TCP Delay, loss and jitter

Now hsflowd can annotate packet samples with netlink tcp_diag info…



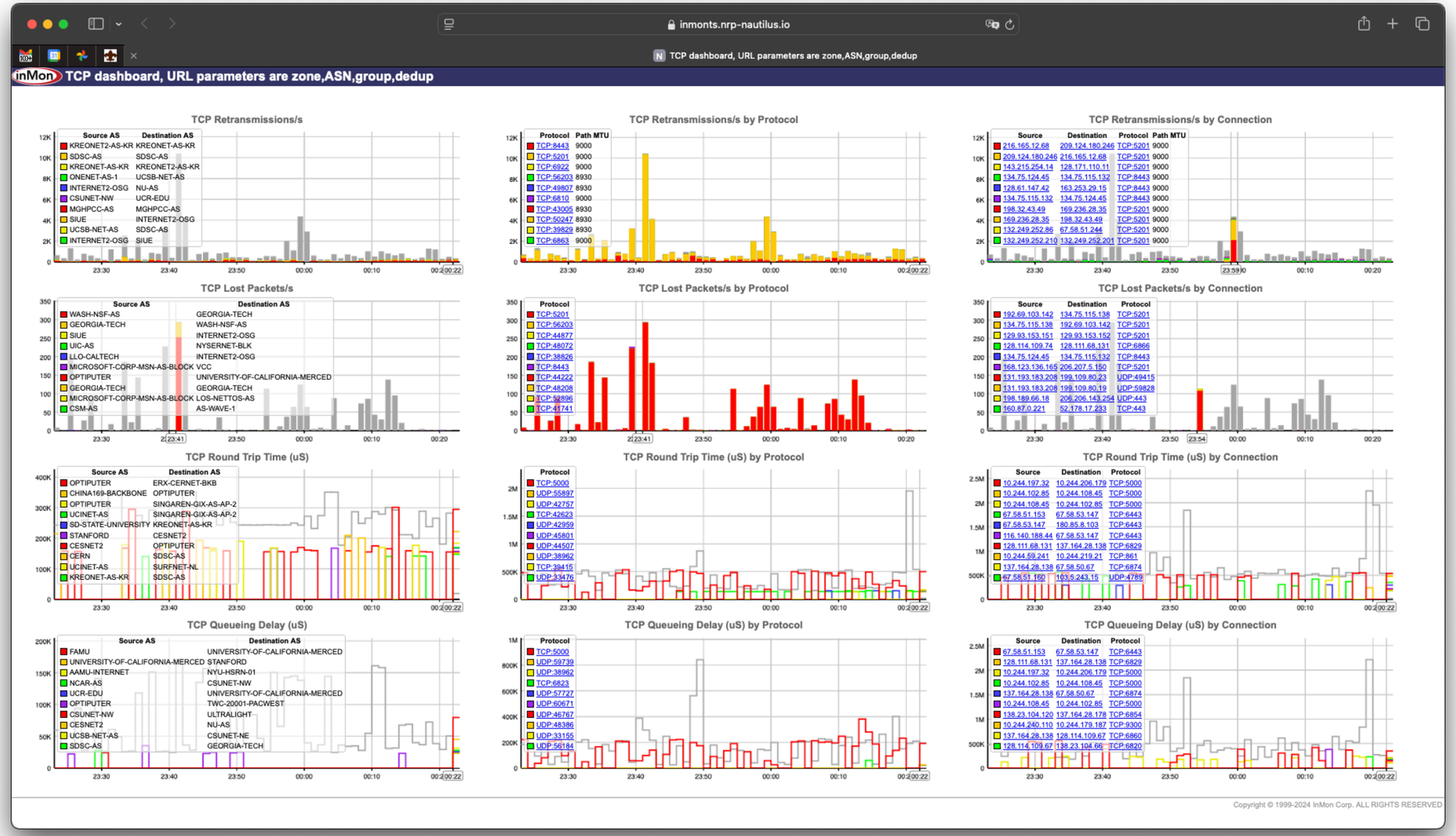
hsflowd = host-sflow daemon = http://sflow.net

TCP Performance Monitoring

```
startSample ----------------------
sampleType_tag 0:1
sampleType FLOWSAMPLE
sampleSequenceNo 153026
sourceId 0:2
meanSkipCount 10
samplePool 1530260
dropEvents 0
inputPort 1073741823
outputPort 2
flowBlock_tag 0:2209
tcpinfo_direction sent
tcpinfo_send_mss 1448
tcpinfo_receive_mss 536
tcpinfo_unacked_pkts 0
tcpinfo_lost_pkts 0
tcpinfo_retrans_pkts 0
tcpinfo_path_mtu 1500
tcpinfo_rtt_uS 773
tcpinfo_rtt_uS_var 137
tcpinfo_send_congestion_win 10
tcpinfo_reordering 3
tcpinfo_rtt_uS_min 0
flowBlock_tag 0:1
flowSampleType HEADER
headerProtocol 1
sampledPacketSize 84
strippedBytes 4
headerLen 66
headerBytes 08-00-27-09-5C-F7-08-00-27-B8-32-6D-08-00-45-C0-00-34-60-79-40-00-01-06-03-7E-0A-00-00-88-0A-00-00-86-84-47-00-B3-50-6C-E7-E7-D8-49-29-17
ED-15-34-00-00-01-01-08-0A-18-09-85-3A-23-8C-C6-61
dstMAC 080027095cf7
srcMAC 080027b8326d
IPSize 66
ip.tot_len 52
srcIP 10.0.0.136
dstIP 10.0.0.134
…
```

TCP info goes out with packet header sample

TCP Performance Monitoring

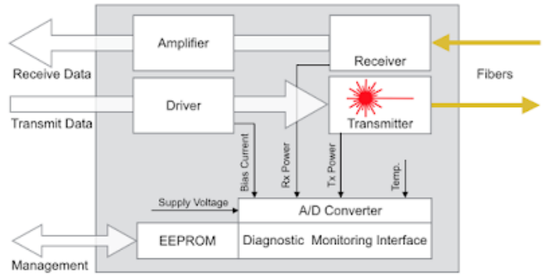


Optional Extensions

Options:

1. Headers of dropped packets
2. TCP delay, loss and jitter
3. Optical interface monitoring
4. Transit delay and queue depth

Optical Interface Monitoring

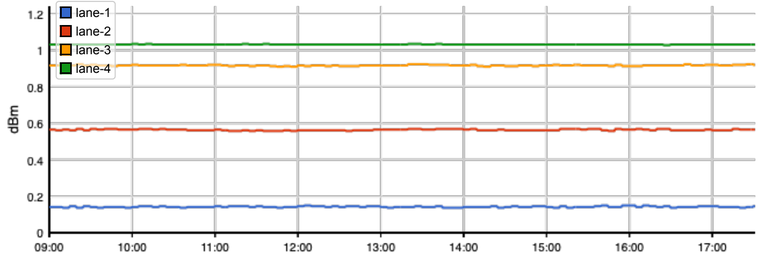



```
struct lane {
  unsigned int index; /* 1-based index of lane within module, 0=unknown */
  unsigned int tx_bias_current; /* microamps */
  unsigned int tx_power;        /* microwatts */
  unsigned int tx_power_min;    /* microwatts */
  unsigned int tx_power_max;    /* microwatts */
  unsigned int tx_wavelength;   /* nanometers */
  unsigned int rx_power;        /* microwatts */
  unsigned int rx_power_min;    /* microwatts */
  unsigned int rx_power_max;    /* microwatts */
  unsigned int rx_wavelength;   /* nanometers */
}

/* Optical SFP / QSFP metrics */
/* opaque = counter_data; enterprise=0; format=10 */
struct sfp {
  unsigned int module_id;
  unsigned int module_num_lanes;      /* total number of lanes in module */
  unsigned int module_supply_voltage; /* millivolts */
  int module_temperature;             /* thousandths of a degree Celsius */
  lane<> lanes;
}
```

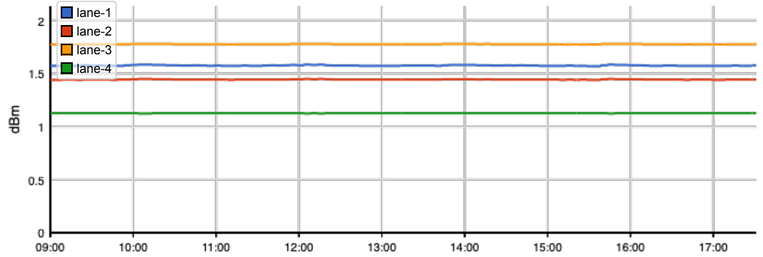
Optical Interface Monitoring

Optical Agent/Port-dashboard v3, URL parameters are agent,port,interval,N

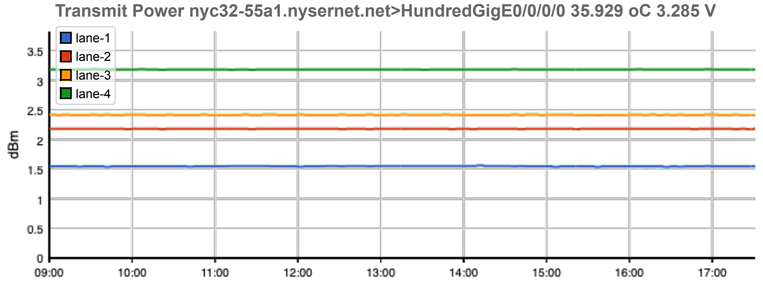
Receive Power nyc32-55a1.nysernet.net>HundredGigE0/0/0/0 35.929 oC 3.285 V



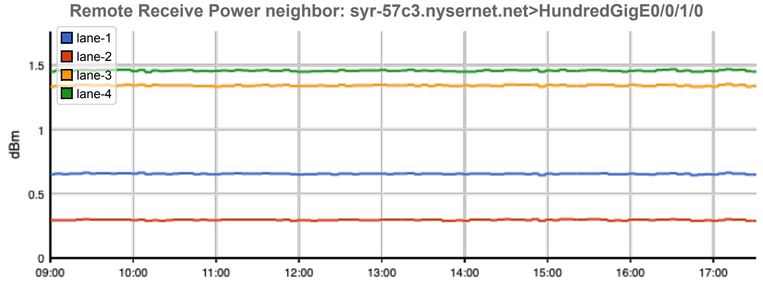
Remote Transmit Power neighbor: syr-57c3.nysernet.net>HundredGigE0/0/1/0



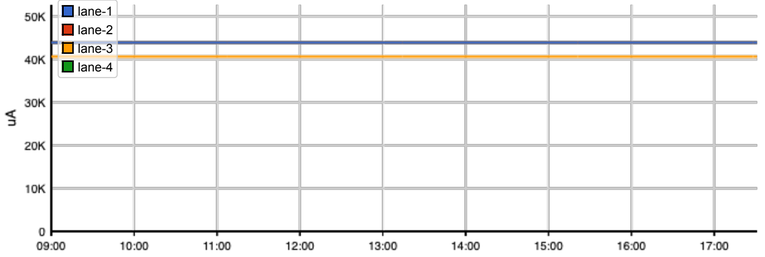
Transmit Power nyc32-55a1.nysernet.net>HundredGigE0/0/0/0 35.929 oC 3.285 V



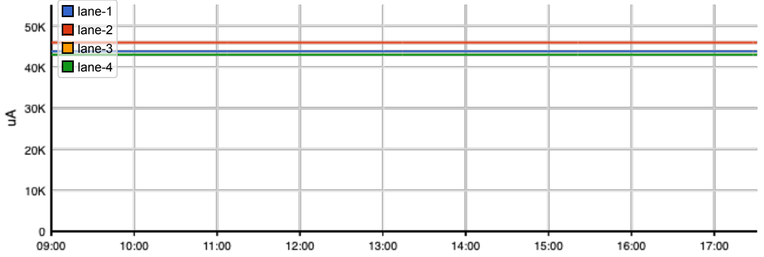
Remote Receive Power neighbor: syr-57c3.nysernet.net>HundredGigE0/0/1/0

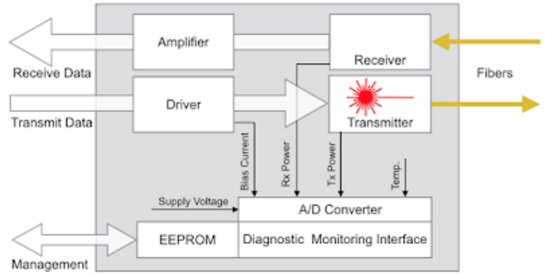


Transmit Bias Current nyc32-55a1.nysernet.net>HundredGigE0/0/0/0 35.929 oC 3.285 V



Remote TX Bias Current neighbor: syr-57c3.nysernet.net>HundredGigE0/0/1/0



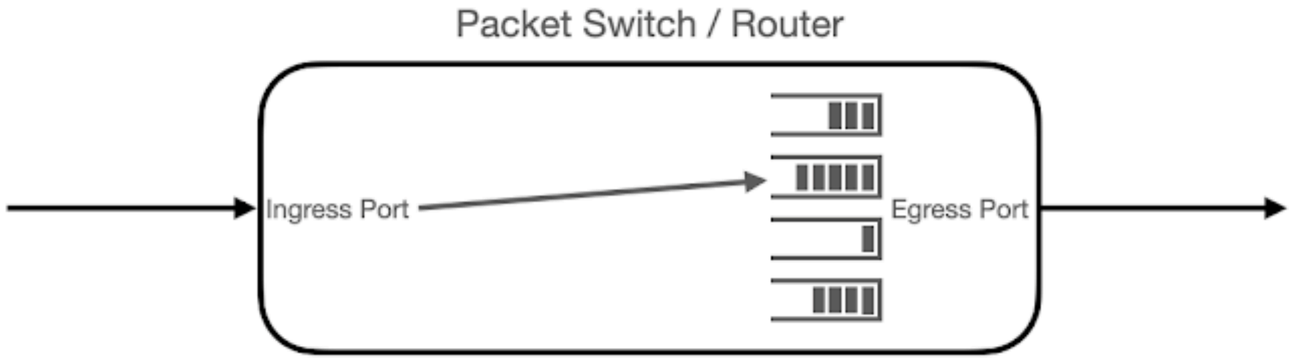


Optional Extensions

Options:

1. Headers of dropped packets
2. TCP delay, loss and jitter
3. Optical interface monitoring
4. Transit delay and queue depth

Transit Delay and Queueing



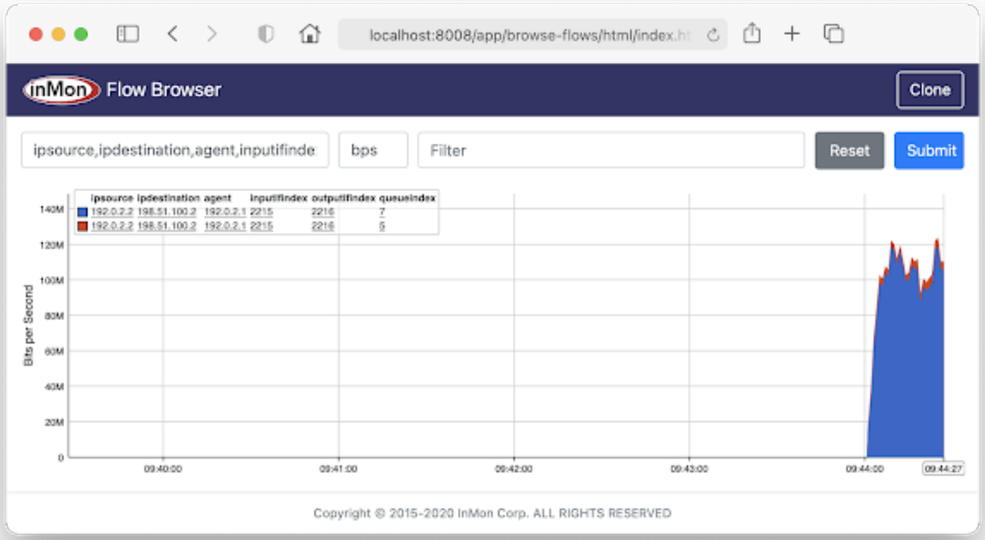
Transit Delay and Queueing

```
startSample ----------------------
sampleType_tag 0:1
sampleType FLOWSAMPLE
sampleSequenceNo 91159
sourceId 0:2216
meanSkipCount 400
samplePool 36463600
dropEvents 0
inputPort 2215
outputPort 2216
flowBlock_tag 0:1036
extendedType egress_queue
egress_queue_id 7
flowBlock_tag 0:1040
extendedType queue_depth
queue_depth_bytes 11354112
flowBlock_tag 0:1039
extendedType transit_delay
transit_delay_nS 839660224
flowBlock_tag 0:1
flowSampleType HEADER
headerProtocol 1
sampledPacketSize 1446
strippedBytes 4
headerLen 128
headerBytes 98-03-9B-8F-B5-CC-98-03-9B-94-C7-D5-08-00-45-16-05-94-12-C7-00-00-FE-11-B8-43-C0-00-02-02-C6-33-64-02-30-39-D4-31-05-80-
D7-1D-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-4
2-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42-42
dstMAC 98039b8fb5cc
srcMAC 98039b94c7d5
IPSize 1428
ip.tot_len 1428
srcIP 192.0.2.2
dstIP 198.51.100.2
IPProtocol 17
IPTOS 22
IPTTL 254
IPID 50962
UDPSrcPort 12345
UDPDstPort 54321
UDPBytes 1408
endSample   ----------------------
```

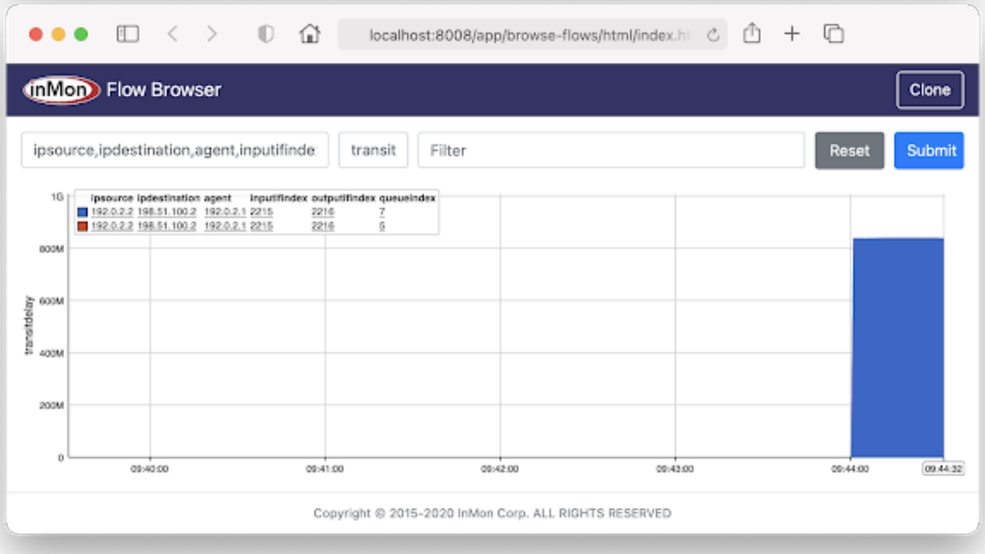
Queueing info goes out with packet header sample

Transit Delay and Queueing

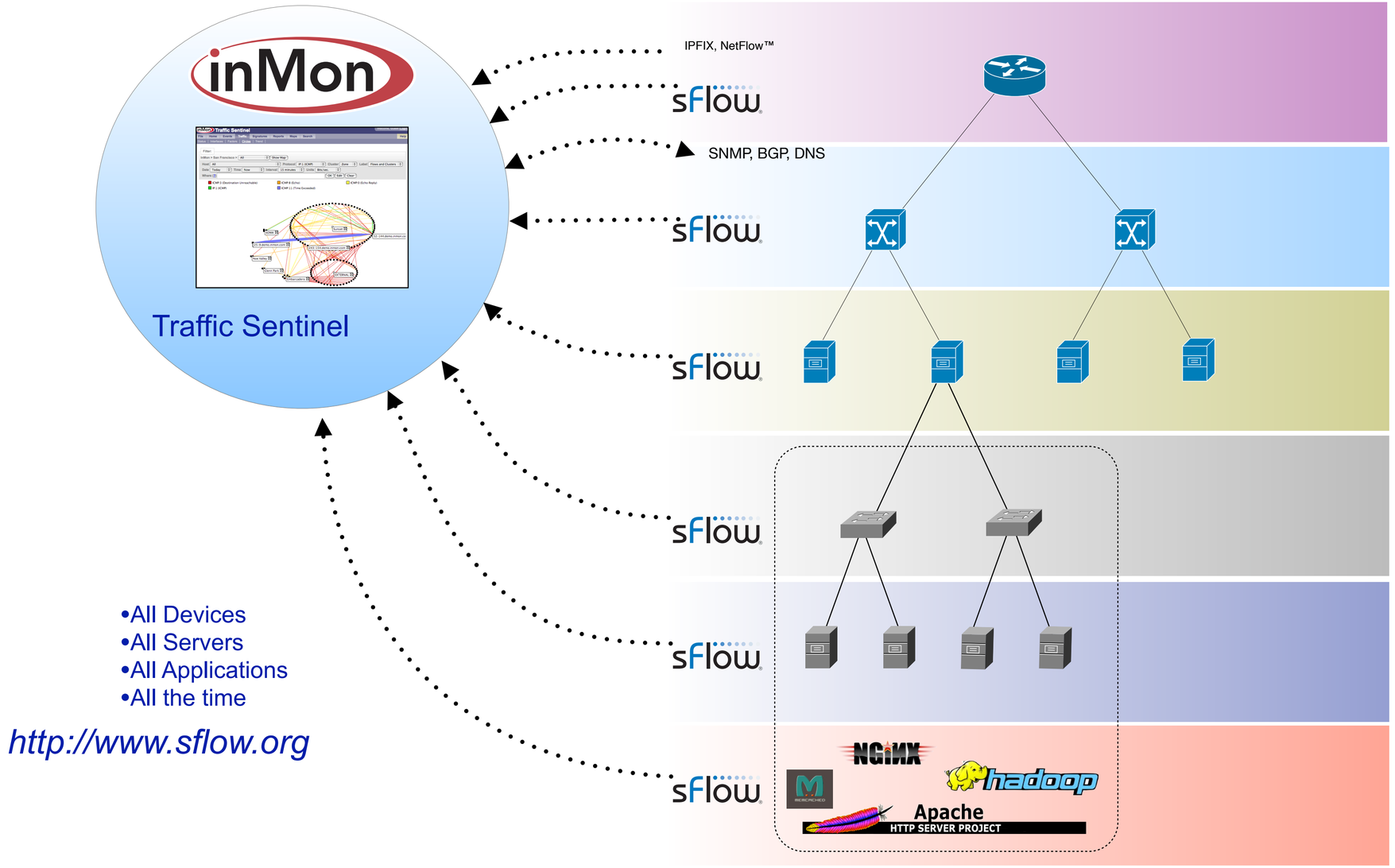
Bits/sec



Transit Delay



Integrated measurement

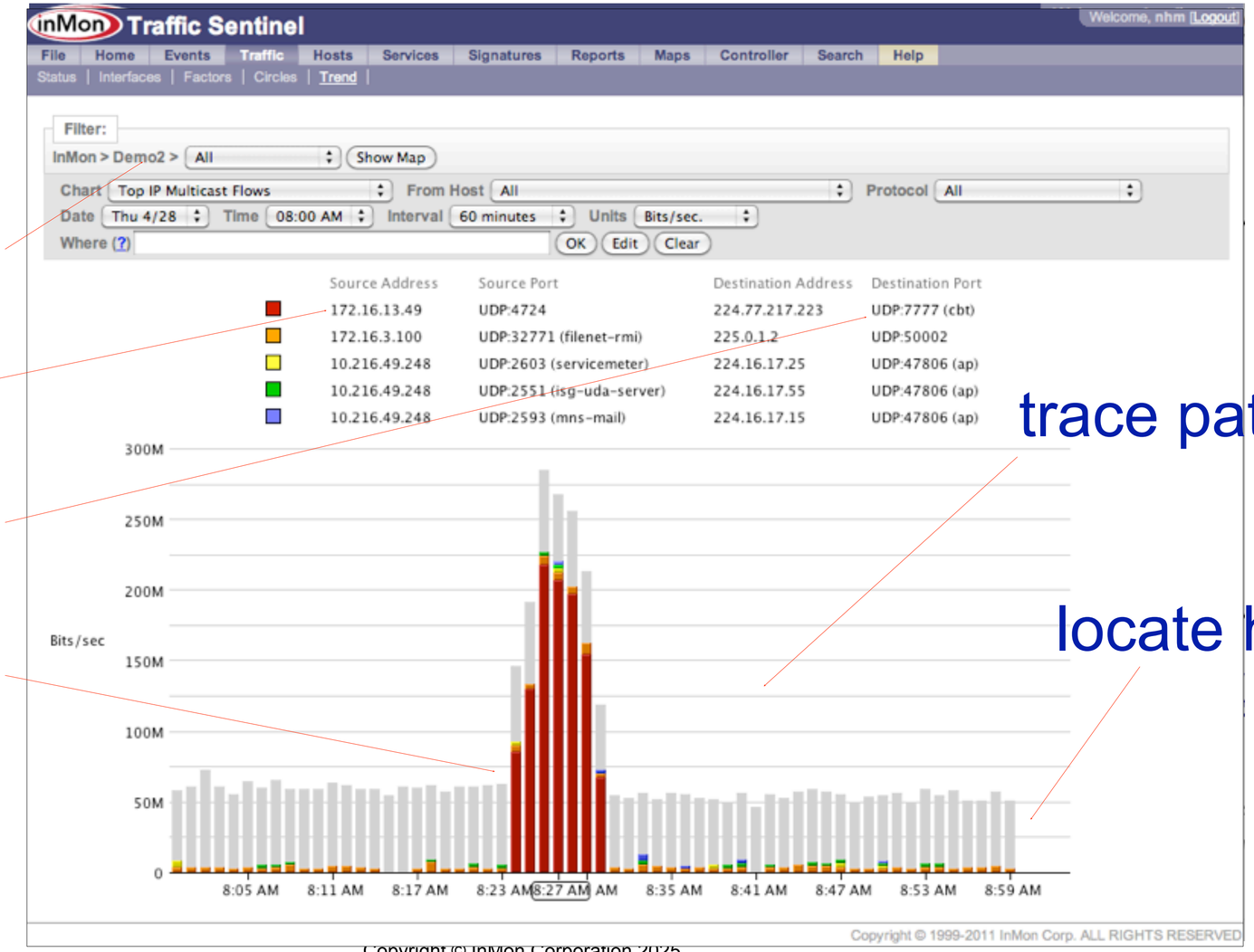


- All Devices
- All Servers
- All Applications
- All the time

http://www.sflow.org

Integrated measurement

Always have context…



where

who

what

when

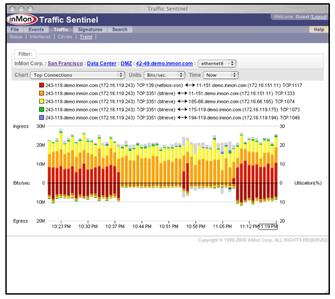
trace path

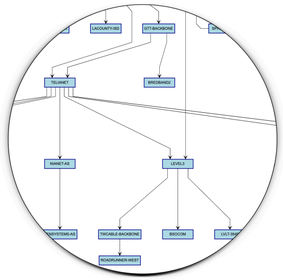
locate hosts

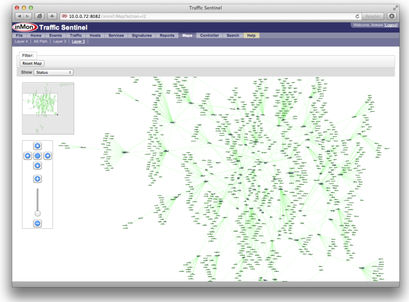
Integrated measurement

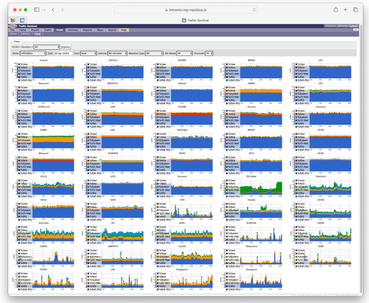
InMon
Traffic Sentinel
data collection and analysis

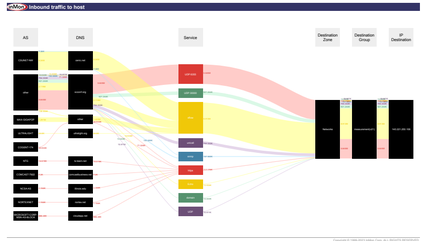
- Scalable: collects data* from 200K+ ports, 4000+ devices
- Collects sFlow, NetFlow, SNMP, BGP, DNS
- Network-wide visualization and trending of traffic patterns
- Threshold alerts
- Multivendor topology discovery
- Location of end-hosts
- Scheduled reporting
- Custom dashboards and maps
- Automatic de-duplication
- Extensible REST API builder

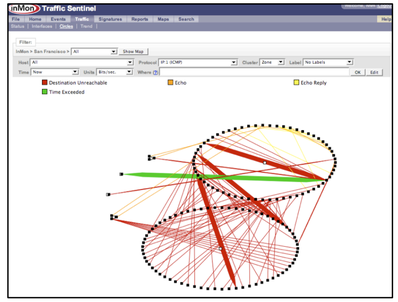




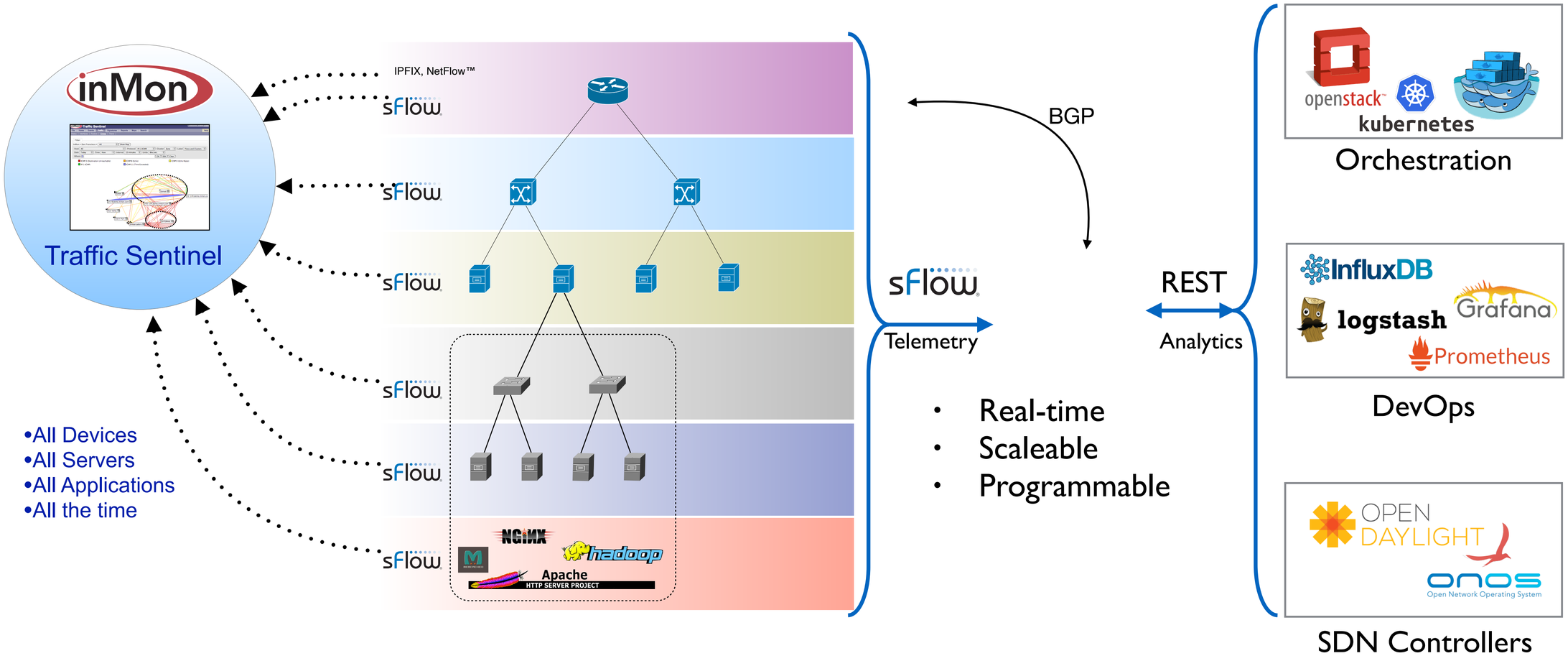




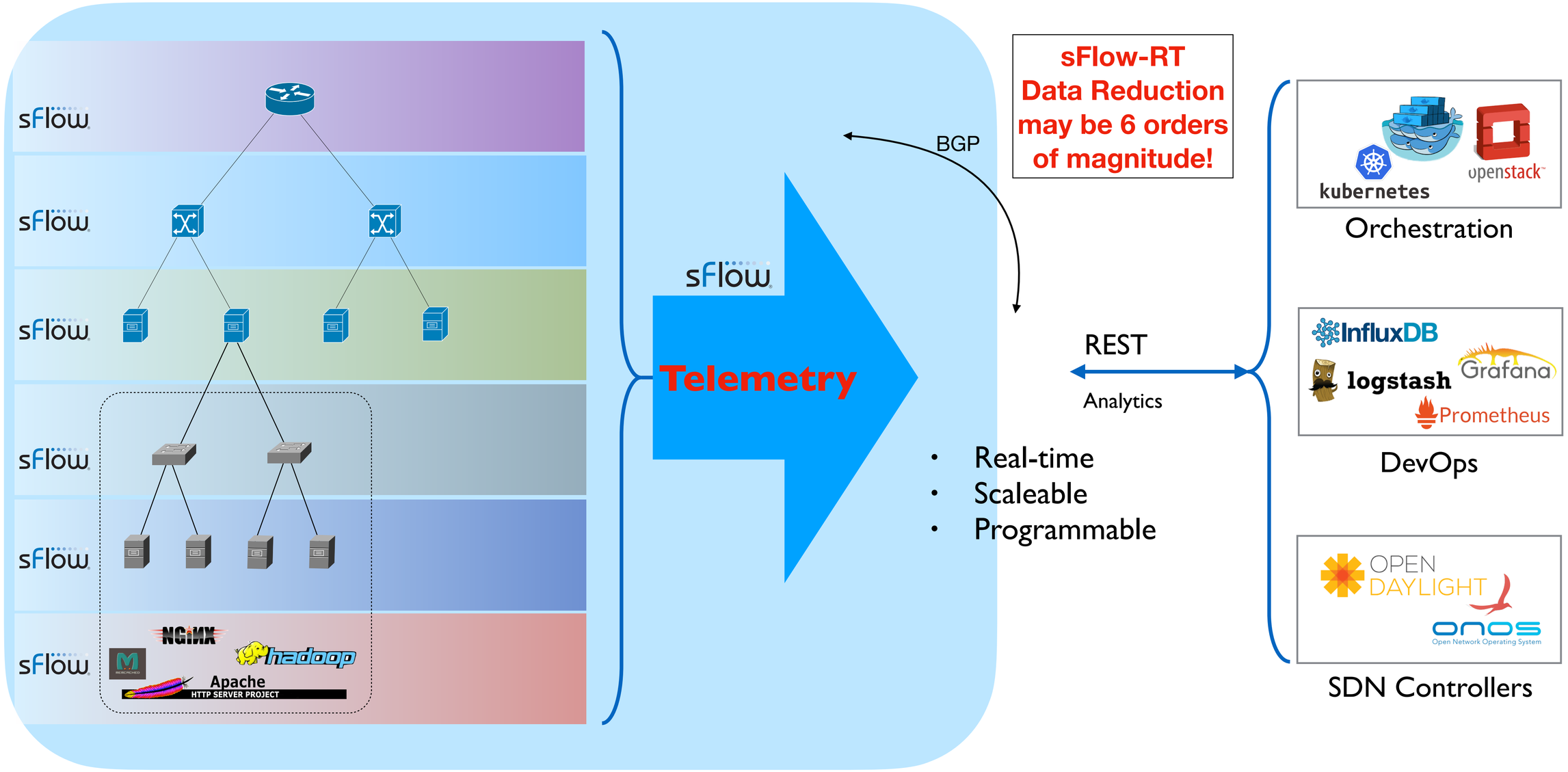




Integrated measurement

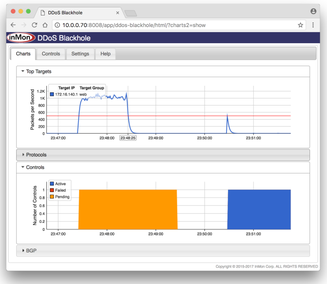


Integrated measurement

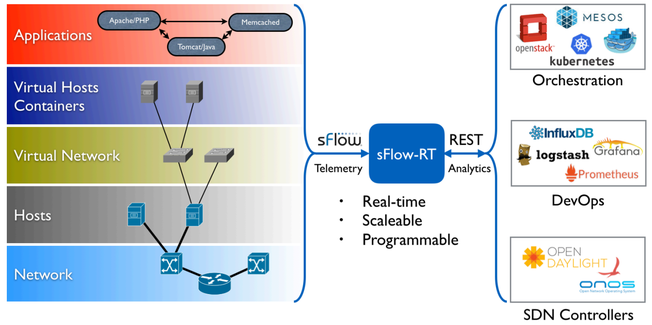


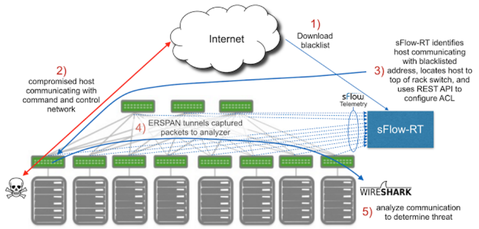
Real-time control with sFlow-RT

sFlow-RT = scriptable real-time analytics engine
(sFlow, BGP, DNS, OpenFlow, REST, FlowSpec)

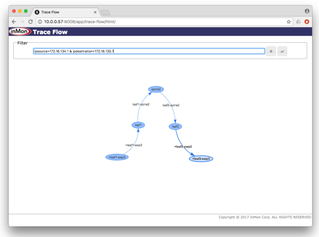


Sub-second DDoS Mitigation

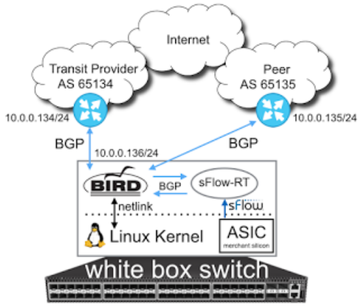




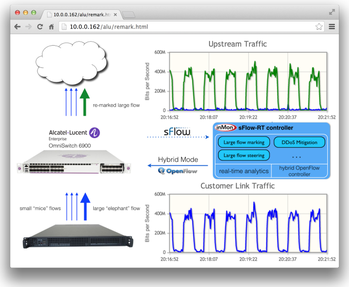
Trigger ERSPAN capture



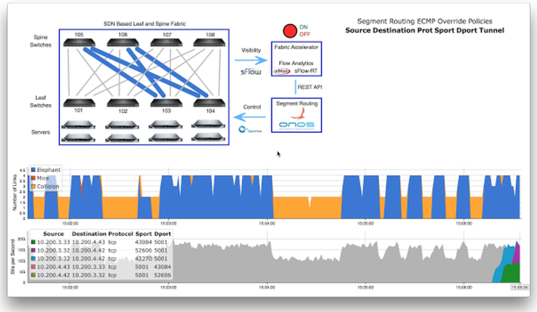
ECMP flow tracing



SIR router



Large flow marking



Large flow steering

sFlow Tutorial 6NRP Jan 28

Agenda:

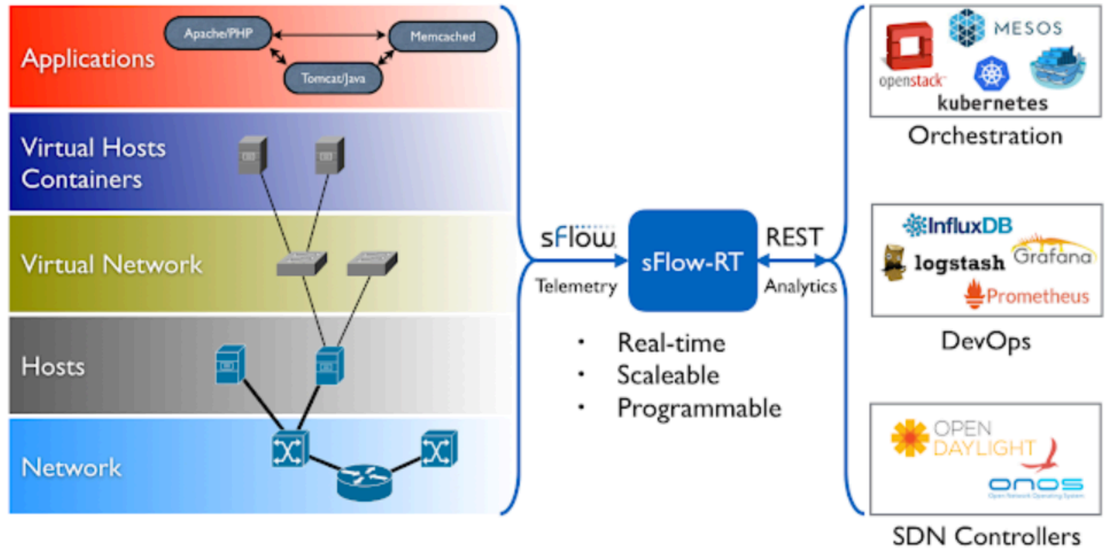
1. Introduction to sFlow
2. Hands-on deployment of sFlow tool-chain
3. Passive TCP delay, loss and jitter measurements
4. Network-wide packet drop analysis

sFlow-RT - Prometheus

https://blog.sflow.com/2023/07/deploy-real-time-network-dashboards.html
visit blog.sflow.com and search for "compose"

Thursday, July 13, 2023

Deploy real-time network dashboards using Docker compose



This article demonstrates how to use docker compose to quickly deploy a real-time network analytics stack that includes the sFlow-RT analytics engine, Prometheus time series database, and Grafana to create dashboards.

sFlow-RT - Prometheus

https://blog.sflow.com/2023/07/deploy-real-time-network-dashboards.html
(visit blog.sflow.com and search for "compose")

```
git clone https://github.com/sflow-rt/prometheus-grafana.git
cd prometheus-grafana
./start.sh
```

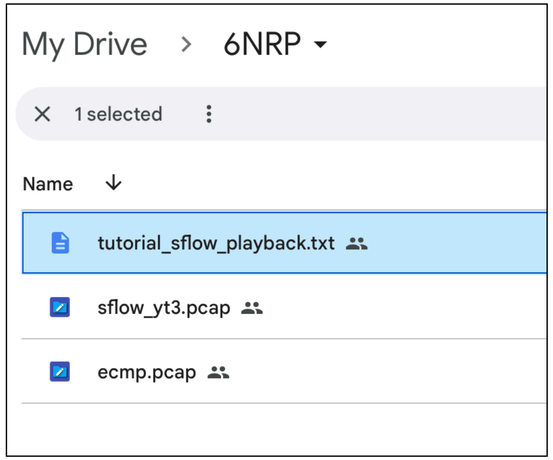
Congratulations, your laptop is now an sFlow collector tool-chain, listening for data on 6343/udp

sFlow-RT - Prometheus

Use sflowtool to play back recorded sFlow capture...

1. Connect to your sflow-rt on http//127.0.0.1:8008

2. Open a second terminal window, make a new directory called "sflowtool" and download files from this drive folder:
https://tinyurl.com/sflow6nrp

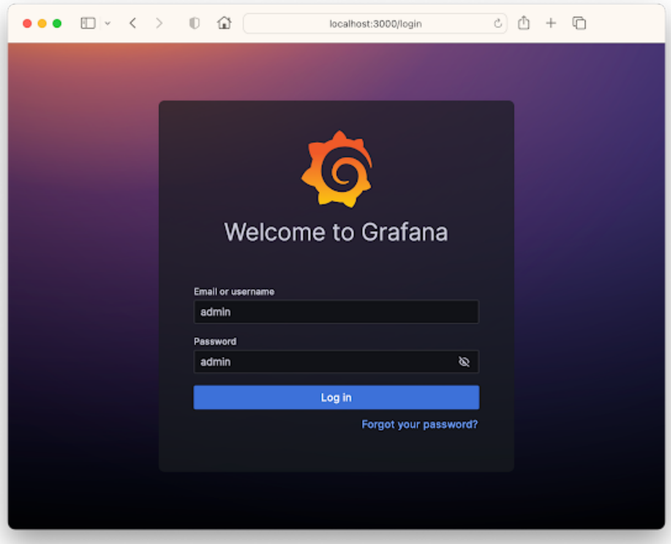


3. Run commands:
- docker pull sflow/sflowtool
- GW=192.168.x.x <--- your own laptop IP address right now
- CAP=sflow_yt3.pcap
- docker run -v $PWD/$CAP:/$CAP sflow/sflowtool -r /$CAP -f $GW/6343 -P 10

4. See traffic in sFlow-RT flow-browser

sFlow-RT - Prometheus

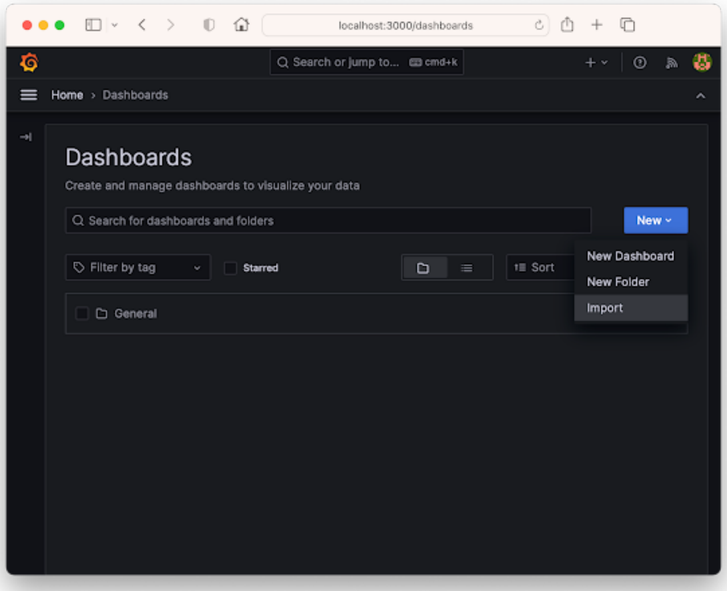
https://blog.sflow.com/2023/07/deploy-real-time-network-dashboards.html



Connect to the Grafana web interface on port 3000 using default user name and password (admin/admin). You will be promted to change the password.

sFlow-RT - Prometheus

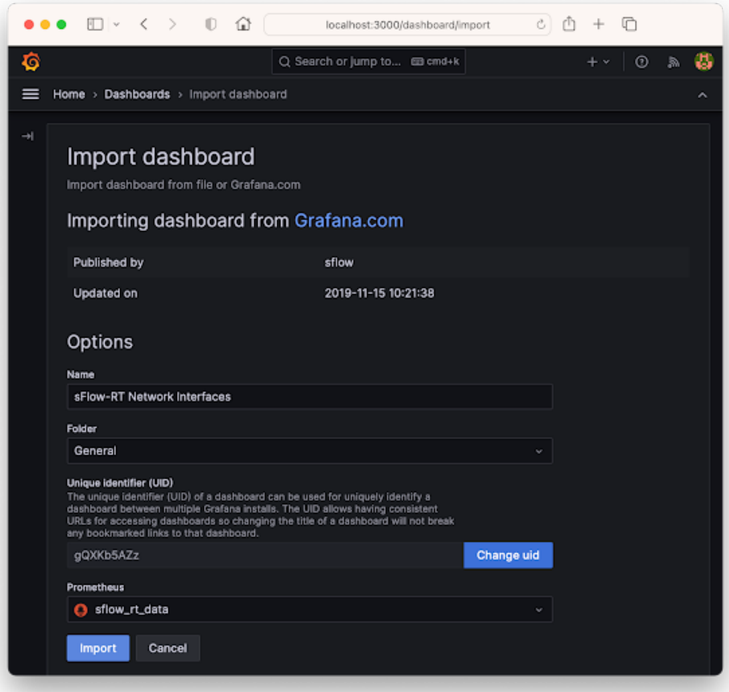
https://blog.sflow.com/2023/07/deploy-real-time-network-dashboards.html

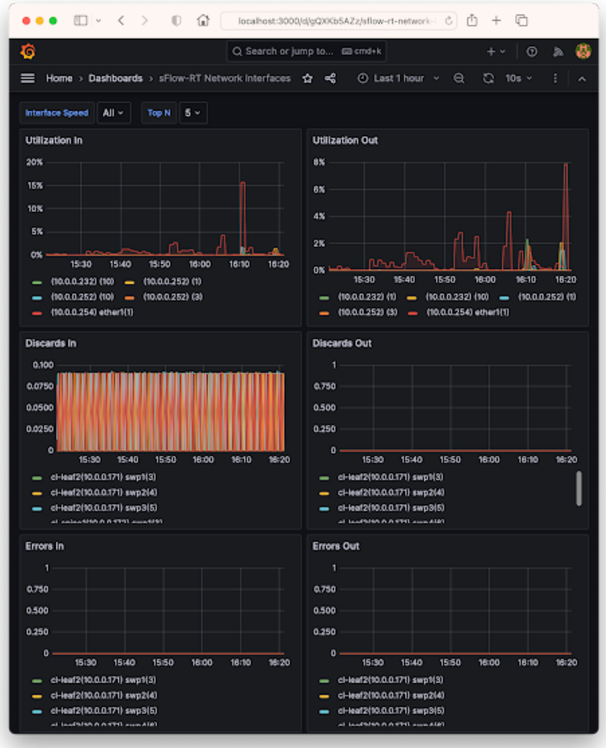


Select the option to Import a new Dashboard. Enter the code *11201* to import sFlow-RT Network Interfaces dashboard from Grafana.com and click on the *Load* button.

sFlow-RT - Prometheus

https://blog.sflow.com/2023/07/deploy-real-time-network-dashboards.html

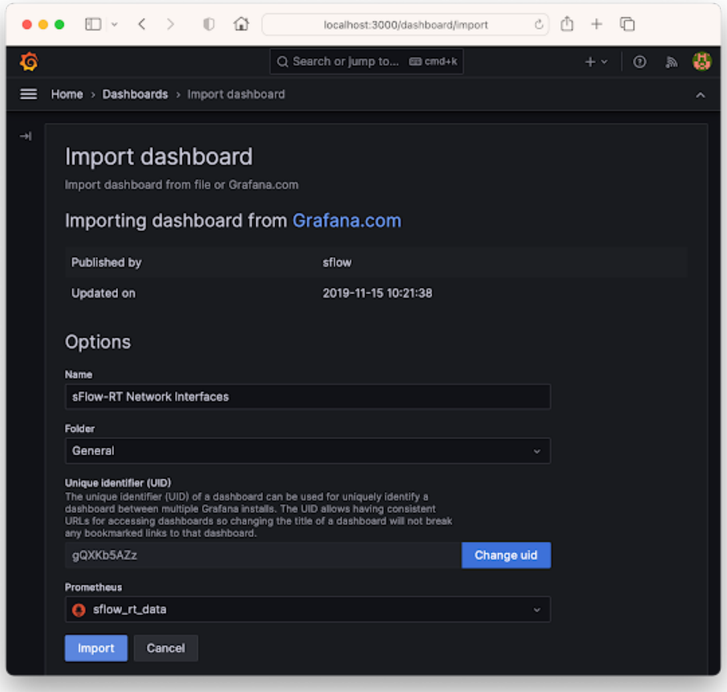


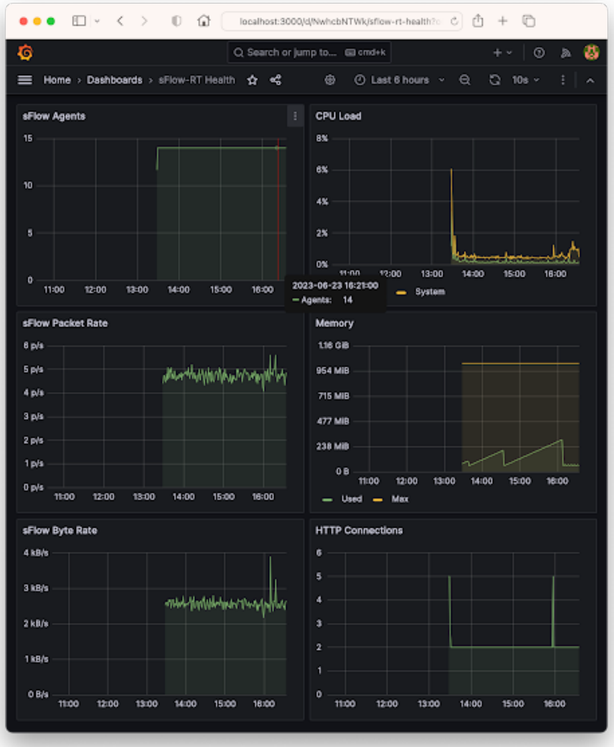


Select the *sflow_rt_data* Prometheus database and click on the *Import* button.

sFlow-RT - Prometheus

https://blog.sflow.com/2023/07/deploy-real-time-network-dashboards.html





Repeat the steps to add the sFlow-RT Health dashboard, code *11096*

sFlow-RT - Prometheus

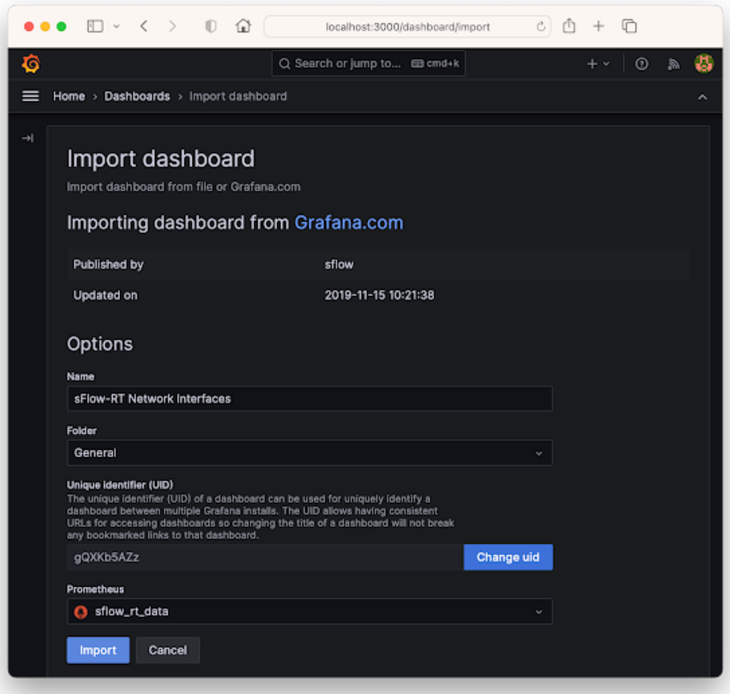
https://blog.sflow.com/2023/07/deploy-real-time-network-dashboards.html

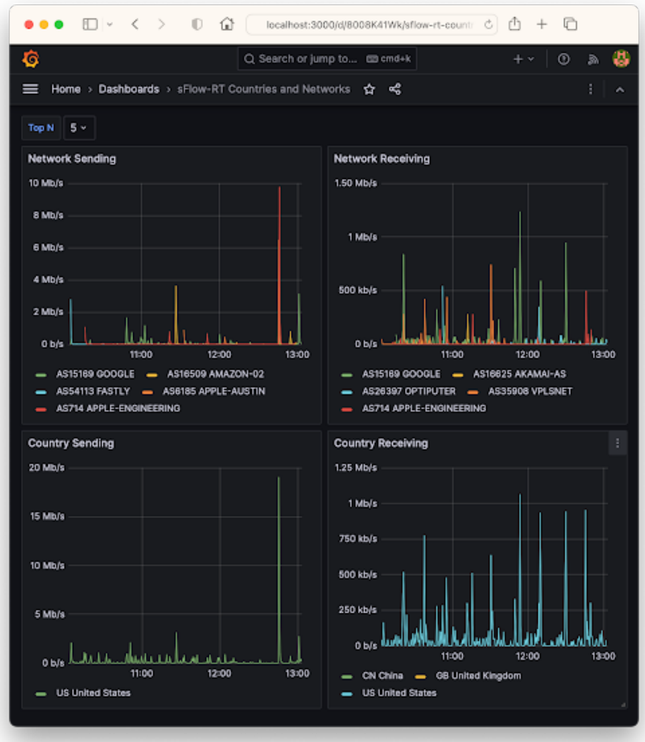
```
- job_name: 'sflow-rt-countries'
    metrics_path: /app/prometheus/scripts/export.js/flows/ALL/txt
    static_configs:
      - targets: ['sflow-rt:8008']
    params:
      metric: ['sflow_country_bps']
      key:
        - 'null:[country:ipsource:both]:unknown'
        - 'null:[country:ipdestination:both]:unknown'
      label: ['src','dst']
      value: ['bytes']
      scale: ['8']
      aggMode: ['sum']
      minValue: ['1000']
      maxFlows: ['100']

  - job_name: 'sflow-rt-asns'
    metrics_path: /app/prometheus/scripts/export.js/flows/ALL/txt
    static_configs:
      - targets: ['sflow-rt:8008']
    params:
      metric: ['sflow_asn_bps']
      key:
        - 'null:[asn:ipsource:both]:unknown'
        - 'null:[asn:ipdestination:both]:unknown'
      label: ['src','dst']
      value: ['bytes']
      scale: ['8']
      aggMode: ['sum']
      minValue: ['1000']
      maxFlows: ['100']
```

Extend prometheus/prometheus.yml to append the above scraper config, then run "docker restart prometheus"

sFlow-RT - Prometheus

https://blog.sflow.com/2023/07/deploy-real-time-network-dashboards.html





Add dashboard 11146 to load the sFlow-RT Countries and Networks dashboard.

sFlow Tutorial 6NRP Jan 28

Agenda:

1. Introduction to sFlow
2. Hands-on deployment of sFlow tool-chain
3. Passive TCP delay, loss and jitter measurements
4. Network-wide packet drop analysis

Passive TCP Monitoring

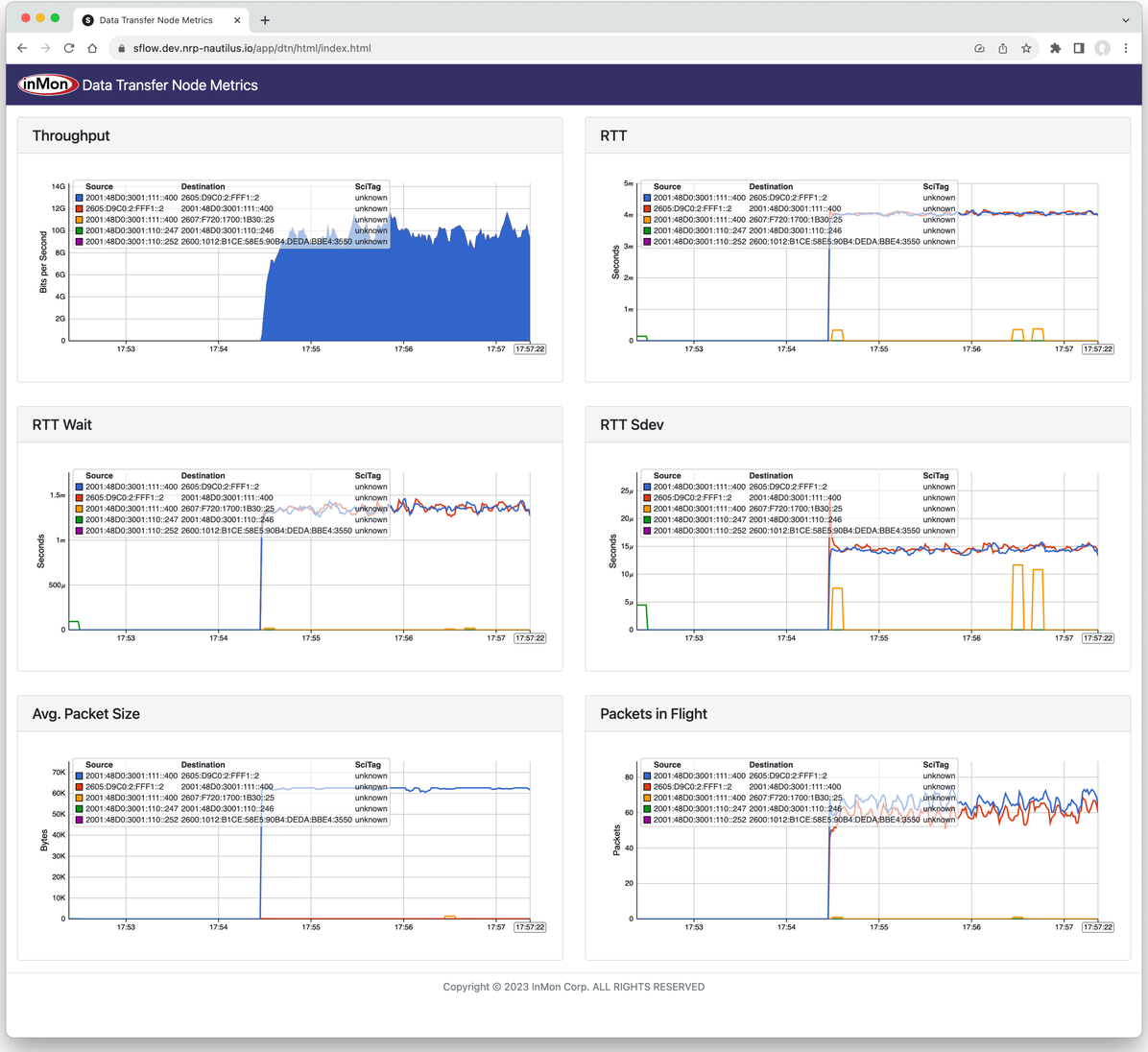
/etc/hsflowd.conf:

```
sflow {
  collector { ip=127.0.0.1 }
  pcap { speed=1G- }
  tcp {}
}
```

https://sflow.net/host-sflow-linux-config.php

Passive TCP Monitoring

Live demo



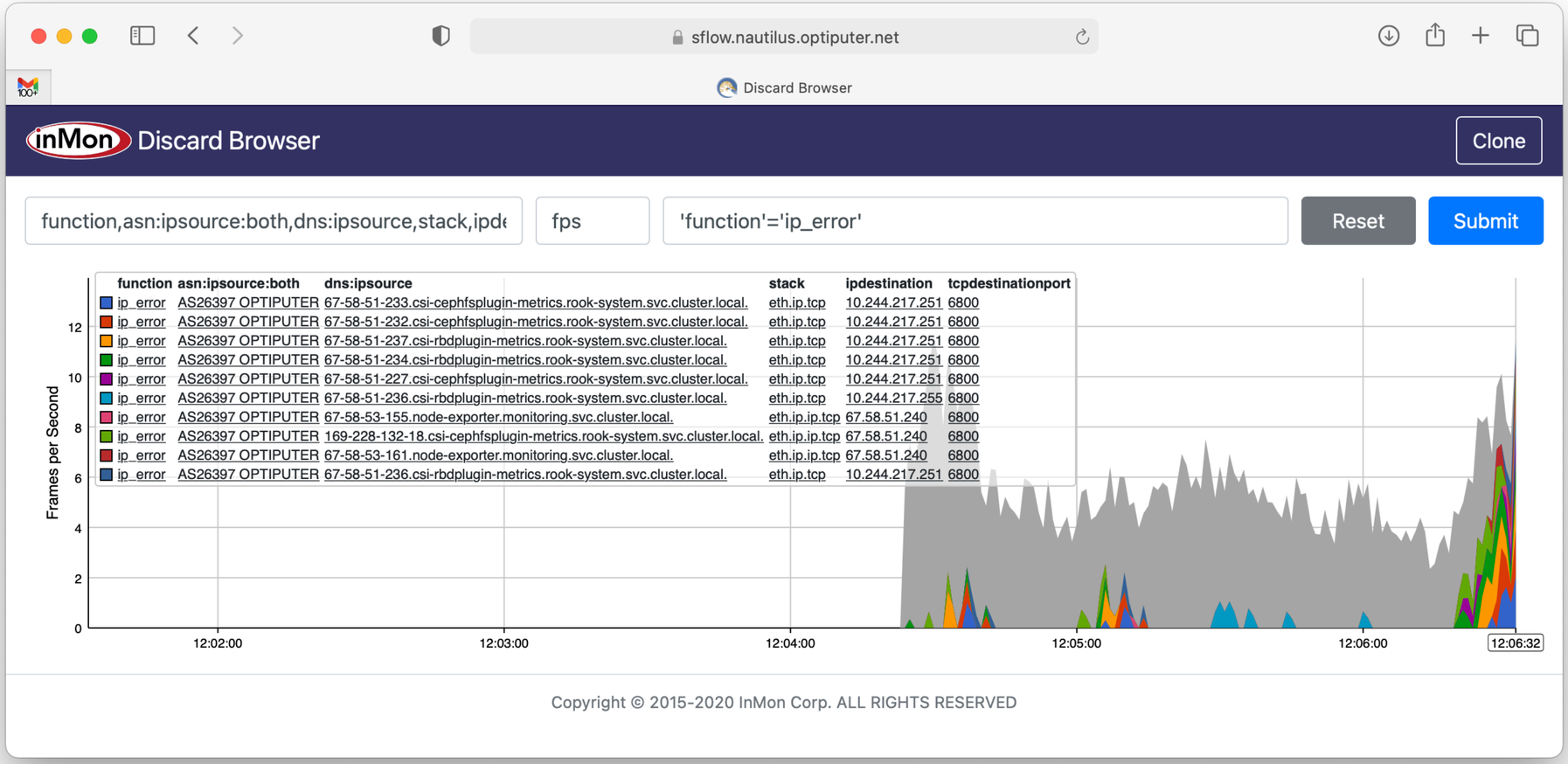
sFlow Tutorial 6NRP Jan 28

Agenda:

1. Introduction to sFlow
2. Hands-on deployment of sFlow tool-chain
3. Passive TCP delay, loss and jitter measurements
4. Network-wide packet drop analysis

Packet Drop Analysis

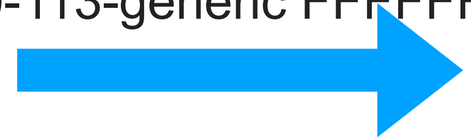
Live demo



Server Drop Monitoring example

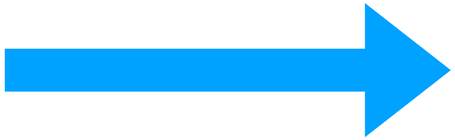
Linux drop reports function=="__udp4_lib_rcv+0x597"

```
# 1. Look up symbol:
root> grep __udp4_lib_rcv /boot/System.map-5.15.0-113-generic
ffffffff81c004a0 T __udp4_lib_rcv

# 2. Add offset and find Linux kernel source code line:
root> eu-addr2line -e /usr/lib/debug/boot/vmlinux-5.15.0-113-generic FFFFFFFF81C00A37
/build/linux-3d8Wab/linux-5.15.0/net/ipv4/udp.c:2465:9
```



sFlow from servers reports packets dropped in host protocol stack (identifying kernel function)

Server Drop Monitoring example



linux / net / ipv4 / udp.c

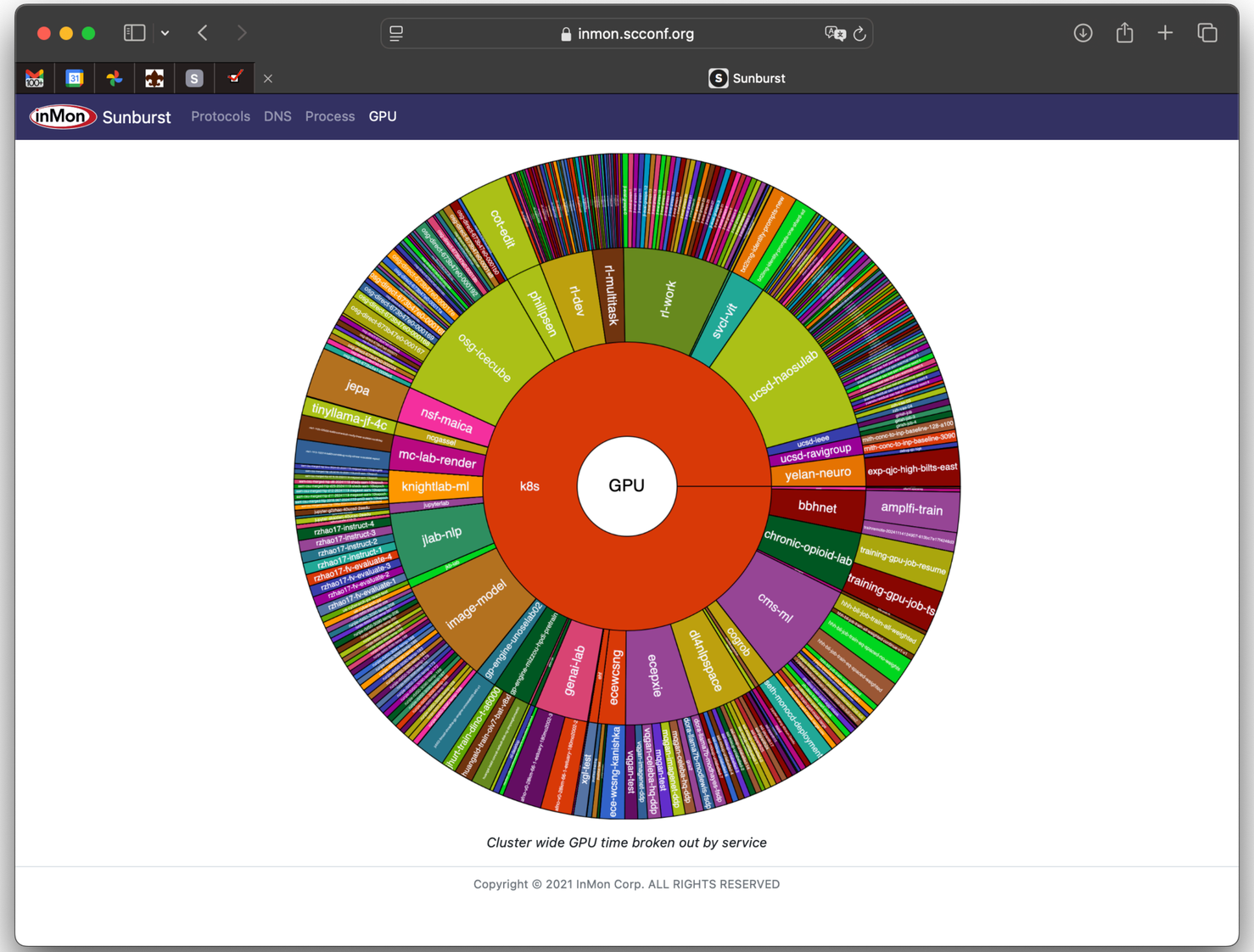
Code | Blame | 3343 lines (2876 loc) · 83.6 KB | Code 55% faster with GitHub Copilot | Raw

```
2455            if (udp_lib_checksum_complete(skb))
2456                    goto csum_error;
2457
2458            __UDP_INC_STATS(net, UDP_MIB_NOPORTS, proto == IPPROTO_UDPLITE);
2459            icmp_send(skb, ICMP_DEST_UNREACH, ICMP_PORT_UNREACH, 0);
2460
2461            /*
2462             * Hmm.  We got an UDP packet to a port to which we
2463             * don't wanna listen.  Ignore it.
2464             */
2465            kfree_skb(skb);
2466            return 0;
2467
2468    short_packet:
2469            net_dbg_ratelimited("UDP%s: short packet: From %pI4:%u %d/%d to %pI4:%u\n",
2470                                proto == IPPROTO_UDPLITE ? "Lite" : "",
2471                                &saddr, ntohs(uh->source),
2472                                ulen, skb->len,
2473                                &daddr, ntohs(uh->dest));
2474            goto drop;
2475
2476    csum_error:
```

sFlow from servers reports packets dropped in host protocol stack (identifying kernel function)

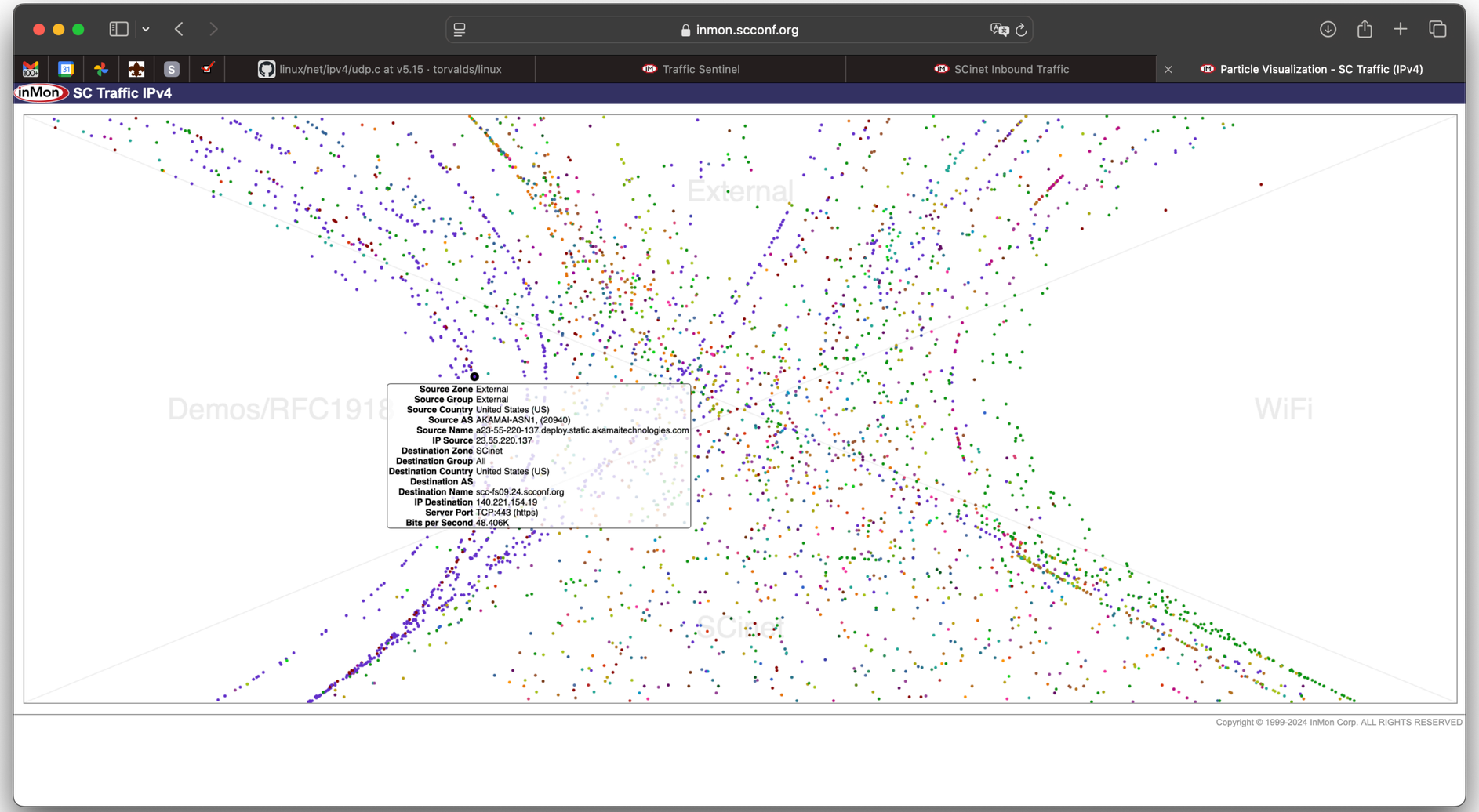
Nautilus Cluster Monitoring

Live demo

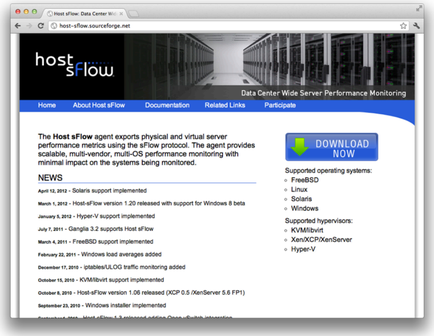


Cluster wide GPU time broken out by service

sFlow Monitoring examples

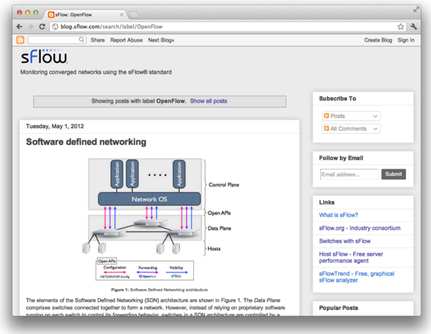


More Information



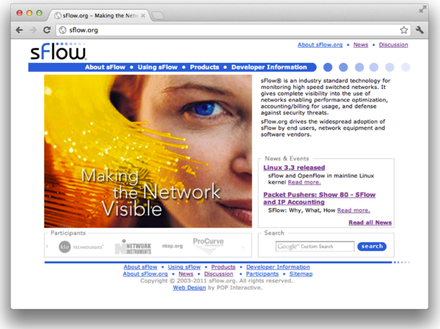
sflow.net

freeware agents



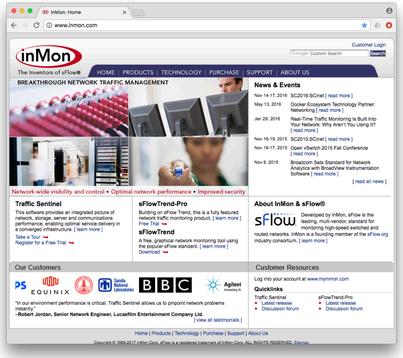
blog.sflow.com

articles



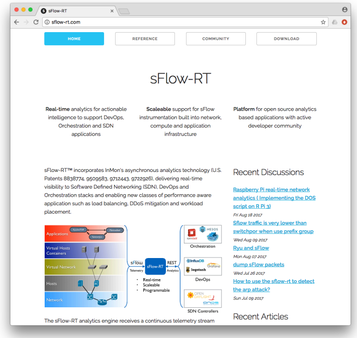
sflow.org

sFlow standard



inmon.com

commercial products



sflow-rt.com

real-time analytics
closed-loop control