Data network visibility and control

“You can’t control what you can’t measure”

Tom DeMarco
Reason 1: Widely supported industry standard
Reason 1: Widely supported industry standard
Reason 1: Widely supported industry standard
Reason 2: Comprehensive

Traffic Sentinel

• All Devices
• All Servers
• All Applications
• All the time

http://www.sflow.org
sFlow Overview: replaces counter polling

“De-synchronized, Parallel Push”

• sFlow agent automatically pushes full set of SNMP ifTable counters\(^1\)
• Compared to SNMP polling, counter push results in 10-20x fewer packets on network, reduces CPU load on switch and on network management software (XDR is easier to encode/decode than SNMP)
• Single sFlow collector can easily monitor 200,000 switch ports with 1 minute granularity. SNMP polling with 5 minute granularity requires 5-10 collectors.

1. ifIndex, ifType, ifType, ifSpeed, ifDirection, ifAdminStatus, ifOperStatus, ifInOctets, ifInUcastPkts, ifInMulticastPkts, ifInBroadcastPkts, ifInDiscards, ifInErrors, ifInUnknownProtos, ifOutOctets, ifOutUcastPkts, ifOutMulticastPkts, ifOutBroadcastPkts, ifOutDiscards, ifOutErrors, ifPromiscuousMode
Traffic Sentinel: Interface counters

- 200,000+ ports
- 1-minute granularity
- Thresholds/alerts
- Compare all interfaces
sFlow Overview: monitors all protocols

- Simple agents: packet headers sent to sFlow collector for decoding.
- Easier to add decodes to central collector than to every device in a multi-vendor network (e.g. IPv6, FCoE etc.)
- Captures complex layering (e.g. MAC/VLAN/MPLS/IPv4/IPv6): critical for tracing packet paths through network.
Traffic Sentinel: Traffic Breakdown

• MAC, VLAN, IP, IPv6, TCP, UDP, MPLS, TRILL, RTP etc. (over 100 fields)
• 1-minute granularity
• Thresholds/alerts
• Automatic de-duplication
• Subnet rollups
TRILL Fabrics

- sFlow on existing Ethernet switches captures the following TRILL fields
  - TRILL RBridge Addresses
  - Forwarding path
  - Hop count
  - Broadcast bit
- As well as inner/outer MAC addresses and encapsulated TCP/IP etc. data
- sFlow monitoring provides information about path utilization, applications using a path etc. Critical for load balancing and troubleshooting TRILL deployments

<table>
<thead>
<tr>
<th>Outer Destination MAC Address</th>
<th>Ethertype = IEEE 802.1Q</th>
<th>UP</th>
<th>C</th>
<th>Outer VID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outer Source MAC Address</td>
<td>Ethertype = TRILL</td>
<td>V</td>
<td>Hop Limit</td>
<td>M</td>
</tr>
<tr>
<td>+-------------------------------+-------------------------+----</td>
<td>-----</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Egress RBridge Address</td>
<td>Ingress RBridge Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+-------------------------------+-------------------------+----</td>
<td>-----</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inner Destination MAC Address</td>
<td>Ethertype = IEEE 802.1Q</td>
<td>UP</td>
<td>C</td>
<td>Inner VID</td>
</tr>
<tr>
<td>+-------------------------------+-------------------------+----</td>
<td>---</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inner Source MAC Address</td>
<td>Ethertype = IEEE 802.1Q</td>
<td>UP</td>
<td>C</td>
<td>Inner VID</td>
</tr>
</tbody>
</table>

Original Ethernet Payload

New FCS
TRILL Fabrics
sFlow Overview: captures packet path

- Each packet sample captures the forwarding path for the packet
- Threading together the paths provides a constantly updating picture of network topology and host locations
- The combination of forwarding table data and packet headers provides an integrated view of traffic. E.g. you can filter on forwarding attributes (VLAN, MPLS, route) and see traffic, or filter on traffic and identify forwarding paths.
Traffic Sentinel: Multivendor topology discovery

Uses:
- sFlow
- CDP
- FDP
- LLDP
- Spanning-tree
- Bridge-tables
- and more...

- Auto-layout
- Mouse-wheel zoom
- Show Status, Traffic (refreshed every minute)
Traffic Sentinel: End-host location

Uses:
sFlow
SNMP

DNS
IP
MAC
Port

With sFlow, host locations can be updated within 60 seconds
Simple agents
sFlow Architecture

Simple agents
sFlow Architecture

Simple agents ➔ Smart collector
sFlow Architecture

Simple agents

Easy to implement

Smart collector
sFlow Architecture

Simple agents → Smart collector

- Easy to implement
- Embedded, wire-speed
sFlow Architecture

Simple agents: Easy to implement, embedded, wire-speed, low cost

Smart collector
sFlow Architecture

Simple agents  Smart collector

Easy to implement
Embedded, wire-speed
Low cost
Counter-push

Monday, July 16, 12
sFlow Architecture

Simple agents → Smart collector

- Easy to implement
- Embedded, wire-speed
- Low cost
- Counter-push
- + packet/transaction sampling
sFlow Architecture

Simple agents
- Easy to implement
- Embedded, wire-speed
- Low cost
- Counter-push
- + packet/transaction sampling

Smart collector
- Network-wide, integrated, visibility and control
sFlow Architecture

Simple agents → sFlow → Smart collector

Internet

sFlow Analyzer
sFlow Architecture

Simple agents → Internet → Smart collector

Network
Internet Architecture

Simple agents
Network
Servers
Virtual switches

sFlow
Analyzer

Smart collector

Monday, July 16, 12
Simple agents
Network
Servers
Virtual switches
Virtual servers

Internet

sFlow Analyzer

sFlow

Smart collector

Monday, July 16, 12
Cross-layer correlation: Application, Host and Network

Standard measurements from different sources designed to be joinable at the collector

e.g. application response time increase correlated directly to congestion on network path
The Host sFlow agent exports physical and virtual server performance metrics using the sFlow protocol. The agent provides scalable, multi-vendor, multi-OS performance monitoring with minimal impact on the systems being monitored.

NEWS

July 7, 2011 - Ganglia 3.2 supports Host sFlow
March 4, 2011 - FreeBSD support implemented
February 22, 2011 - Windows load averages added
December 17, 2010 - iptables/ULOG traffic monitoring added
October 15, 2010 - KVM/libvirt support implemented
October 8, 2010 - Host-sFlow version 1.06 released (XCP 0.5 / XenServer 5.6 FP1)
September 23, 2010 - Windows installer implemented
September 1, 2010 - Host-sFlow 1.3 released adding Open vSwitch integration
July 29, 2010 - New project wiki site launched
July 23, 2010 - Host-sFlow version 1.0 released
June 16, 2010 - DNS service discovery implemented (DNS-SD)
May 1, 2010 - Host sFlow project started

The latest release of the host-sflow sources can be downloaded at http://sourceforge.net/projects/host-sflow

Supported operating systems:
- FreeBSD
- Linux
- Windows

Supported hypervisors:
- KVM/libvirt
- Xen/XCP/XenServer
Examples:
NFS/CIFS transactions  (file path, bytes, response-time, socket)
HTTP requests        (URL, user-agent, mime-type, bytes, response-time, socket)
Memcached lookups     (key, value-bytes, hit/miss, socket)
Database queries      (query#, response-time, socket)

Application layer measurements much more valuable when correlated with performance of every component in underlying infrastructure!
Host sFlow distributed agent

```
memcached
httpd
java
/etc/hsflowd.auto
sflowovsd
Open vSwitch
/ovs-vsctl
sFlow collector
/etc/hsflowd.conf

other
/other
JSON-API
DNS-SD
DNS server
```

- Host sFlow distributed agent
- `/etc/hsflowd.conf`
- `/etc/hsflowd.auto`
- `hsflowd`
Host sFlow - JSON API

```json
{
  "flow_sample": {
    "app_name": "myapp",
    "sampling_rate": 100,
    "app_operation": {
      "operation": "task.start",
      "attributes": "id=123&user=root",
      "status_descr": "OK",
      "status": 0,
      "req_bytes": 43,
      "resp_bytes": 234,
      "uS": 2000
    },
    "app_initiator":{"actor":"123"},
    "app_target":{"actor":"231"},
    "extended_socket_ipv4": {
      "protocol": 6,
      "local_ip": "10.0.0.1",
      "remote_ip": "10.0.0.23",
      "local_port": 123,
      "remote_port": 43032
    }
  }
}
```

- UDP msg to hsflowd on localhost:36343
- most fields optional
- sampling in app. or hsflowd
- counters in app. or hsflowd
- hsflowd sends binary sFlow-APPLICATION feed to configured collectors.
### XenMotion bandwidth

<table>
<thead>
<tr>
<th>Server Address</th>
<th>Server Port</th>
<th>Client Address</th>
<th>Bytes From Server</th>
<th>Bytes To Server</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>xenserver2 (10.0.0.16)</td>
<td>TCP:80 (www-http)</td>
<td>xenserver1 (10.0.0.20)</td>
<td>2.73M</td>
<td>149.54M</td>
<td>152.26M</td>
</tr>
<tr>
<td>ganglia (10.0.0.112)</td>
<td>TCP:11211 (memcache)</td>
<td>xenvm1 (10.0.0.150)</td>
<td>773.75K</td>
<td>1.07M</td>
<td>1.85M</td>
</tr>
<tr>
<td>openfiler (10.0.0.18)</td>
<td>TCP:3260 (iscsi-target)</td>
<td>xenserver1 (10.0.0.20)</td>
<td>251.93K</td>
<td>21.21K</td>
<td>273.14K</td>
</tr>
<tr>
<td>openfiler (10.0.0.18)</td>
<td>TCP:3260 (iscsi-target)</td>
<td>xenserver2 (10.0.0.16)</td>
<td>167.92K</td>
<td>72.85K</td>
<td>240.77K</td>
</tr>
<tr>
<td>openfiler (10.0.0.18)</td>
<td>TCP:2049 (nfs)</td>
<td>xenserver1 (10.0.0.20)</td>
<td>0</td>
<td>211.39K</td>
<td>211.39K</td>
</tr>
</tbody>
</table>

---

Monday, July 16, 12
XenMotion bandwidth
Application throughput and response time

---

### Table: Memcache Protocol, Command, Status, Key, Duration

<table>
<thead>
<tr>
<th>Memcache Protocol</th>
<th>Memcache Command</th>
<th>Memcache Status</th>
<th>Memcache Key</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>SET</td>
<td>STORED</td>
<td>bruts</td>
<td>16.70μ</td>
</tr>
<tr>
<td>ASCII</td>
<td>GET</td>
<td>EXISTS</td>
<td>bruts</td>
<td>4.25μ</td>
</tr>
<tr>
<td>ASCII</td>
<td>GET</td>
<td>NOT FOUND</td>
<td>bruts</td>
<td>4.21μ</td>
</tr>
</tbody>
</table>

---

### Duration in Seconds

- 00:50 - 00:55
- 01:00 - 01:05
- 01:10 - 01:15
- 01:20 - 01:25
- 01:30 - 01:35
- 01:40 - 01:45

---

Monday, July 16, 12
Application throughput and response time

<table>
<thead>
<tr>
<th>Memcache Protocol</th>
<th>Memcache Command</th>
<th>Memcache Status</th>
<th>Memcache Key</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>GET</td>
<td>NOT FOUND</td>
<td>brutis</td>
<td>938.33</td>
</tr>
<tr>
<td>ASCII</td>
<td>SET</td>
<td>STORED</td>
<td>brutis</td>
<td>130.00</td>
</tr>
<tr>
<td>ASCII</td>
<td>GET</td>
<td>EXISTS</td>
<td>brutis</td>
<td>37.50</td>
</tr>
</tbody>
</table>
Application throughput and response time

Drop in throughput during XenMotion of Memcache client
sFlow-APPLICATION example: transactions

HTTP Methods
1 Jun, 04:00 – 1 Jun, 05:00, interval=1 min.

HTTP Status
1 Jun, 04:00 – 1 Jun, 05:00, interval=1 min.
sFlow-APPLICATION example: transaction detail

<table>
<thead>
<tr>
<th>URI Host</th>
<th>URI Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>goku.brightcove.com</td>
<td>/1pix.gif</td>
<td>0.63</td>
</tr>
<tr>
<td><a href="http://www.milty.com">www.milty.com</a></td>
<td>/</td>
<td>0.57</td>
</tr>
<tr>
<td><a href="http://www.milty.com">www.milty.com</a></td>
<td>/module/setup.php</td>
<td>0.43</td>
</tr>
<tr>
<td>c.brightcove.com</td>
<td>/services/messagebroker/amf</td>
<td>0.40</td>
</tr>
<tr>
<td><a href="http://www.milty.com">www.milty.com</a></td>
<td>/adspace.php</td>
<td>0.37</td>
</tr>
</tbody>
</table>

Monday, July 16, 12
sFlow-APPLICATION example: latency

<table>
<thead>
<tr>
<th>URI Host</th>
<th>URI Path</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>talkgadget.google.com</td>
<td>/talkgadget/channel/bind</td>
<td>216.39M</td>
</tr>
<tr>
<td>ds.addthis.com</td>
<td>/red/psi/sites/www.mttv.com/p.json</td>
<td>3.47M</td>
</tr>
<tr>
<td>c1142172.cdn.cloudflare.rackspacecloud.com</td>
<td>/100x75/0c08ui84r.jpg</td>
<td>1.13M</td>
</tr>
<tr>
<td>c1142172.cdn.cloudflare.rackspacecloud.com</td>
<td>/100x75/eghxushmko.jpg</td>
<td>1.12M</td>
</tr>
<tr>
<td>c1142172.cdn.cloudflare.rackspacecloud.com</td>
<td>/100x75/gdsussbdf4.jpg</td>
<td>1.12M</td>
</tr>
</tbody>
</table>
Why Monitor Everything?

1. Troubleshooting - always have context
Why Monitor Everything?

1. Troubleshooting - always have context

trace path

locate hosts
Why Monitor Everything?

2. Put Network and Server teams on same page

3. Full “Observability” required for automated control
OpenFlow + sFlow = Adaptive Control
OpenFlow + sFlow = Adaptive Control
OpenFlow + sFlow = Adaptive Control

• Detailed, low-latency measurements from sFlow allows OpenFlow controller to adapt network to changing traffic patterns (load balancing, DDoS mitigation etc.).
OpenFlow + sFlow = Adaptive Control

- Detailed, low-latency measurements from sFlow allows OpenFlow controller to adapt network to changing traffic patterns (load balancing, DDoS mitigation etc.).

- OpenFlow can be optimized for efficiency (e.g. by using wildcards), sFlow provides visibility to detect and manage large flows.
Network OS

Application

... ...

Application

Network OS

Control Plane

Open APIs

Data Plane

Hosts

Open APIs

Configuration

Forwarding

Visibility

NETCONF/OF-Config

OpenFlow

sFlow
sFlow vs NetFlow/IPFIX

- Packets
- i/f counters

Flow process:
1. Sample
2. Decode
3. Hash
4. Flow cache
5. Flush
6. Send

NetFlow/IPFIX process:
- Sample
- Poll
- Send

NetFlow/IPFIX vs sFlow.
sFlow vs NetFlow/IPFIX: system-wide NetFlow/IPFIX

Applications:
• TCP/IP Accounting
• Security (associations)

Collector

Limited fields
Large, unpredictable latency

Packets

NetFlow/IPFIX
sFlow vs NetFlow/IPFIX: system-wide sFlow

- All fields
- All protocols
- All encapsulations
- Sub-second latency
- De-synchronized
- I/F counters too

Applications:
- Almost anything
  (software defined)
More Information

host-sflow.sourceforge.net

blog.sflow.com

sflow.org