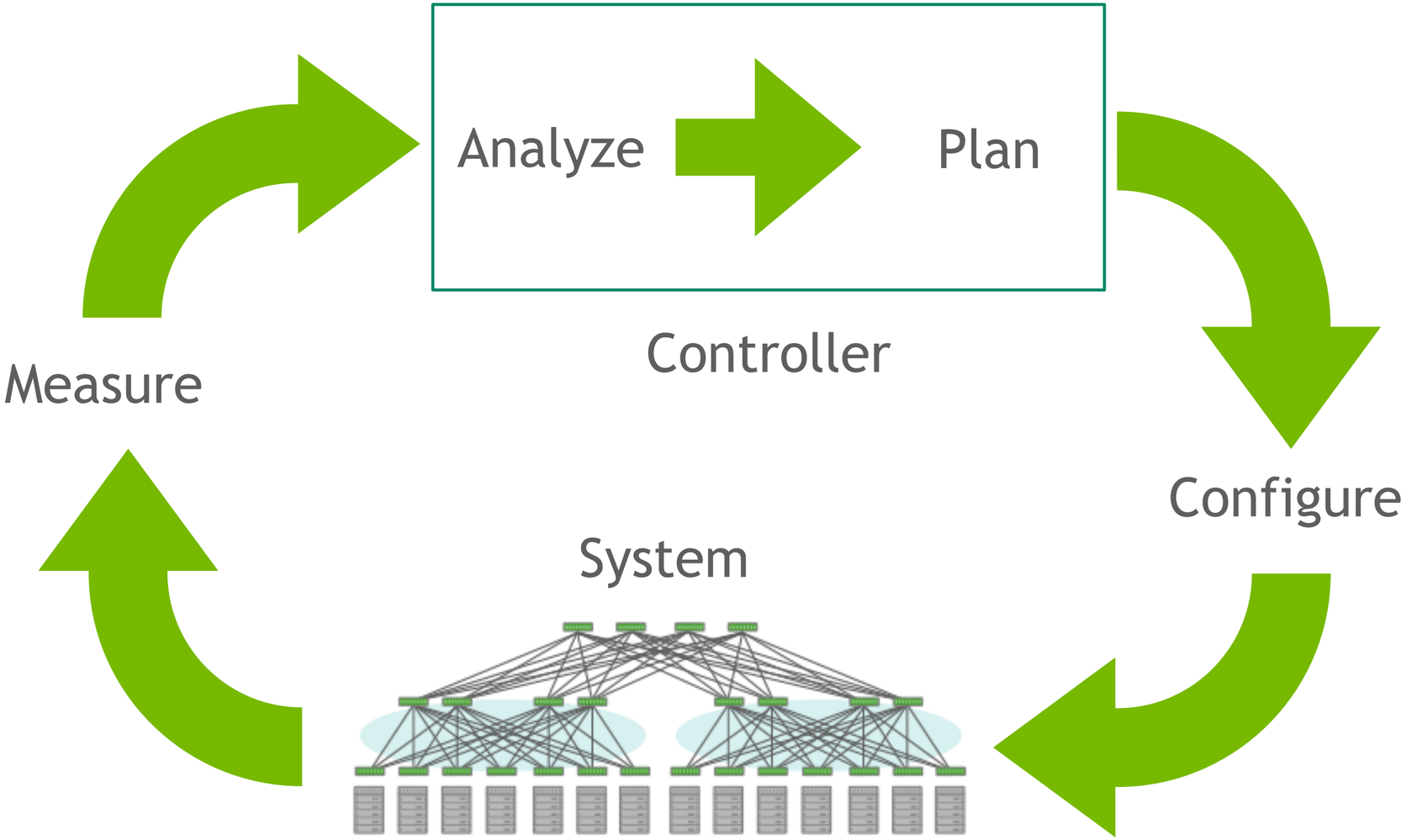
“ If you can’t measure it, you can’t improve it”

— Lord Kelvin

WHY MEASURE?

FEEDBACK CONTROL



STABILITY REQUIREMENTS

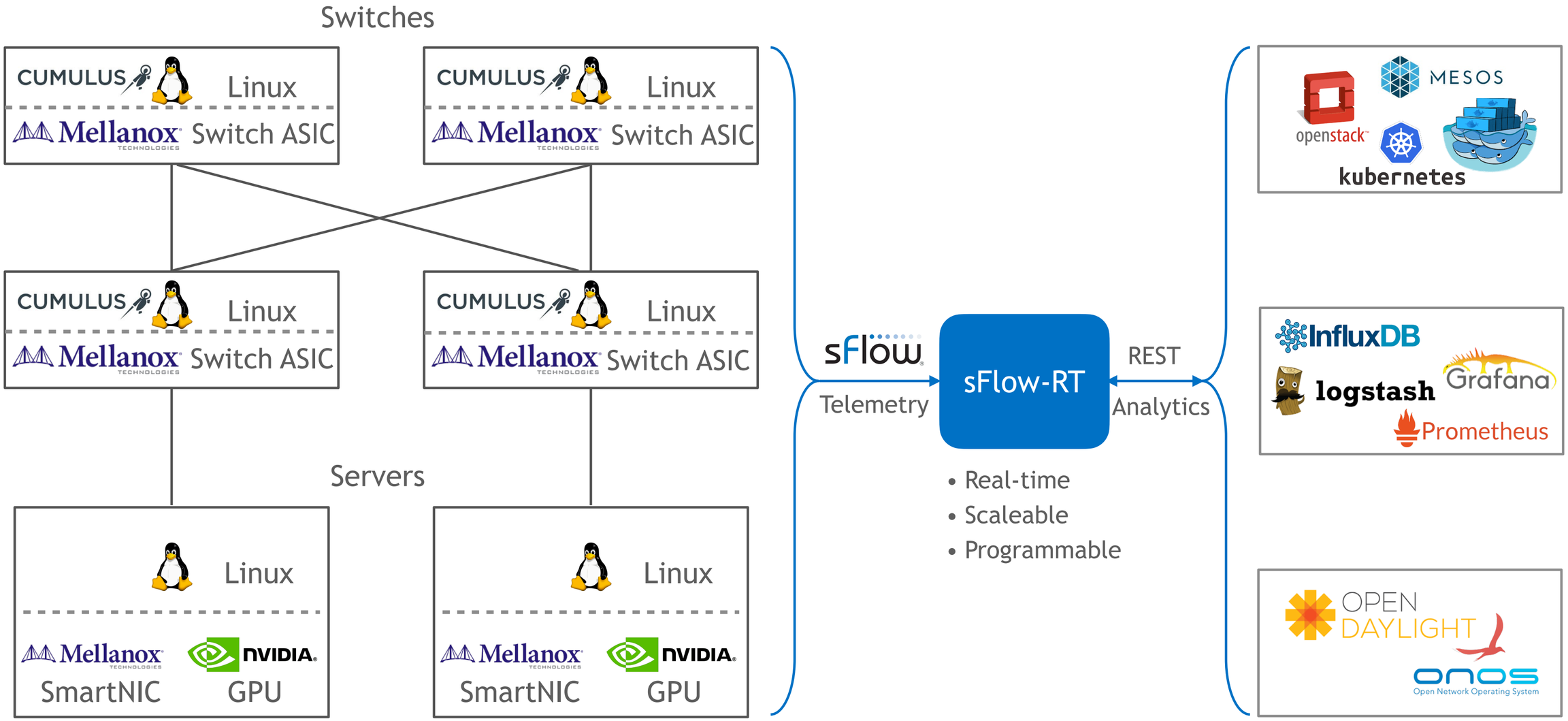
- **Observable**
  - Ability to monitor all important system states
  - Example: To safely drive a car you need to see the road ahead, mirrors, and speedometer
- **Controllable**
  - Ability to influence all important system states
  - Example: To control a car's speed and direction you need to have a steering wheel, accelerator, and brakes
- **Responsive**
  - Feedback loop must be fast enough to track changes
  - Example: To avoid a car accident you need to react quickly if the car ahead brakes suddenly.

“In God we trust. All others bring data.”

— Dr. Edwards Deming

WHAT IS SFLOW?

NETWORK AND SYSTEM OBSERVABILITY



Embedded monitoring of all switches, all servers, all applications, all the time

Consistent measurements shared between multiple management tools

STANDARD COUNTERS

- Network (maintained by hardware in network devices)
  - MIB-2 ifTable: ifInOctets, ifInUcastPkts, ifInMulticastPkts, ifInBroadcastPkts, ifInDiscards, ifInErrors, ifUnkownProtos, ifOutOctets, ifOutUcastPkts, ifOutMulticastPkts, ifOutBroadcastPkts, ifOutDiscards, ifOutErrors
- Host (maintained by operating system kernel)
  - GPU: device_count, processes, gpu_time, mem_time, mem_total, mem_free, ecc_errors, energy, temperature, fan_speed
  - CPU: load_one, load_five, load_fifteen, proc_run, proc_total, cpu_num, cpu_speed, uptime, cpu_user, cpu_nice, cpu_system, cpu_idle, cpu_wio, cpu_intr, cpu_sintr, interupts, contexts
  - Memory: mem_total, mem_free, mem_shared, mem_buffers, mem_cached, swap_total, swap_free, page_in, page_out, swap_in, swap_out
  - Disk IO: disk_total, disk_free, part_max_used, reads, bytes_read, read_time, writes, bytes_written, write_time
- Application (maintained by application)
  - HTTP: method_option_count, method_get_count, method_head_count, method_post_count, method_put_count, method_delete_count, method_trace_count, method_connect_count, method_other_count, status_1xx_count, status_2xx_count, status_3xx_count, status_4xx_count, status_5xx_count, status_other_count

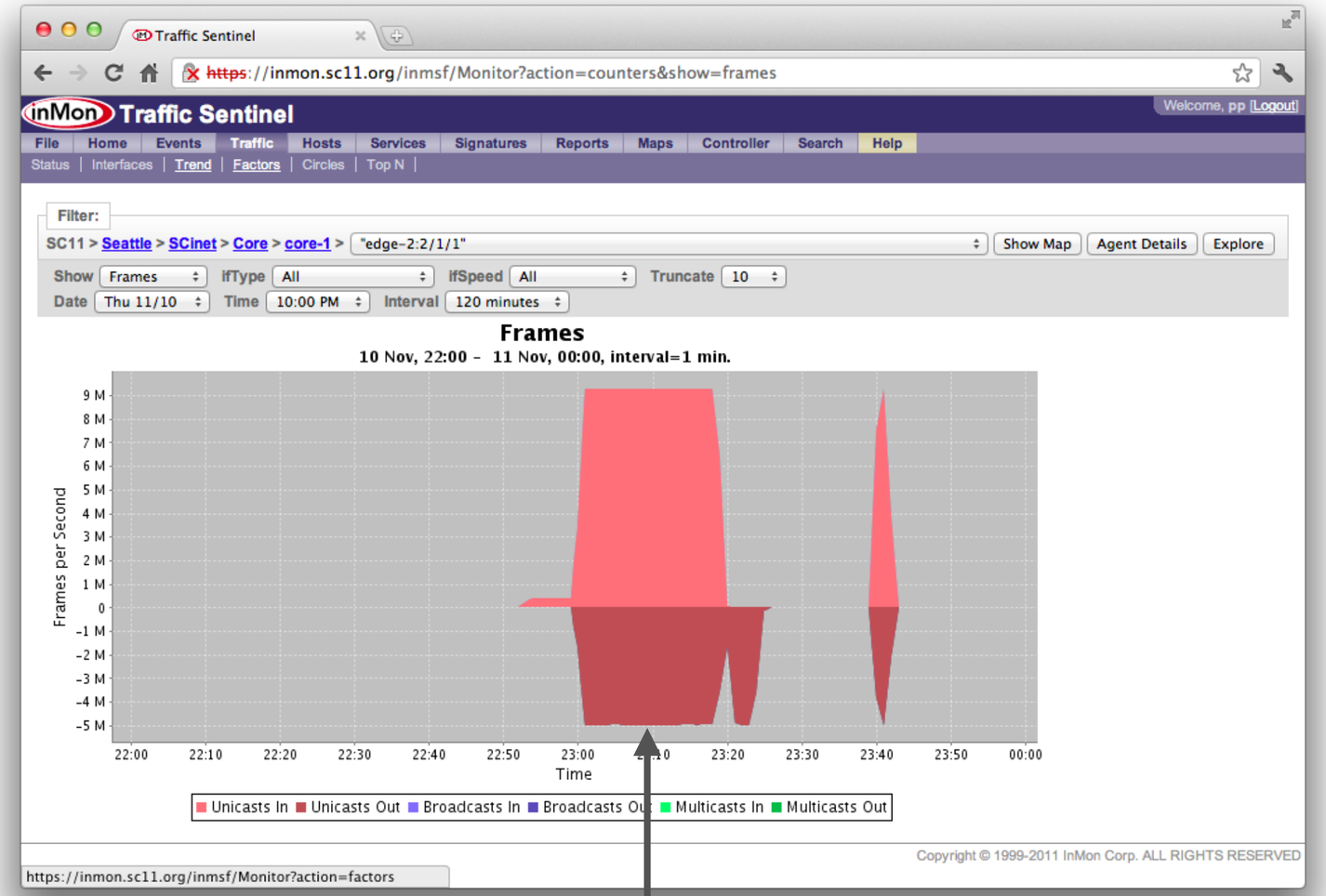
SFLOW'S SCALABLE PUSH PROTOCOL

- Simple
  - standard structures containing densely packed blocks of counters
  - extensible (tag, length, value)
  - RFC 1832: XDR encoded (big endian, quad-aligned, binary) - simple to encode/decode
  - unicast UDP transport
- Minimal configuration
  - collector address
  - polling interval
- Cloud friendly
  - flat, two tier architecture: many embedded agents → central "smart" collector
  - sFlow agents automatically start sending metrics on startup, automatically discovered
  - eliminates complexity of maintaining polling daemons (and associated configurations)

COUNTERS AREN’T ENOUGH

Don’t tell you why there is a spike

- Counters tell you there is a problem, but not why.
- Counters summarize performance by dropping high cardinality attributes:
  - ip addresses
  - protocols
  - ports
- Need to be able to efficiently disaggregate counter by attributes in order to understand root cause of performance problems
- How do you get this data when there are millions of packets per second?



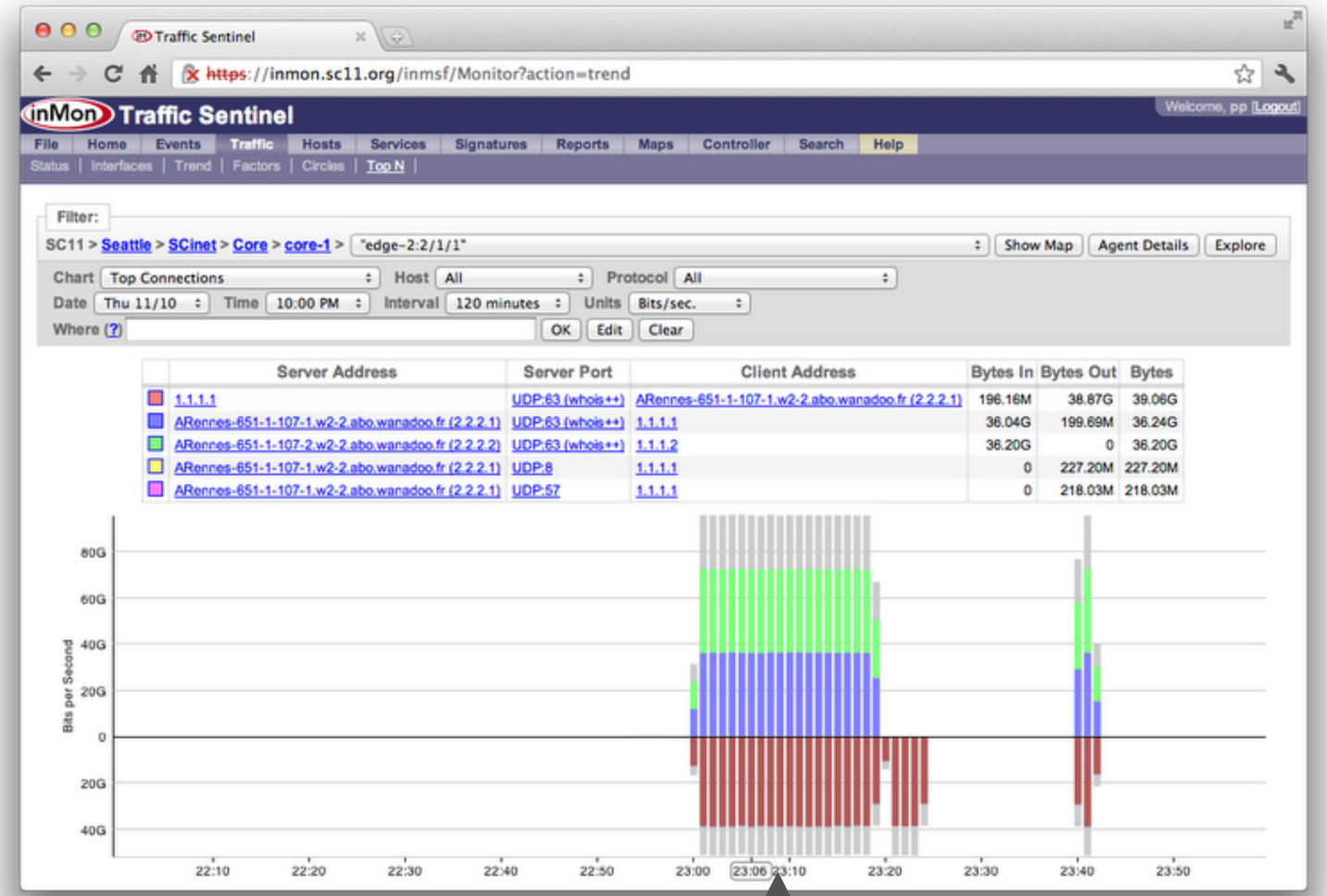
Why the spike in traffic?

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

RANDOM SAMPLING

Scaleable method of adding details

- Random sampling is lightweight
- Unbiased results with known accuracy
- Sampled packet header (128 bytes)
- Forwarding state associated with sampled packet (e.g. ingress/egress port, VLAN, next hop, CIDR etc.)
- Linux kernel instrumenation randomly samples packets
- Offload to hardware instrumentation on network switch scales functionality to Terabit/s speeds
- Identify top sources, destinations, connections, protocols



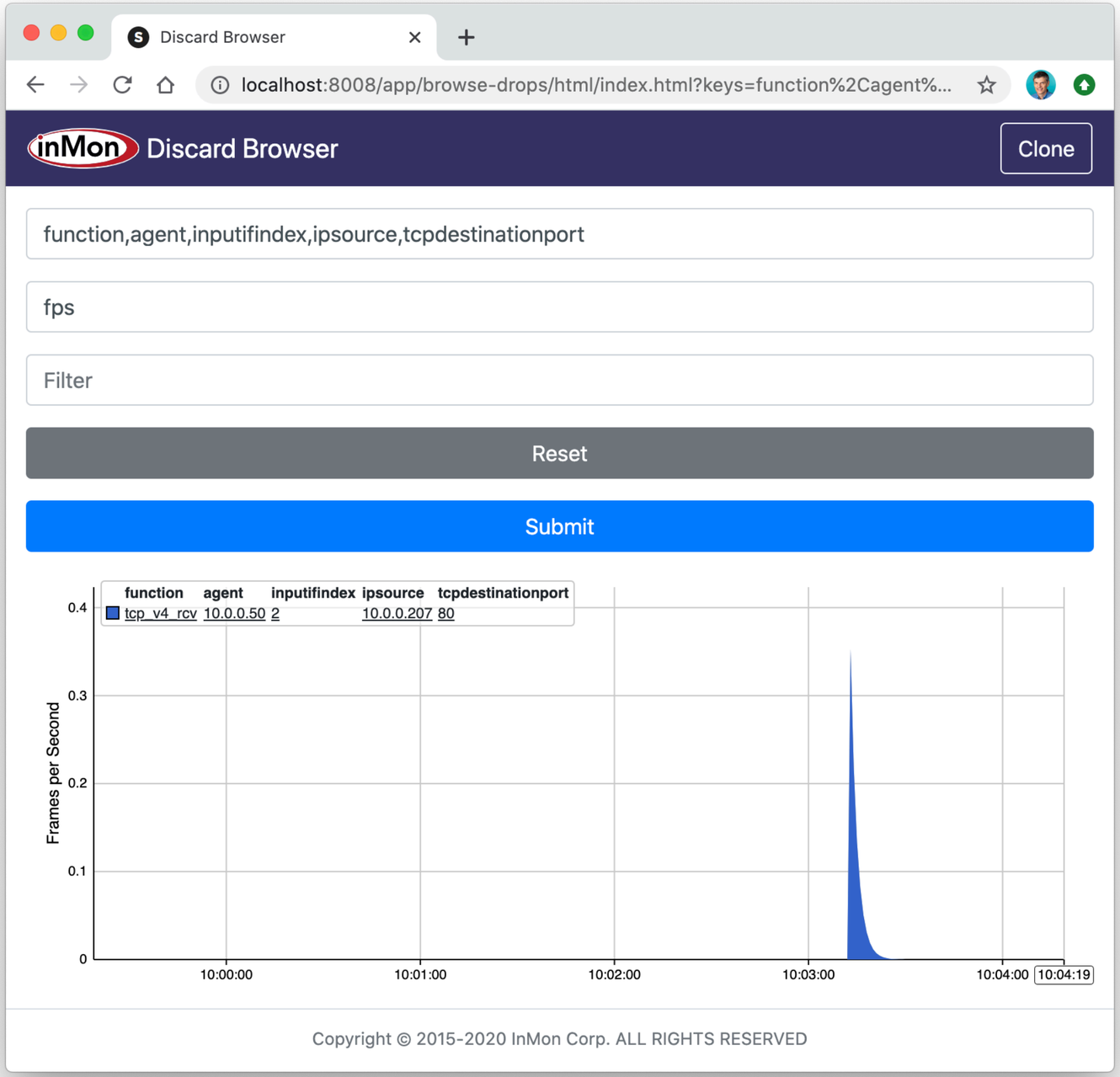
Break out traffic by client, server and port

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

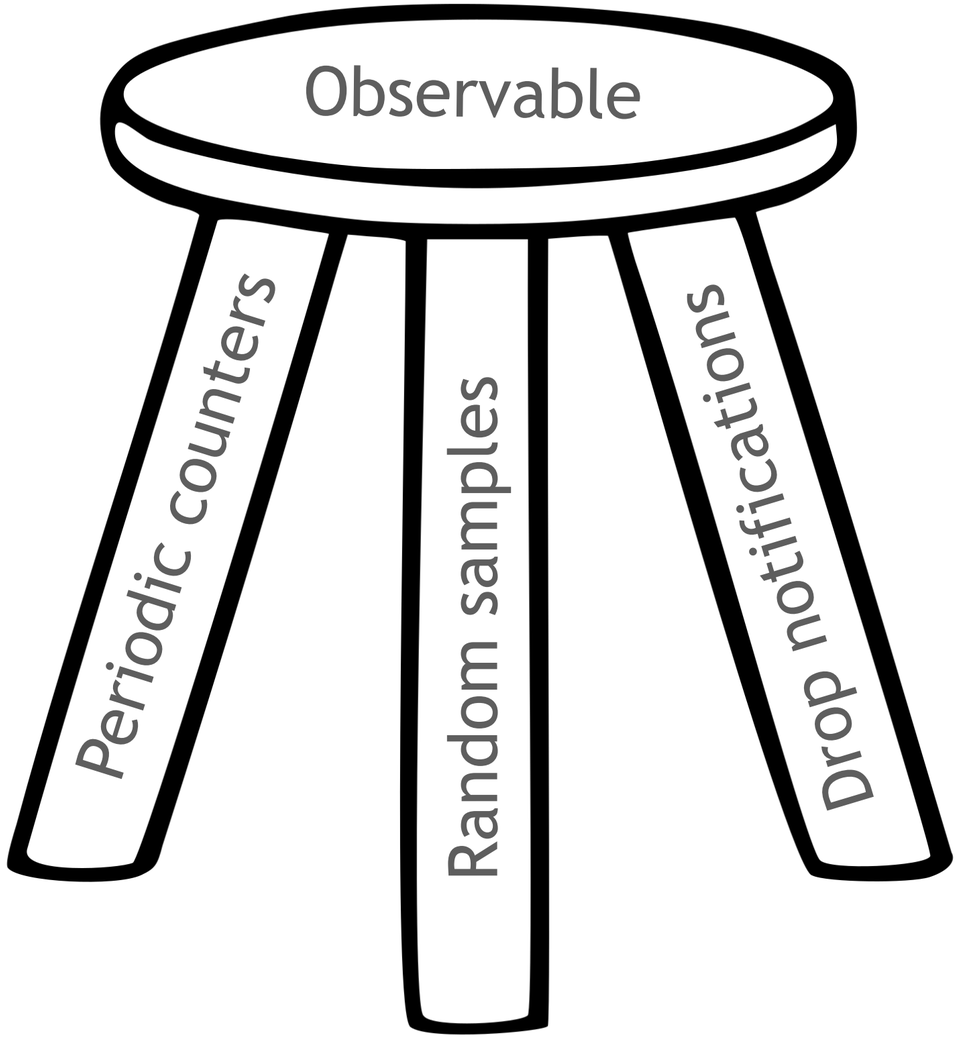
NEW! PACKET DROP EVENTS

Details of rare but critical drop events

- Discard counters tell you that packets are being dropped, but not root cause
- Packet discards are rare and unlikely to be sampled
- Packet discards can severely impact performance and availability of critical services
- Packet discard notifications in sFlow:
  - Discarded packet header (128 bytes)
  - Reason for dropping packet
- Linux kernel netlink drop_monitor API reports each dropped packet and the reason it was dropped
- Offload to hardware instrumentation on a network switch scales functionality to Terabit/s network speeds
- Identify sources, destinations, protocols, locations of discards



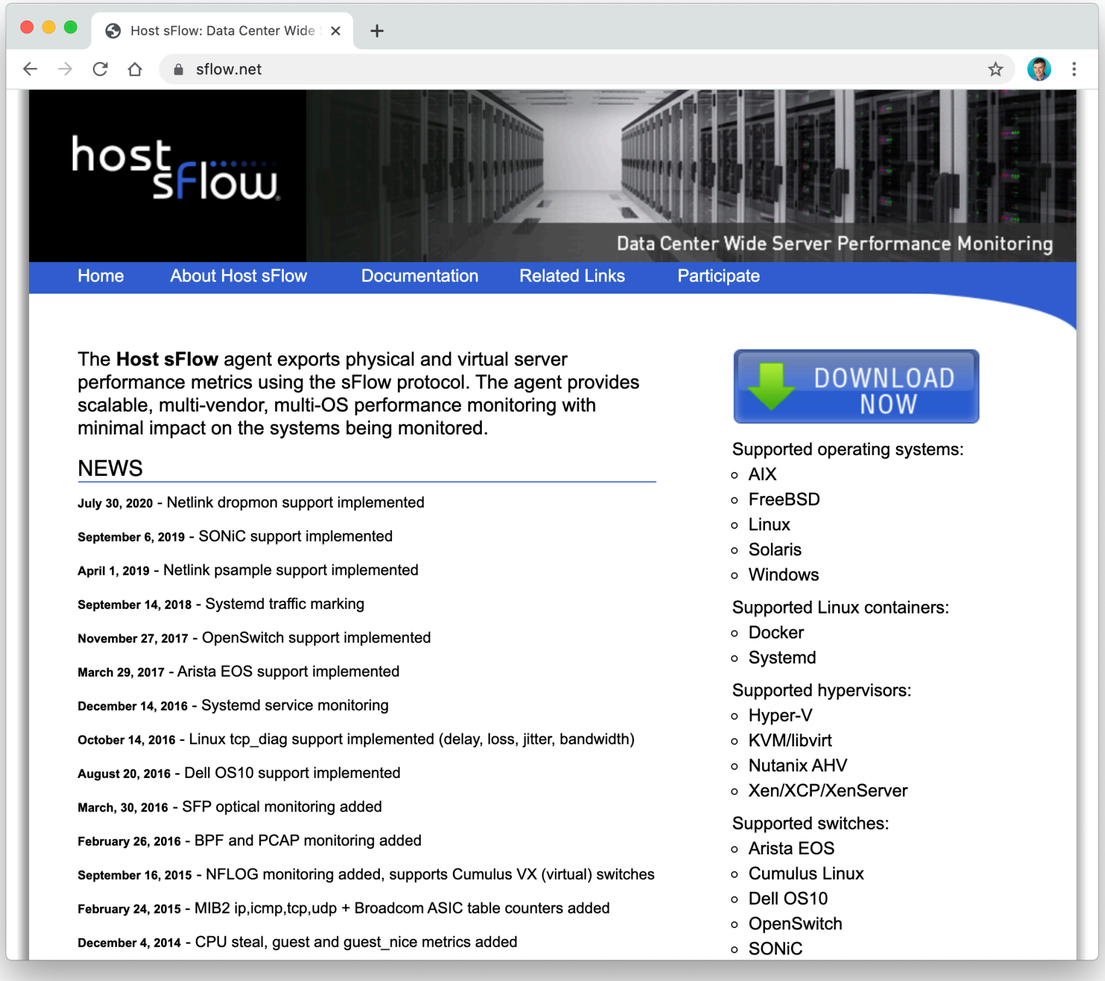
MEASUREMENTS STREAMS COMBINE FOR OBSERVABILITY



sFlow's integrated data model combines measurement streams for comprehensive view

SFLOW IS AN INDUSTRY STANDARD

Broad multi-vendor support ensures observability of physical network



HOST SFLOW

Extends observability into software network edge

docker

kubernetes

KVM

NVIDIA

systemd

Xen Cloud Platform

GPU CLUSTER DEMO

Configure sFlow agents

Edit /etc/hsflowd.conf

Host sFlow agent pre-installed on Cumulus Linux

Download server package from https://sflow.net

Same software and configuration on switches and servers minimises operational complexity

```
# Minimal configuration
sflow {
 collector { ip = 10.31.234.46 }
}
```

sFlow-RT analyzer

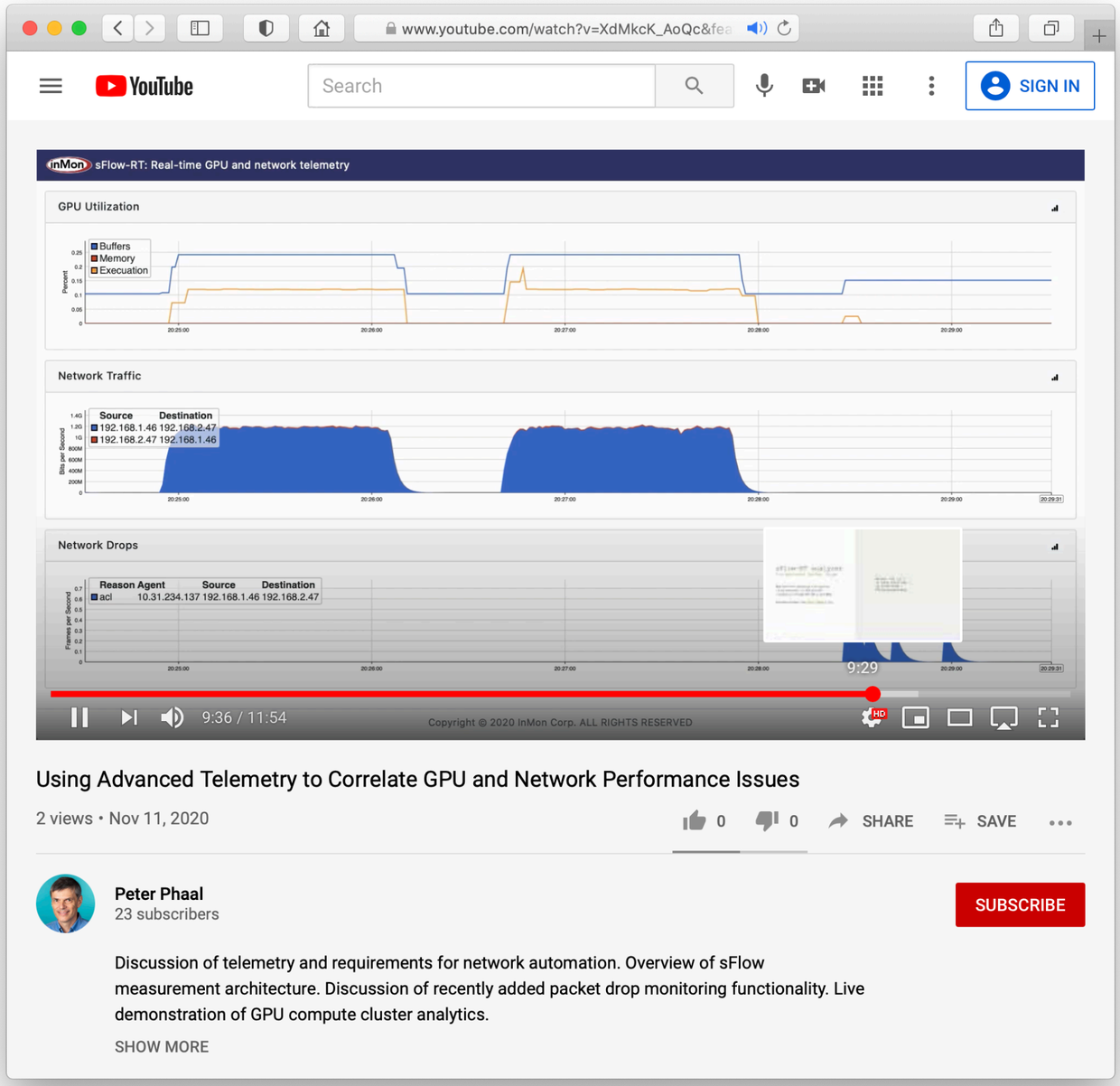
Pre-packaged Docker image

Real-time sFlow analytics as a microservice:

- sFlow telemetry in on UDP port 6343
- Analytics out through REST API on port 8008

Download packages from https://sflow-rt.com

```
docker run -d \
-p 6343:6343/udp \
-p 8008:8008 \
sflow/prometheus
```



Talk and demo available on YouTube

https://youtu.be/XdMkcK_AoQc