



# SharkFest '18 US



## sFlow: Theory and Practice of a Sampling Technology

and Its Analysis with Wireshark

Simone Mainardi, PhD

ntop

mainardi@ntop.org

@simonemainardi



# Outline



- What is sFlow? When is it useful and when it is not
- How does sFlow work? Agents, collectors, packets and sampling techniques
- Using Wireshark to master sFlow



# What is sFlow? [1/2]



- sFlow is a sampling technology designed to export
  - Network devices information (à la SNMP)
  - Packets traversing network devices (à la ERSPAN)



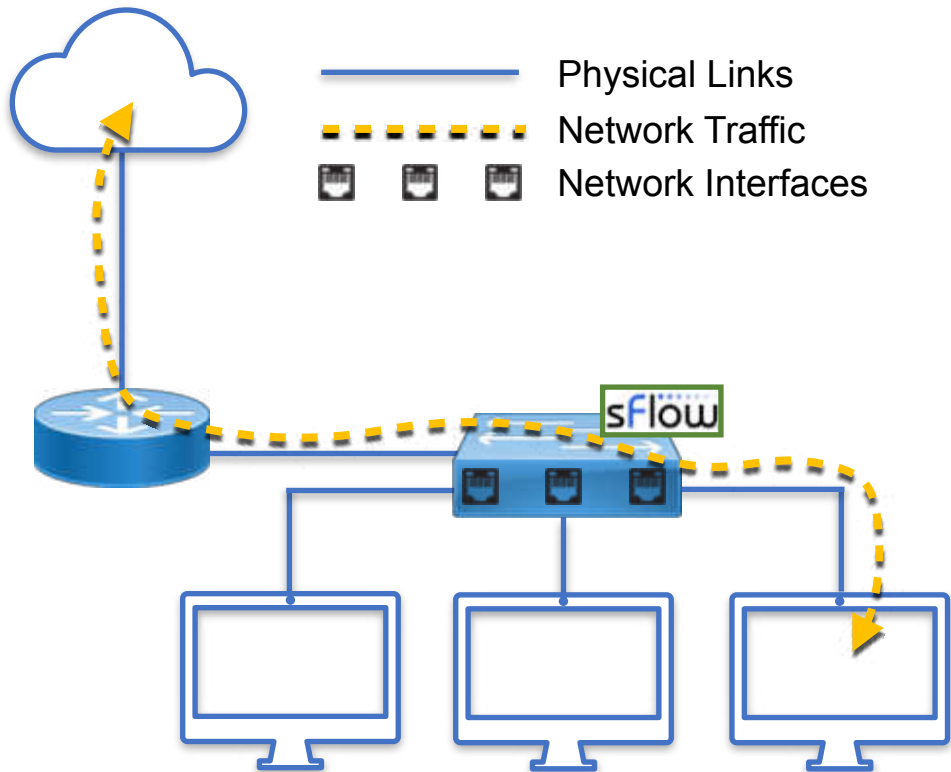
# What is sFlow? [2/2]



- Network-wide visibility is obtained by means of configurable sampling
  - Counters samples
  - Flow samples
- Samples are periodically put in sFlow UDP datagrams and pushed over the network



# sFlow Visibility



- Device visibility 
  - Counter Samples

- Traffic visibility 
  - Flow Samples



# sFlow Counter Samples



- Interfaces status, speed, type
- Cumulative input and output bytes/packets, errors, ...



Network Traffic

Network Interfaces

No.	Time	Source	Destination	Protocol	Length	NumSamples
1	0.000000	72.0.112.100	10.0.3.250	sFlow	570	4
2	2.024941	72.0.112.100	10.0.3.250	sFlow	570	4
3	3.204524	72.0.112.100	10.0.3.250	sFlow	530	2

```
▼ Counters sample, seq 480927
0000 0000 0000 0000 0000 ..... = Enterprise: standard sFlow (0)
..... 0000 0000 0010 = sFlow sample type: Counters sample (2)
Sample length (byte): 160
Sequence number: 480927
0000 0000 ..... = Source ID type: 0
.... 0000 0000 0000 0000 1000 0001 = Source ID index: 129
Counters records: 2
├ Ethernet interface counters
├ Generic interface counters
  0000 0000 0000 0000 0000 ..... = Enterprise: standard sFlow (0)
..... 0000 0000 0001 = Format: Generic interface counters (1)
Flow data length (byte): 88
Interface index: 129
Interface Type: 6
Interface Speed: 1000000000
Interface Direction: Full-Duplex (1)
.....1 = IfAdminStatus: Up
.....1 = IfOperStatus: Up
Input Octets: 2748679350265805
Input Packets: 821400103
Input Multicast Packets: 2931491300
Input Broadcast Packets: 347804767
Input Discarded Packets: 0
Input Errors: 0
Input Unknown Protocol Packets: 0
Output Octets: 13117755552772
Output Packets: 1177201173
Output Multicast Packets: 2077420003
Output Broadcast Packets: 70506004
Output Discarded Packets: 0
Output Errors: 0
Promiscuous Mode: 1
▼ Counters sample, seq 480928
```





# sFlow Flow Samples



- Random selection of a fraction of the packets observed



Network Traffic

Network Interfaces

No.	Time	Source	Destination	Protocol	Length	NumSamples
1	0.000000	72.0.132.100	10.0.3.250	sFlow	878	4
2	2.024941	72.0.132.100	10.0.3.250	sFlow	878	4
3	3.204524	72.0.132.100	10.0.3.250	sFlow	538	2

```

Flow sample, seq 947406146
0000 0000 0000 0000 0000 ..... = Enterprise: standard sFlow (0)
..... = sFlow sample type: Flow sample (1)
Sample length (byte): 200
Sequence number: 947406146
0000 0000 ..... = Source ID class: 0
..... = Index: 130
Sampling rate: 1 out of 1024 packets
Sample pool: 2957455472 total packets
Dropped packets: 0
Input interface (ifIndex): 130
Multiple outputs: unknown number
Flow record: 2
* Raw packet header
  0000 0030 0000 0000 0000 ..... = Enterprise: standard sFlow (0)
  Format: Raw packet header (1)
  Flow data length (byte): 144
  Header protocol: Ethernet (1)
  Frame Length: 1366
  Payload removed: 8
  Original packet length: 128
  * Header of sampled packet: 0105e0008000900b394436080045000540981f4200f731...
    > Ethernet II, Src: LannerEL_39:44:36 (00:90:0b:39:44:36), Dst: IPv4cast_66:09 (01:00:5e:00:66:09)
      > Internet Protocol Version 4, Src: 192.168.47.34, Dst: 224.0.104.9
        0100 ..... = Version: 4
        .... 0101 = Header Length: 20 bytes (5)
        > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
          Total Length: 1344
          Identification: 0x981f (38943)
          Flags: 0x4000, Don't fragment
          Time to Live: 255
          Protocol: UDP (17)
          Header checksum: 0xe6b0 [validation disabled]
          [Header checksum status: Unverified]
  
```



# When is sFlow Useful? [1/2]



- Network-wide estimations of top:
  - Layer-7 application protocols usage (e.g., HTTP, YouTube, Skype)
  - Sources
  - Destinations
  - Conversations
  - Ports
- Detect volumetric attacks







# When is sFlow Useful? [2/2]



- Capacity planning
- Traffic engineering (eg., decide to establish a new peering, buy more bandwidth)
- Network topology adjustments (e.g., bring guys communicating the most onto the same link)
- Detect network issues (e.g, switches port status changes)
- Link congestion





# When is sFlow NOT Useful? [1/2]



- Detect bottom-sources, -destinations, -ports, -Layer-7 application protocols, ...
- Feed signature-based  
Intrusion Prevention/Intrusion Detection Systems  
(IDS/IPS)





# When is sFlow NOT Useful? [2/2]



- Stateful protocols analyses
  - No SEQ number analysis
- Sessions reconstruction
  - No TCP reassembly
- Detect Low-and-Slow network attacks
- Content-based network forensics
  - No extraction of files, images, documents





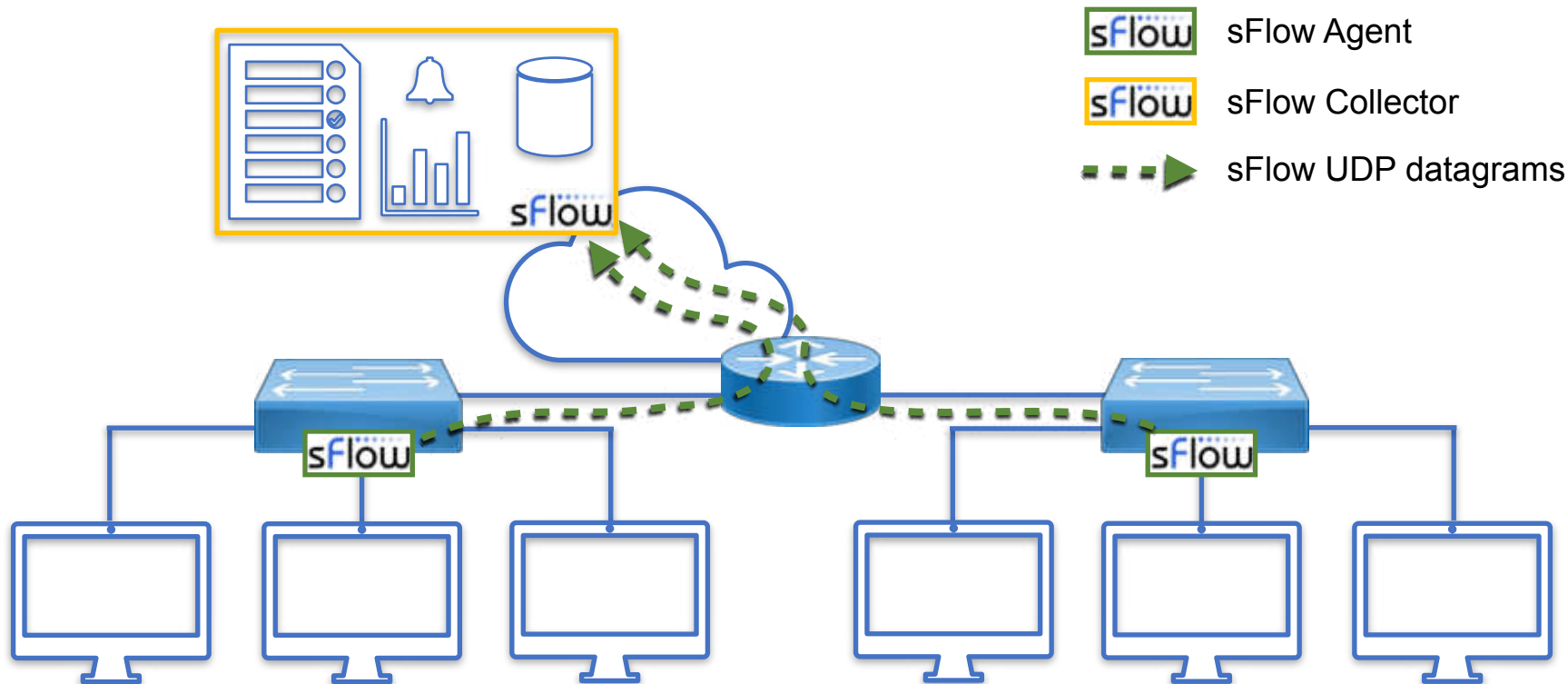
# sFlow Monitoring Systems



- sFlow Agents
  - Embedded in switches
  - Marshal samples into UDP Datagrams to send them to one or more sFlow collectors
- sFlow Collectors
  - Receive UDP Datagrams from sFlow Agents
  - Process received data (e.g., to troubleshoot, create and store traffic time series, alert on unexpected traffic patterns)

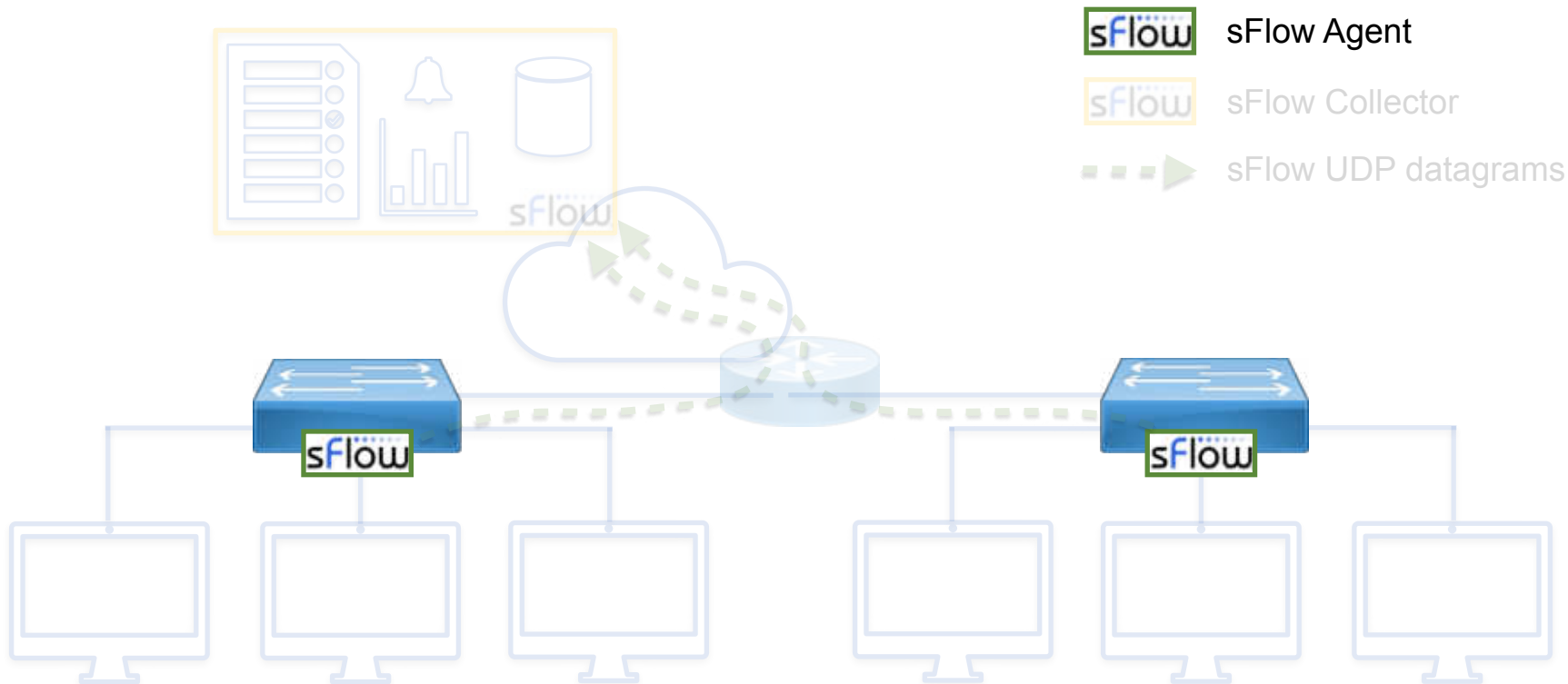


# sFlow Monitoring Systems





# sFlow Monitoring Systems: Agents





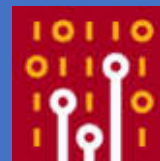
# sFlow Embedded Agents



- Tens of manufacturers
  - A10, Aerohive, AlexaIA, ALUe, Allied Telesis, Arista, Aruba, Big Switch, Brocade, Cisco, Cumulus, DCN, Dell, D-Link, Edge-Core, Enterasys, Extreme, F5, Fortinet, HPE, Hitachi, Huawei, IBM, IP Infusion, Juniper, NEC, Netgear, OpenSwitch, Open vSwitch, Oracle, Pica8, Plexxi, Pluribus, Proxim, Quanta, Silicom, SMC, ZTE, and ZyXEL, etc.
- (Non-exhaustive) list maintained at <https://sflow.org/products/network.php>



# sFlow Software Agents



- Host sFlow agent (<https://github.com/sflow/host-sflow>)
- Oses: AIX, FreeBSD, Linux, Solaris, and Windows
- Docker containers
- Hypervisors: Hyper-V, KVM/libvirt, Nutanix AHV and Xen hypervisors
- Supported switches, Arista EOS, Cumulus Linux, Dell OS10, OpenSwitch







# sFlow Monitoring Systems: Collectors



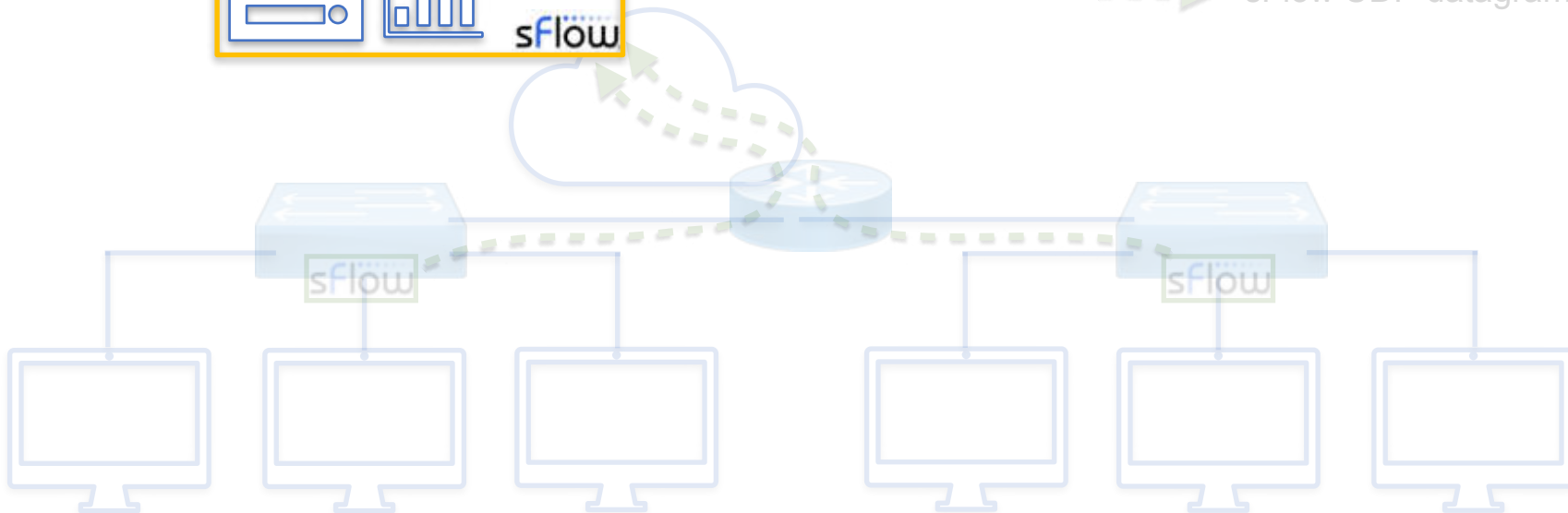
sFlow Agent



sFlow Collector



sFlow UDP datagrams





# sFlow Collectors [1/4]



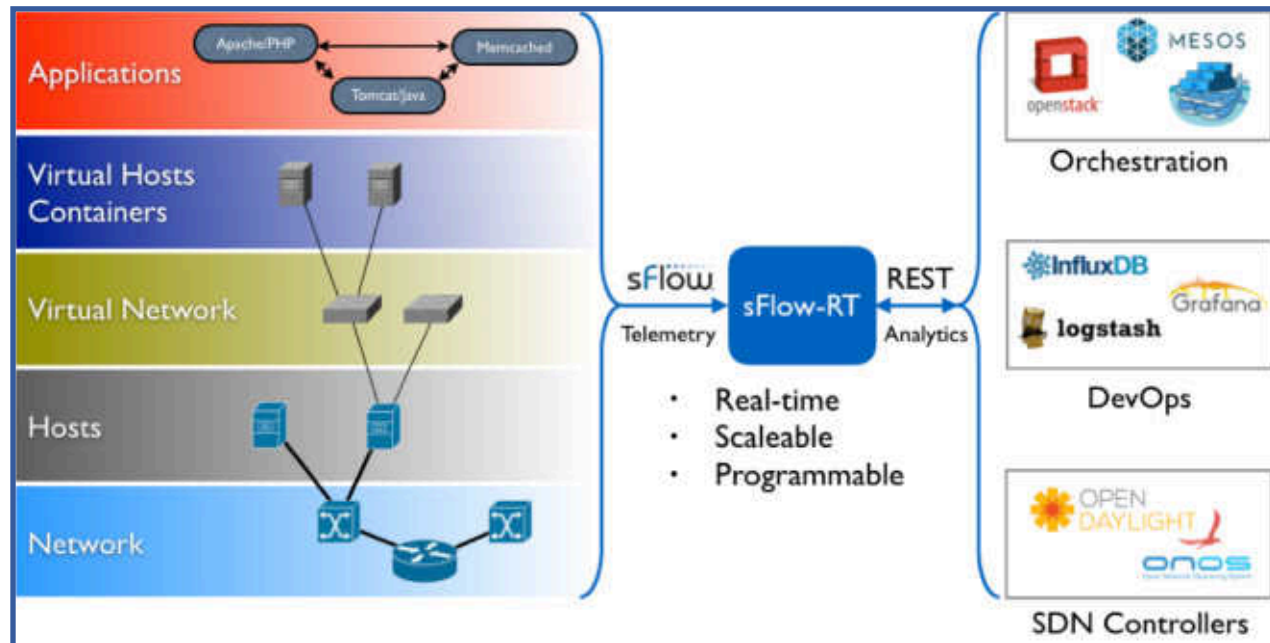
- sFlow Toolkit
  - Basic command line utilities (output to pcap, sFlow to NetFlow, txt)
- sFlowTrend/sFlowTrend-Pro
  - Graphical tool to generate live statistics network interfaces, top sources/destinations, top applications, ...



# sFlow Collectors [2/4]



- sFlow-RT
  - Scriptable collector via REST/JavaScript
  - Retrieve metrics, set thresholds, receive notifications, ...





# sFlow Collectors [3/4]



- ntopng
  - Graphical tool to generate live and historical statistics on sources and destinations, network conversations (who talks to whom), and network interfaces
  - Facilitates the correlation of sources and destinations with the physical ports they are using

	Application	L4 Proto	Client	Server	Duration	Flow Exporter	Bytes	Info
Info	SSL	TCP	79.136.102.9	nopr.inleed.net :53722	0 sec	All Flow Exporters	99 MB	
Info	? Unknown	UDP	nopr.inleed.net :6881	121.211.80.113 :44474	0 sec	Flow Exporter 10.0.2.154		
Info	HTTP	TCP	46.59.102.200	165.231.120.232 :44411	0 sec	Flow Exporter 10.0.2.253	17 MB	
						Flow Exporter 185.189.48.162[server.vali.se]		
						<b>Client</b>	5.97 Mbit/s	23.92 MB



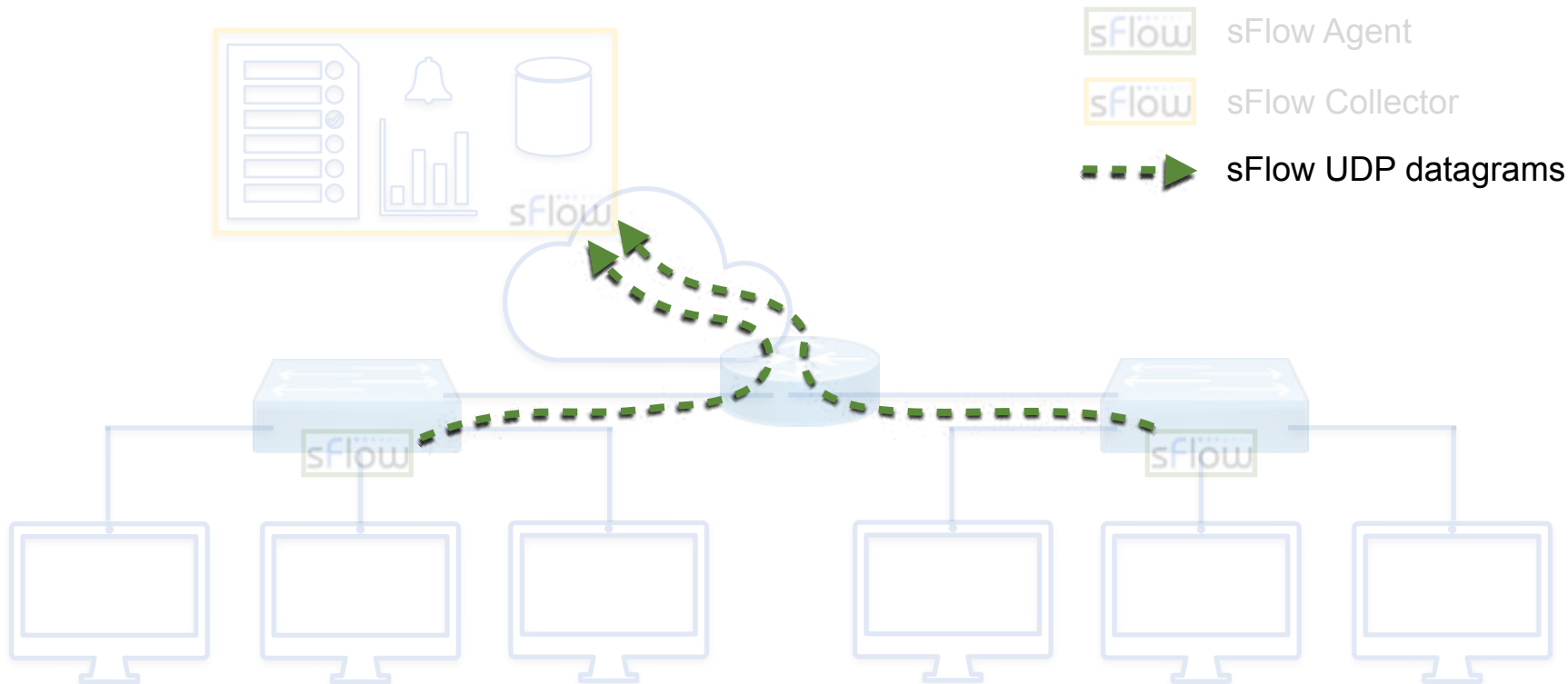
# sFlow Collectors [4/4]



- Wireshark
  - Dissect sFlow traffic
  - Dissect packets in flow samples as if they were regular packets
  - Lua plugin to see aggregated information
- (Non-exhaustive) list available at <https://sflow.org/products/collectors.php>



# sFlow Monitoring Systems: Transport





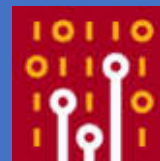
# sFlow Transport



- sFlow works over UDP
  - Reduced memory and CPU wrt TCP
  - Robust in congested networks
    - Higher delays and lost packets increase but there is no need to buffer any data nor to wait for retransmissions
- sFlow packets are sequenced so the application can detect losses



# sFlow ↑ Push Architecture [1/2]



- sFlow UDP datagrams are periodically and unsolicitedly sent by each agent to one or more collectors
- Collectors don't need to discover new agents
- Reduced workload
  - Collectors don't have to generate reqs and match reqs/resps
  - Agents don't have to parse and process reqs





# sFlow ↑ Push Architecture [2/2]



- Increased security
  - Agents don't have to listen on open ports
  - Firewalls only have to allow mono-directional agent-to-collector communications
- Reduced latency
  - No need to establish connections



# sFlow Sampling Processes



- Two different sampling processes in sFlow
- Counters Sampling
  - Produce Counter Samples
- Statistical Packets Sampling
  - Produce Flow Samples





# sFlow Counters Sampling [1/3]

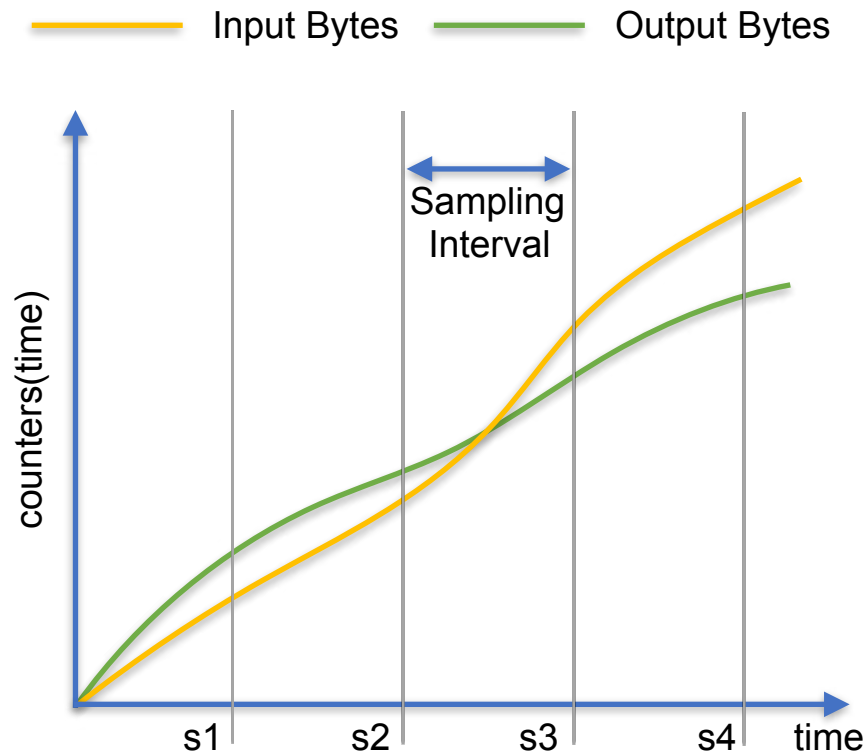


- Produce counter values for the Counter Samples
- Periodic sampling of network interfaces counters (e.g, input and output bytes and packets)
- sFlow agents are configured with a Sampling Interval
  - One sample every Sampling Interval





# sFlow Counters Sampling [2/3]



- $\Delta = \text{Sampling Interval}$
- $sX = X\text{th counter sample}$
- $s1 = \text{counters}(\Delta)$
- $s2 = \text{counters}(2\Delta)$
- ...
- $sN = \text{counters}(N\Delta)$



# sFlow Counters Sampling [3/3]



- Sampling Interval is intended to be the maximum time between two consecutive counter samples
- Counter samples may be taken opportunistically to "pad" other sFlow datagrams



# sFlow Packets Sampling



- Produce packets for the Flow Samples
- Must ensure that any packet observed has an equal chance of being sampled
- Sampling rate is configurable





# Sampling Accuracy



- Sampling, although unable to offer 100% exact results, are able to provide results with a statistically-quantifiable accuracy



# An Example of Packets Sampling: HTTP



- 1,000,000 packets transit the network
- 10,000 packets are sampled at random (1%)
- 1,000 of the samples represent HTTP traffic





# Estimating the Actual Number of HTTP Packets



- If 1,000 of the samples represent HTTP traffic, then how many of the original 1M packets were actually HTTP?
  - At least 1,000 (those that have been sampled)
  - At most 991,000 (990,000 unsampled + 1,000 HTTP samples)
  - ... but neither of these two values is at all likely...





# Best Estimate of the Actual Number of HTTP Packets



- It is most likely that the fraction of HTTP traffic is in the same ratio as its fraction of the samples
- 1,000 of the 10,000 samples, i.e., 10%
- This gives a value of 100,000 packets as the best estimate of the total number of HTTP packets





# How Confident We can Be?



- Of course it is very unlikely that there were exactly 100,000 HTTP packets
- A small range of values can be specified that are very likely, say 95% likely, to contain the actual value





# Calculating the Confidence



- Calculating the confidence boils down to estimating the variance of the best estimate (closed-form solution exists)
- We are 95% confident that the actual number of HTTP packets falls somewhere between 94,120 and 105,880





# Confidence as a % [1/2]

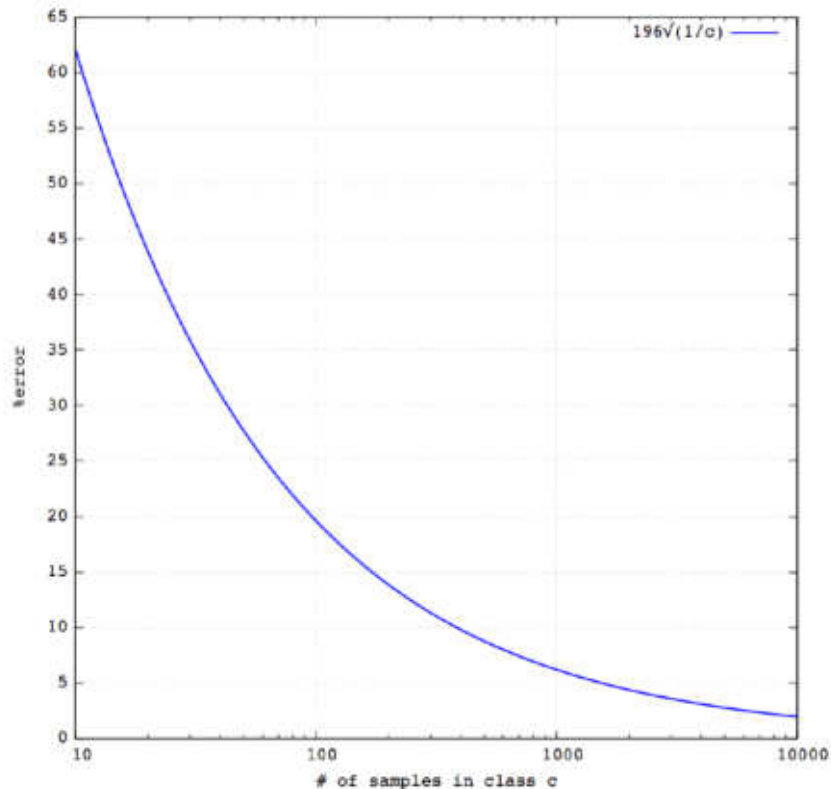


- The confidence range calculated can also be expressed as a percentage of the best estimate
- One can say that the actual value is, with high probability, within a %error from the best estimate
- In other words the largest likely error is %error





# Confidence as a % [2/2]



- Depends only on the number of samples  $c$
- Independent from the total number of frames
- Same confidence:
  - 1,000 Pps sampling rate of 1%
  - 1,000,000 Pps sampling rate of 0,001%



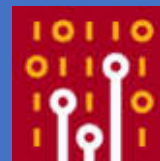
# sFlow vs Other Technologies



- Several other technologies have been developed over the years to provide network-wide visibility
  - Cisco NetFlow (v1, v5, v7, v8, v9)
  - IPFIX
  - SNMP (v1, v2c, v2c, v3)
  - RMON



# sFlow vs SNMP [1/2]






- SNMP provides what sFlow provides with counter samples but...
- ... there is no concept of flow samples in SNMP
- With SNMP you can tell how much bandwidth is being used but...
- ... you cannot tell who is using the bandwidth





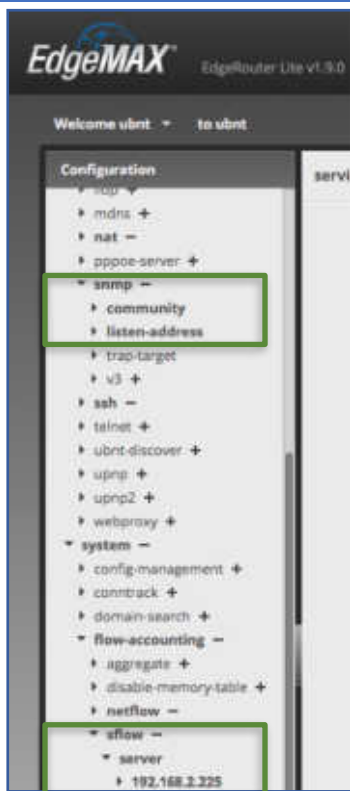
# sFlow vs SNMP [2/2]



	sFlow	SNMP
Transport	UDP	UDP
Architecture	↑ PUSH	↓ PULL
Device Visibility		
Traffic Visibility		



# sFlow vs SNMP Traffic



- Ubiquiti EdgeRouter Lite
- Configured with
  - sFlow
  - SNMP
- Assess the traffic required to have counters for one interface





# sFlow vs SNMP: sFlow Traffic



One sample per packet

Counter samples

Only for interface with id 3

186-Byte packets

ubiquity\_sFlow pcap

Expression... +

`((sflow_245.numsamples == 1) && (sflow_245.sampletype == 2) && (sflow_245.ifindex == 3))`

No.	Time	Source	Destination	Protocol	Length	NumSamples	Info
36	6.820105	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
93	18.530577	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
199	37.440200	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
241	48.810444	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
340	69.604585	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
473	97.650854	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
516	105.068770	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
572	117.429964	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
615	125.425581	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
679	137.892357	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
722	147.080032	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
821	167.928194	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
878	178.731487	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...



# sFlow vs SNMP: SNMP Traffic



```
$ snmpget -v2c -cntop 192.168.2.1 ifHCInOctets.3
IF-MIB::ifHCInOctets.3 = Counter64: 57111598398
$ snmpget -v2c -cntop 192.168.2.1 ifHCOutOctets.3
IF-MIB::ifHCOutOctets.3 = Counter64: 1310307062699
$ snmpget -v2c -cntop 192.168.2.1 ifHCInUcastPkts.3
IF-MIB::ifHCInUcastPkts.3 = Counter64: 510083567
$ snmpget -v2c -cntop 192.168.2.1 ifHCOutUcastPkts.3
IF-MIB::ifHCOutUcastPkts.3 = Counter64: 921959741
$ snmpget -v2c -cntop 192.168.2.1 ifHighSpeed.3
IF-MIB::ifHighSpeed.3 = Gauge32: 1000
```

781  
Bytes

No.	Time	Source	Destination	Protocol	Length	NumSamples	Info
1	0.000000	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.6.3
2	0.036348	192.168.2.1	172.16.2.141	SNMP	81		get-response 1.3.6.1.2.1.31.1.1.1.6.3
3	2.238210	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.10.3
4	2.274659	192.168.2.1	172.16.2.141	SNMP	82		get-response 1.3.6.1.2.1.31.1.1.1.10.3
5	10.530200	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.7.3
6	10.601663	192.168.2.1	172.16.2.141	SNMP	80		get-response 1.3.6.1.2.1.31.1.1.1.7.3
7	17.594939	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.11.3
8	17.656787	192.168.2.1	172.16.2.141	SNMP	80		get-response 1.3.6.1.2.1.31.1.1.1.11.3
9	30.435430	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.15.3
10	30.470999	192.168.2.1	172.16.2.141	SNMP	78		get-response 1.3.6.1.2.1.31.1.1.1.15.3



# sFlow vs SNMP: sFlow Traffic



sFlow: 1 186-Byte packet  
SNMP: 10 packets  
with total length  
781 Bytes

The image displays two screenshots from the Wireshark network traffic analysis tool. The top screenshot shows a capture of sFlow traffic from 'ubiquity\_sFlow.pcap' with a display filter '(sflow\_245.numsamples == 1) && (sflow\_245.samplotype == 2) && (sflow\_245.ifindex == 3)'. The table below shows several sFlow packets, with the 'Length' column highlighted in red for one of them.

No.	Time	Source	Destination	Protocol	Length	NumSamples	Info
36	6.820105	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub...
93	18.530577	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub...
199	37.440200	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub...
241	48.810444	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...
340	69.604585	192.168.2.1	192.168.2.225	sFlow	186	1	V5, agent 192.168.80.149, sub-agen...

The bottom screenshot shows a capture of SNMP traffic from 'SNMP\_snmpget.pcap'. The table below shows 10 SNMP packets, with the 'Length' column highlighted in red for each row.

No.	Time	Source	Destination	Protocol	Length	NumSamples	Info
821	1 0.000000	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.6.3
822	2 0.036348	192.168.2.1	172.16.2.141	SNMP	81		get-response 1.3.6.1.2.1.31.1.1.1.6.3
823	3 2.238210	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.10.3
824	4 2.274659	192.168.2.1	172.16.2.141	SNMP	82		get-response 1.3.6.1.2.1.31.1.1.1.10.3
825	5 10.530200	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.7.3
826	6 10.601663	192.168.2.1	172.16.2.141	SNMP	80		get-response 1.3.6.1.2.1.31.1.1.1.7.3
827	7 17.594939	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.11.3
828	8 17.656787	192.168.2.1	172.16.2.141	SNMP	80		get-response 1.3.6.1.2.1.31.1.1.1.11.3
829	9 30.435430	172.16.2.141	192.168.2.1	SNMP	76		get-request 1.3.6.1.2.1.31.1.1.1.15.3
830	10 30.470999	192.168.2.1	172.16.2.141	SNMP	78		get-response 1.3.6.1.2.1.31.1.1.1.15.3



# sFlow, Wireshark and ntop



- Wireshark can be used with sFlow traffic to
  - Dissect sFlow packets
  - Dissect packets in sFlow flow samples
- Using the a Lua plugin by ntop Wireshark can be used also as an sFlow collector



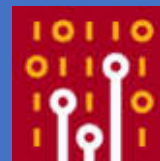
# DEMOS



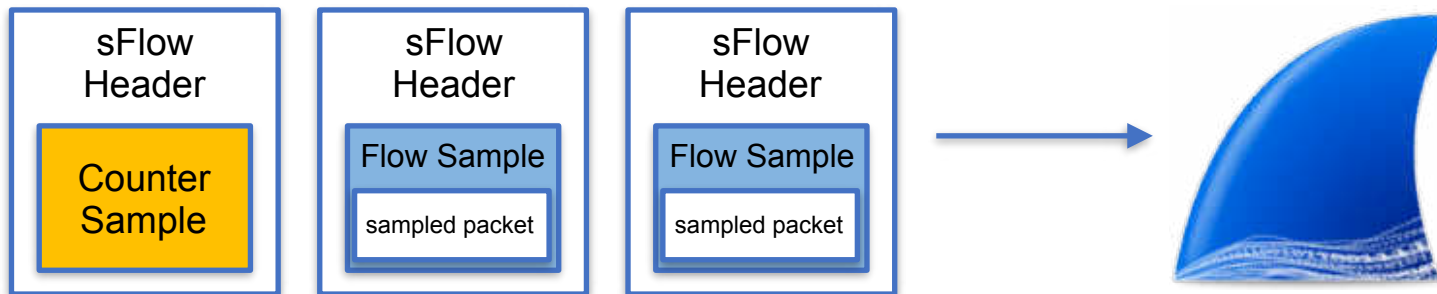
- Live sFlow traffic courtesy of our friend Jens Olsson at hosting provider Inleed
- Three switches generating sFlow that we will get via SSH



# DEMO #1: Wireshark + sFlow Traffic [1/2]



- A closer look at sFlow traffic with Wireshark







# DEMO #1: Wireshark + sFlow Traffic [2/2]



Execute a remote tcpdump via SSH  
(could have used Wireshark extcap sshdump)

output to stdout

filter to just get  
sFlow traffic

```
ssh root@<remote-host> "tcpdump -s0 -nnei ens3 -w - 'port 6343'" \  
| wireshark -k -i -
```

pipe the ssh stdout...

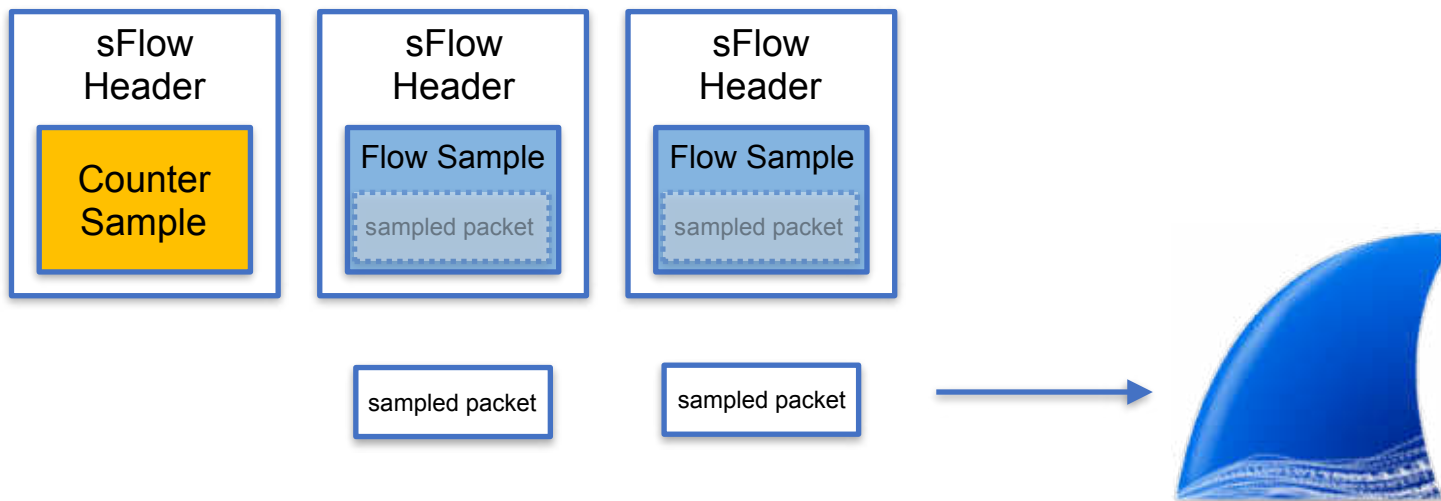
...to the Wireshark stdin



# DEMO #2: Wireshark + sFlow Sampled Packets [1/2]



- sflowtool required to extract packets  
<https://github.com/sflow/sflowtool.git>





# DEMO #2: Wireshark + sFlow Sampled Packets [2/2]



Execute a remote tcpdump via SSH

output to stdout

filter to just get  
sFlow traffic

```
ssh root@<remote-host> "tcpdump -s0 -nnei ens3 -w - 'port 6343' \" \  
| ./src/sflowtool -t -r - \  
| wireshark -k -i -"
```

pipe the sflowtool stdout...

...to the Wireshark stdin

read from stdin...

... and output packets  
contained in flow samples  
to stdout

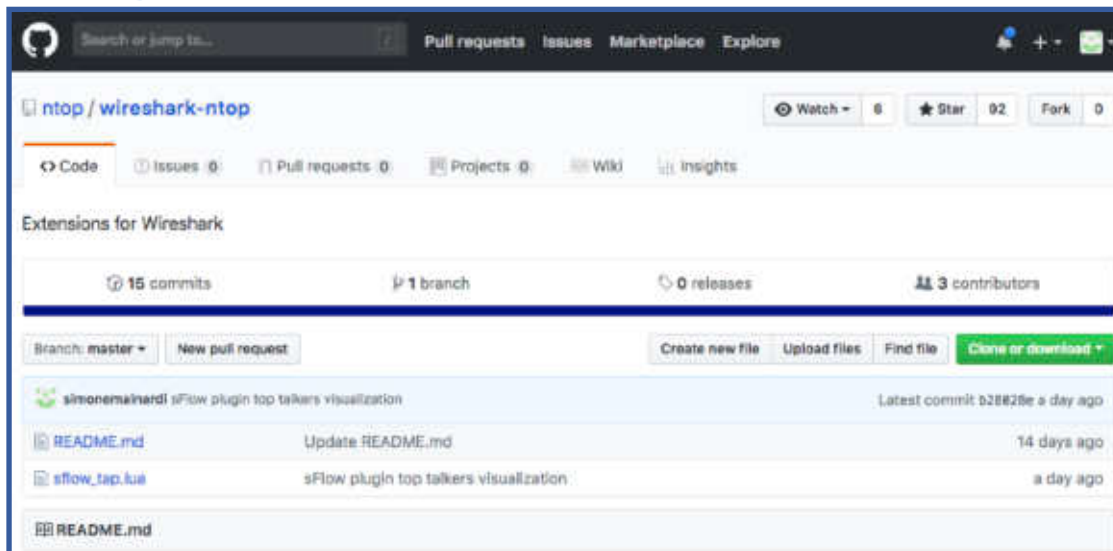


# DEMO #3: Wireshark as an sFlow Collector [1/2]



Fork me on GitHub

- Lua plugin `sflow_tap.lua` is available at <https://github.com/ntop/wireshark-ntop>





# DEMO #3: Wireshark as an sFlow Collector [2/2]



Wireshark - sFlow Counters

INTERFACE	IN BYTES	OUT BYTES	IN RATE	OUT RATE	UTILIZATION
1	8.25 GB	956.24 GB	0.00 B/s	69.80 Kb/s	0.00 %
2	5.61 GB	120.73 GB	151.03 Kb/s	4.89 Mb/s	0.49 %
25	9.11 TB	20.54 TB	97.41 Mb/s	215.22 Mb/s	21.52 %
26	21.51 TB	9.05 TB	207.72 Mb/s	86.93 Mb/s	2.08 %
TOTAL	30.64 TB	30.65 TB	305.28 Mb/s	307.10 Mb/s	

Wireshark - sFlow Top Talkers

SOURCE	SOURCE BYTES	SOURCE RATE	DEST	DEST BYTES	DEST RATE
agent: 10.0.2.154					
79.136.102.9	171.17 MB	1.36 Mb/s	192.165.9.17	191.19 MB	1.36 Mb/s
185.189.49.21	30.75 MB	424.55 Kb/s	185.189.49.20	5.74 MB	1.63 Mb/s
94.254.123.37	17.17 MB	278.42 Kb/s	79.136.102.9	3.43 MB	78.12 Kb/s
151.101.1.62	5.74 MB	1.63 Mb/s	107.167.113.42	2.86 MB	0.00 B/s
185.189.51.174	5.72 MB	274.79 Kb/s	81.229.134.60	2.86 MB	0.00 B/s
agent: 10.0.2.253					
94.254.123.37	20.03 MB	1.06 Mb/s	185.189.49.4	13.70 MB	3.15 Mb/s
213.80.97.15	13.70 MB	3.15 Mb/s	5.150.195.205	11.48 MB	22.05 Mb/s
192.168.0.162	12.04 MB	27.41 Kb/s	194.71.138.160	11.44 MB	223.92 Kb/s



# Take-Home



- sFlow is a pretty lightweight technology to have an overall view of your network devices and the traffic they are handling
  - Is this device overloaded? Who's consuming all this bandwidth?
- Wireshark is suitable not only to dissect and inspect sFlow packets but also to provide devices interfaces status and top talkers information!
  - `sflow_tap.lua` plugin available at: <https://github.com/ntop/wireshark-ntop>
- Contact me: [mainardi@ntop.org](mailto:mainardi@ntop.org) / [@simonemainardi](https://twitter.com/simonemainardi)



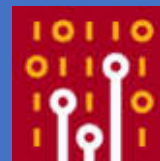
# Appendix



- Effects of lost sFlow packets
- Packet Sampling:
  - Strategies
  - Formulas
  - Statistical Background
- Demonstration screenshots



# Effects of Lost sFlow Packets



- Lost counter samples
  - Values are cumulative, new (updated) values will be sent in the next sample
  - Almost impossible to miss the detection of a counter wrap (64-bit counters)
- Lost flow samples
  - Changes in the actual sampling rate

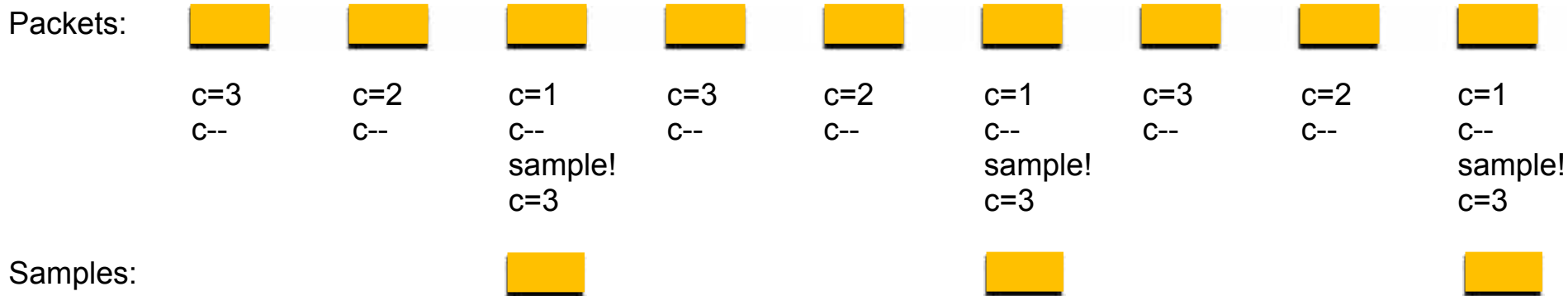




# Packets Sampling Strategies [1/2]



- One packet in N is sampled
  - Initialize a counter to N
  - Decrement the counter with each packet
  - Sample the packet when the counter reaches 0
- Example with N=3

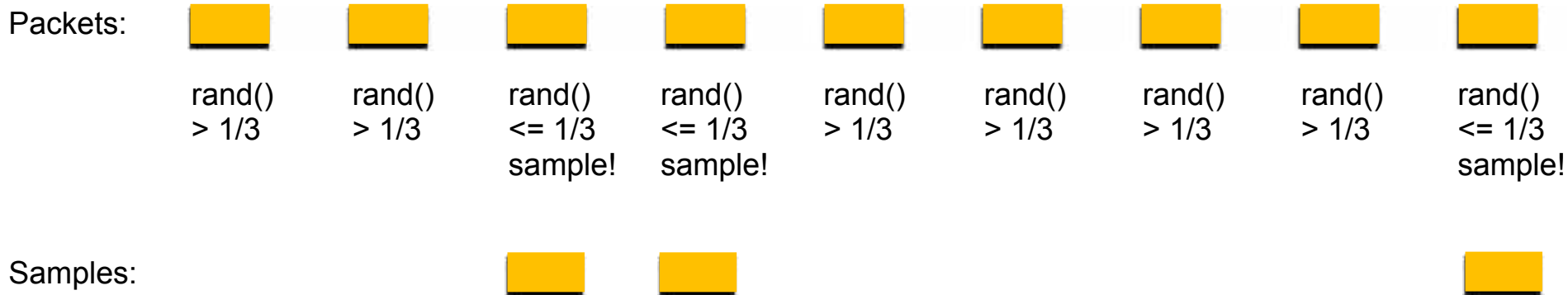




# sFlow Packets Sampling [2/2]



- One packet in  $N$  (on average) is sampled
  - Draw a random number  $0 \leq r \leq 1$
  - Sample if  $r \leq 1/N$
- Synchronization with periodic traffic patterns is prevented with randomness
- Example with  $N=3$ ,  $\text{rand}() = \text{random } [0,1]$  number generator





# Estimating the Actual Number of HTTP Packets



- If 1,000 of the samples represent HTTP traffic, then how many of the original 1M packets were actually HTTP?
  - At least 1,000 (those that have been sampled)
  - At most 991,000 (990,000 unsampled + 1,000 HTTP samples)
  - ... but neither of these two values is at all likely...





# Best Estimate of the Actual Number of HTTP Packets



- It is most likely that the fraction of HTTP traffic is in the same ratio as its fraction of the samples
- 1,000 of the 10,000 samples, i.e., 10%
- This gives a value of 100,000 packets as the best estimate of the total number of HTTP packets





# How Confident We can Be?



- Of course it is very unlikely that there were exactly 100,000 HTTP packets
- A small range of values can be specified that are very likely, say 95% likely, to contain the actual value





# Calculating the Confidence [1/3]



- Calculating the confidence boils down to estimating the variance of the best estimate



# Calculating the Confidence [2/3]



- $N = 1,000,000$  packets transited
- $n = 10,000$  packets sampled
- $c = 1,000$  HTTP samples
- $N_c = 100,000 = \text{best estimate} = c / n * N$
- The variance of the best estimate  $N_c$  is
$$\sigma^2 = N^2 * c * (1 - c / n) * 1 / (n * (n - 1))$$
$$= 9,000,000$$



# Calculating the Confidence [3/3]



- The 95% confidence is within 1.96 standard deviations from the best estimate  
[ $N_c - 1.96\sigma$ ;  $N_c + 1.96\sigma$ ]
- In the HTTP example
  - $\sigma^2 = 9,000,000$
  - $\sigma = 3,000$
  - [100,000 - 1.96 \* 3000, 100,000 + 1.96 \* 3000]  
= [94,120; 105,880]
- We are 95% confident that the actual number of HTTP packets falls somewhere between 94,120 and 105,880







# Confidence as a % [1/3]



- The confidence range calculated can also be expressed as a percentage of the best estimate
- One can say that the actual value is, with high probability, within a %error from the best estimate
- In other words the largest likely error is %error



# Confidence as a % [2/3]



- The estimate of the percentage error %error  
 $\%error = \sqrt{1 / c}$
- In the HTTP example
  - $\%error = 196 * \sqrt{1 / c}$   
 $= 196 * \sqrt{1 / 1,000}$   
 $= 6.20 \%$
- The largest likely error is 6.20 %
- Note: %error formula given is an approximation and only works well when  $n \gg c$





# Statistical Background



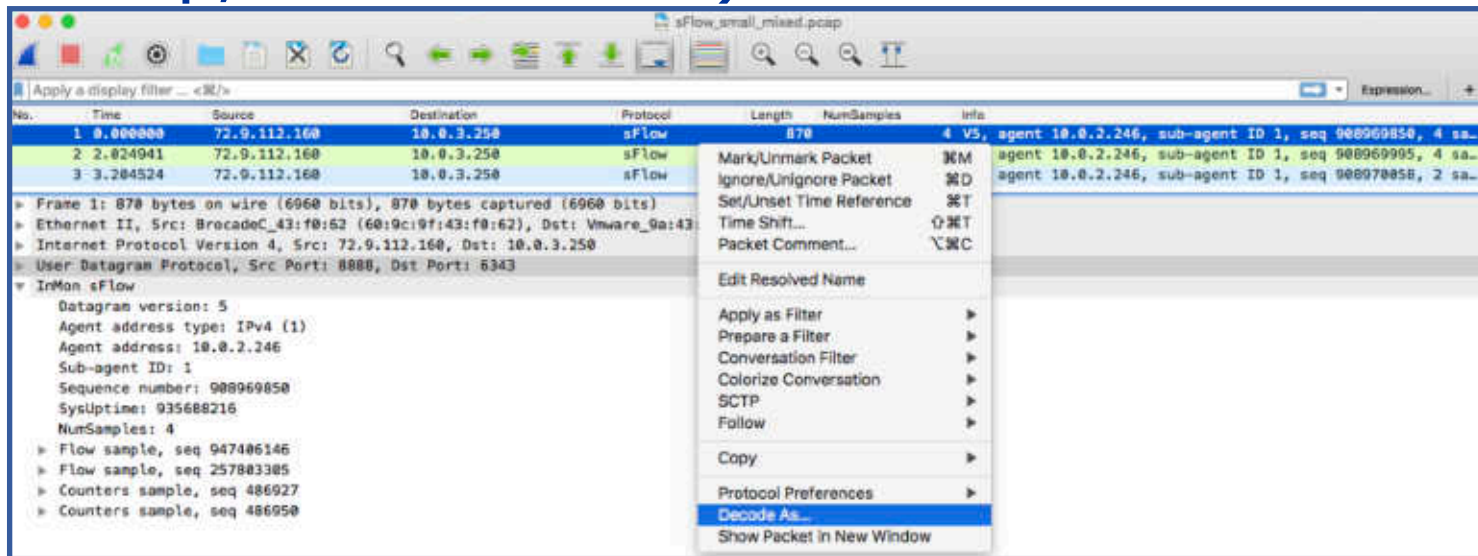
- Assumption is that packet sampling can be modeled by the binomial distribution
- Prove that measured statistics can be used to accurately estimate the parameters of the actual theoretical binomial distribution
- Use the central limit theorem to compute the confidence intervals of a normal curve



# DEMO: Wireshark + sFlow Traffic



- Simply feed Wireshark with sFlow traffic (pcap, extcap, live interfaces)





# DEMO: Wireshark + sFlow Flow Samples



The image shows a Wireshark interface with a packet capture from interface eth0. The display filter is set to 'eth0'. The packet list pane shows several packets, with packet 389 highlighted. The packet details pane shows the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and Data (20 bytes).

No.	Time	Source	Destination	Protocol	Length	NumSamps	Info
384	40.000000	Cisco_36:8f:3f	Broadcast	ARP	64		who has 105.109.49.230? Tell 105.109.48.1
385	40.000000	194.68.59.32	60.226.18.20	OpenVPN	746		MessageType: P_DATA_V2
386	40.000000	194.68.59.34	90.152.134.226	OpenVPN	190		MessageType: P_DATA_V1
387	40.000000	194.68.59.76	84.120.203.209	OpenVPN	1415		MessageType: P_DATA_V1
388	41.000000	76.76.231.9	46.50.182.198	TCP	70		68956 → 443 [ACK] Seq=1 Ack=1 Win=1988 Len=0 TSval=8718...
389	41.000000	194.68.59.76	2.21.240.211	TCP	60		[TCP Previous segment not captured] 52727 → 443 [ACK] S...
390	41.000000	194.68.59.30	89.153.165.148	OpenVPN	757		MessageType: P_DATA_V2
391	41.000000	194.68.59.30	80.153.165.148	OpenVPN	757		MessageType: P_DATA_V2
392	41.000000	194.68.59.36	134.3.254.70	OpenVPN	878		MessageType: P_DATA_V2
393	41.000000	194.68.59.76	80.209.209.84	ESP	1562		ESP (SPI=0x2974ccb0)
394	41.000000	194.68.59.76	189.82.204.142	TCP	60		[TCP Previous segment not captured] 7773 → 443 [ACK] S...
395	41.000000	194.68.59.80	188.187.146.175	SSL	1506		[TCP Previous segment not captured] , Continuation Data
396	41.000000	194.68.59.56	31.16.258.172	OpenVPN	790		MessageType: P_DATA_V2
397	41.000000	194.68.59.80	188.187.146.175	SSL	1596		[TCP Previous segment not captured] , Continuation Data
398	42.000000	194.68.59.56	31.16.258.172	OpenVPN	746		MessageType: P_DATA_V2
399	42.000000	194.68.59.36	134.3.254.70	OpenVPN	878		MessageType: P_DATA_V2

Frame 358: 82 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface eth0  
Ethernet II, Src: IntelCor\_d0:23:3a (00:1b:21:0d:23:3a), Dst: Cisco\_36:8f:3f (f8:00:c0:36:8f:3f)  
Internet Protocol Version 4, Src: 194.68.59.56, Dst: 93.7.116.34  
User Datagram Protocol, Src Port: 5074, Dst Port: 22222  
Data (20 bytes)



# DEMO: Wireshark as an sFlow Collector [1/2]



The screenshot shows the Wireshark interface with the 'Tools' menu open and 'sFlow' selected. The 'Counters' sub-menu is also open, showing a list of 'Talkers'. The main window displays the 'sFlow Counters' window with three tables of statistics for different agents.

agent: 10.0.2.154	INTERFACE	IN BYTES	OUT BYTES	IN RATE	OUT RATE	UTILIZATION
	1	8.25 GB	956.24 GB	0.00 B/s	69.80 Kb/s	0.00 %
	2	5.61 GB	120.73 GB	151.03 Kb/s	4.89 Mb/s	0.49 %
	25	9.11 TB	20.54 TB	97.41 Mb/s	215.22 Mb/s	21.52 %
	26	21.51 TB	9.05 TB	207.72 Mb/s	86.93 Mb/s	2.08 %
	TOTAL	30.64 TB	30.65 TB	305.28 Mb/s	307.10 Mb/s	

agent: 10.0.2.253	INTERFACE	IN BYTES	OUT BYTES	IN RATE	OUT RATE	UTILIZATION
	3	250.62 MB	4.89 GB			
	4	23.27 GB	536.29 MB			
	6	66.67 GB	65.07 GB	301.51 Kb/s	64.24 Kb/s	0.03 %
	10	4.92 GB	109.29 GB	0.00 B/s	53.48 Kb/s	0.00 %
	14	220.22 GB	8.92 TB	15.98 Kb/s	98.10 Kb/s	0.00 %
	16	24.87 TB	43.79 TB	3.16 Mb/s	1.12 Mb/s	0.32 %
	17	40.61 TB	7.62 TB	8.90 Mb/s	3.42 Mb/s	0.89 %
	19	30.63 TB	5.46 TB	5.46 Mb/s	2.48 Mb/s	0.55 %
	20	105.59 TB	79.46 TB	51.32 Mb/s	7.29 Mb/s	5.13 %
	24	7.24 GB	418.61 GB	3.85 Kb/s	247.46 Kb/s	0.02 %
	25	215.70 TB	204.28 TB	263.15 Mb/s	197.08 Mb/s	2.63 %
	26	10.08 TB	78.23 TB	5.55 Mb/s	104.29 Mb/s	1.04 %
	TOTAL	427.78 TB	428.35 TB	337.86 Mb/s	316.15 Mb/s	

agent: 185.189.48.162	INTERFACE	IN BYTES	OUT BYTES	IN RATE	OUT RATE	UTILIZATION
	1	0.00 B	425.00 B			
	4	211.15 MB	37.61 GB	381.87 b/s	59.93 Kb/s	0.06 %
	5	693.78 KB	5.23 MB			
	6	19.29 KB	850.00 B			
	8	29.82 TB	29.64 TB	128.61 Mb/s	123.21 Mb/s	1.29 %
	TOTAL	29.82 TB	29.68 TB	128.61 Mb/s	123.27 Mb/s	



# DEMO: Wireshark as an sFlow Collector [2/2]



The screenshot shows the Wireshark interface with the 'Tools' menu open, navigating to 'ntop' > 'sFlow' > 'Counters' > 'Talkers'. The main window displays a table of sFlow Top Talkers data.

agent:	SOURCE	SOURCE BYTES	SOURCE RATE	DEST	DEST BYTES	DEST RATE
10.0.2.154	79.136.102.9	171.17 MB	1.36 Mb/s	192.165.9.17	191.19 MB	1.36 Mb/s
	185.189.49.21	30.75 MB	424.55 Kb/s	185.189.49.20	5.74 MB	1.63 Mb/s
	94.254.123.37	17.17 MB	278.42 Kb/s	79.136.102.9	3.43 MB	78.12 Kb/s
	151.101.1.62	5.74 MB	1.63 Mb/s	107.167.113.42	2.86 MB	0.00 B/s
	185.189.51.174	5.72 MB	274.79 Kb/s	81.229.134.60	2.86 MB	0.00 B/s
10.0.2.253	94.254.123.37	20.03 MB	1.06 Mb/s	185.189.49.4	13.70 MB	3.15 Mb/s
	213.80.97.15	13.70 MB	3.15 Mb/s	5.150.195.205	11.48 MB	22.05 Mb/s
	192.168.0.162	12.04 MB	27.41 Kb/s	194.71.138.160	11.44 MB	223.92 Kb/s
	194.68.59.159	11.44 MB	223.92 Kb/s	192.176.45.219	11.44 MB	1.06 Mb/s
	185.102.102.3	8.58 MB	24.00 Mb/s	192.176.45.237	11.41 MB	156.44 Kb/s
185.189.48.162	194.68.59.88	1.95 GB	1.57 Mb/s	46.223.129.1	1.89 GB	131.07 Kb/s
	194.68.59.56	759.24 MB	6.36 Mb/s	31.16.250.172	733.50 MB	6.36 Mb/s
	194.68.59.36	666.09 MB	7.08 Mb/s	134.3.254.70	625.27 MB	7.08 Mb/s
	194.68.59.54	567.68 MB	6.00 Mb/s	95.222.30.6	559.70 MB	6.00 Mb/s
	194.68.59.78	347.23 MB	12.19 Mb/s	80.209.209.84	340.56 MB	870.69 Kb/s