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

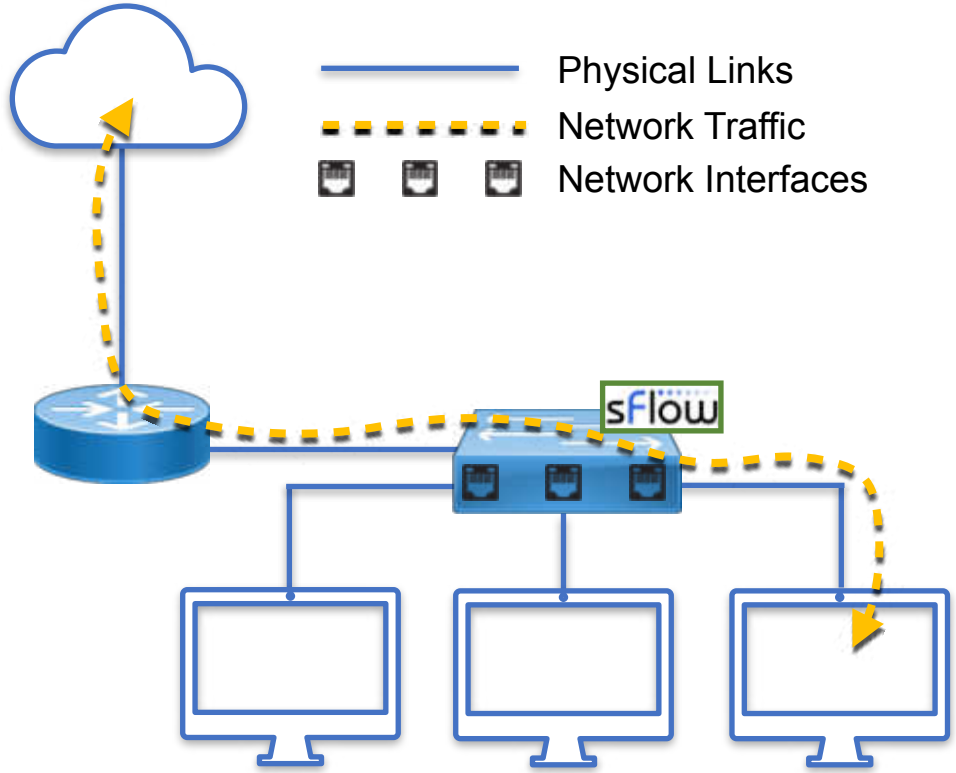
What is sFlow? [1/2]

- 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

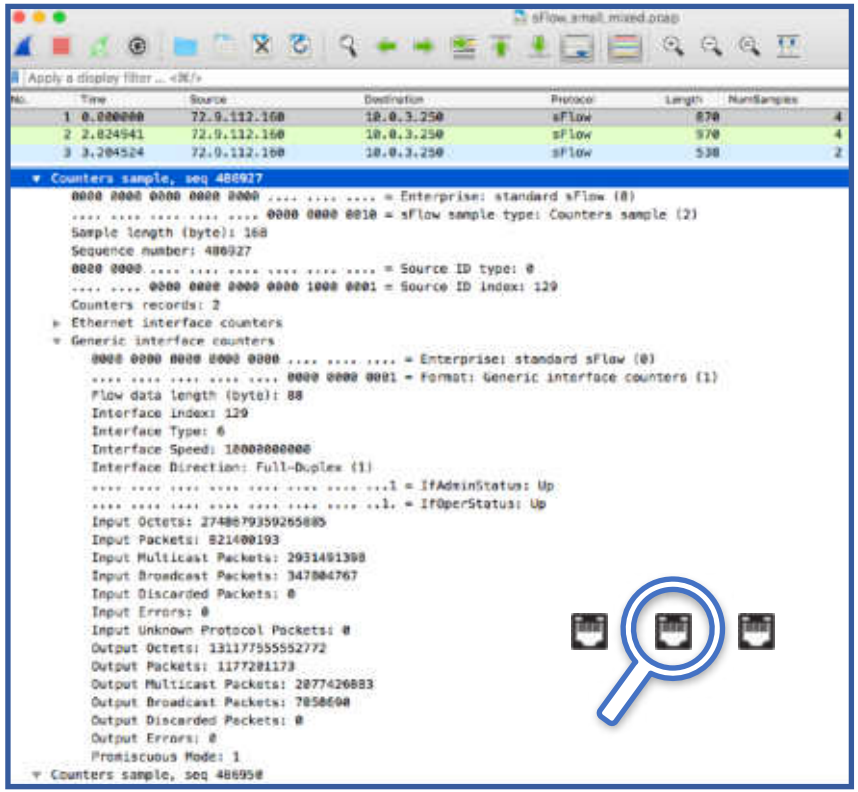
- Traffic visibility
  - Flow Samples

sFlow Counter Samples

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

Network Traffic

Network Interfaces

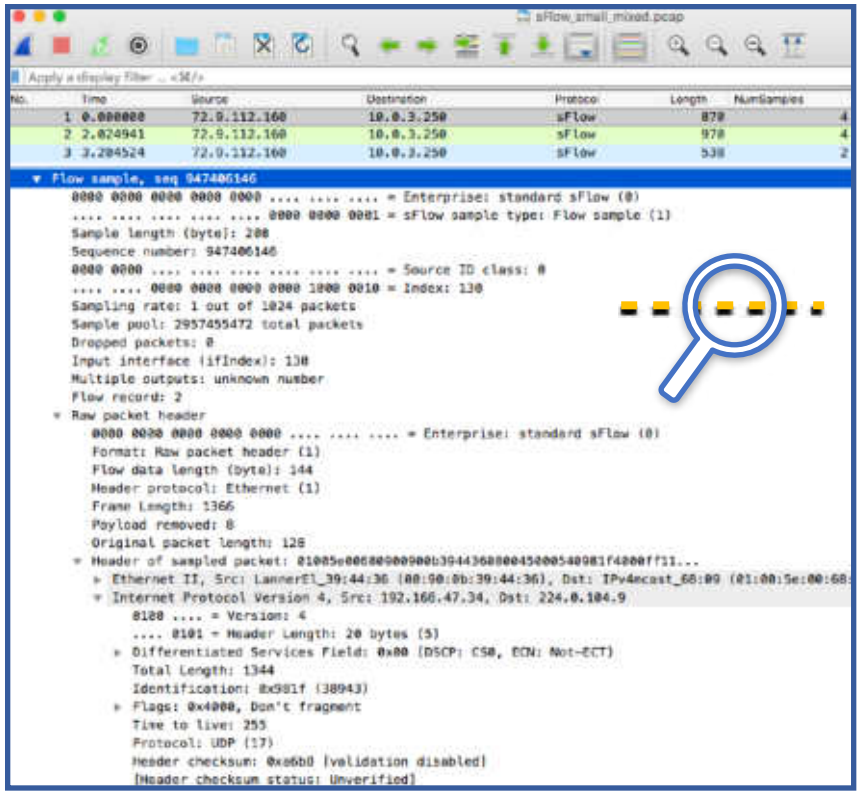


sFlow Flow Samples

- Random selection of a fraction of the packets observed

Network Traffic

Network Interfaces



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)

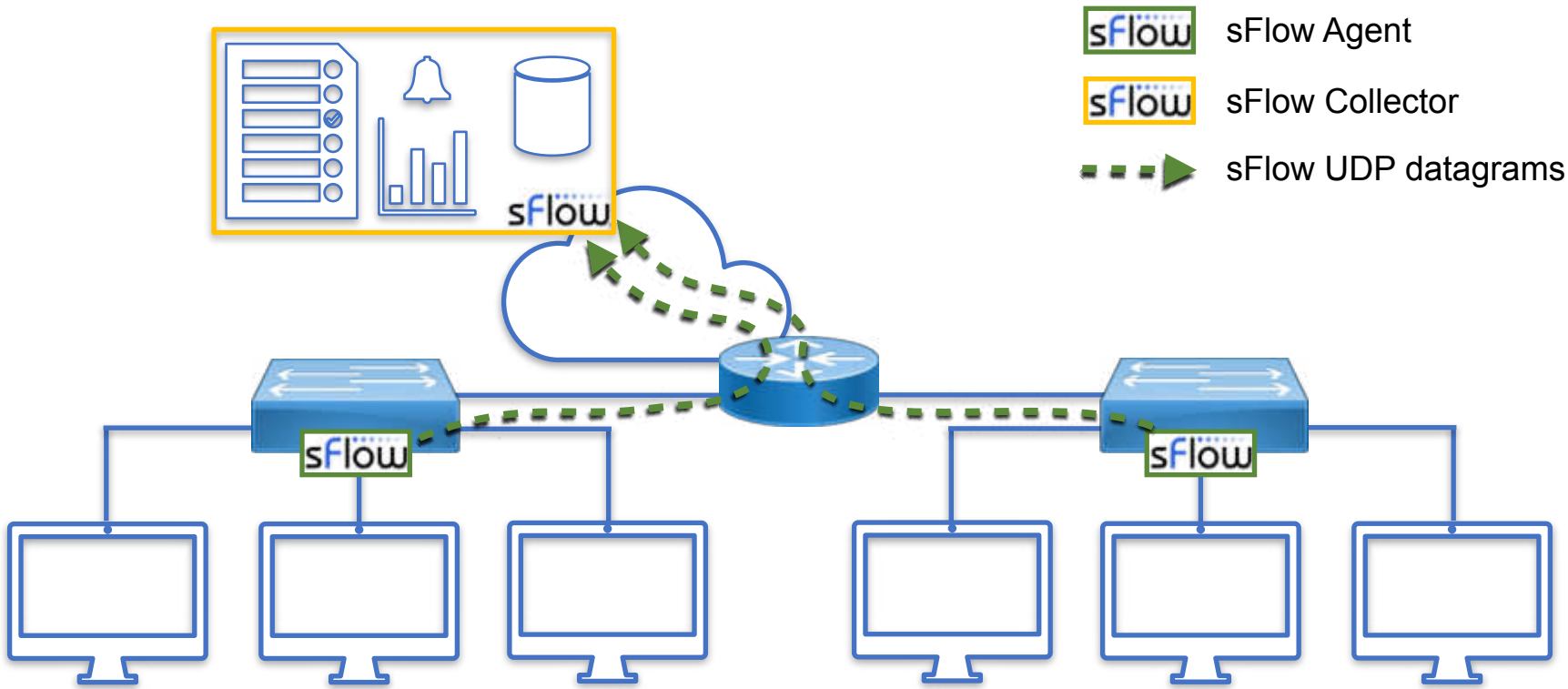
When is sFlow NOT Useful? [2/2]

- 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

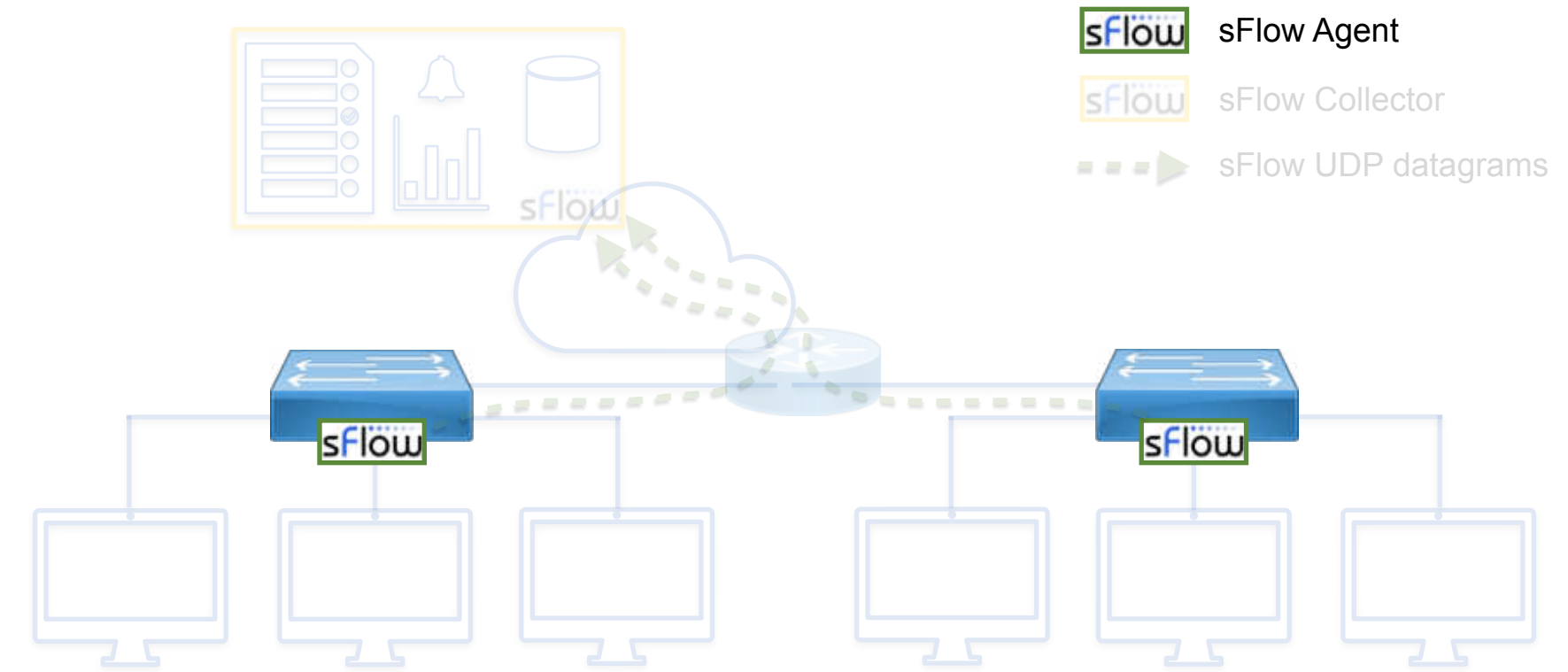
sFlow Monitoring Systems

- 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, AlexalA, 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

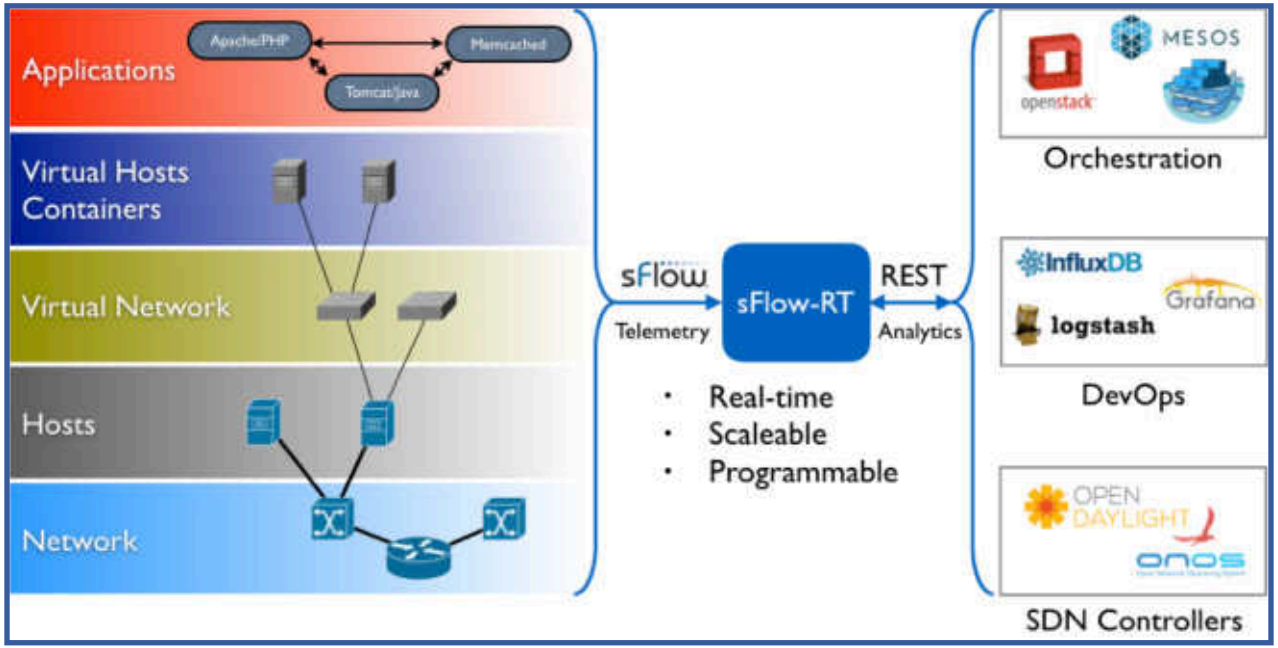
sFlow UDP datagrams

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, …

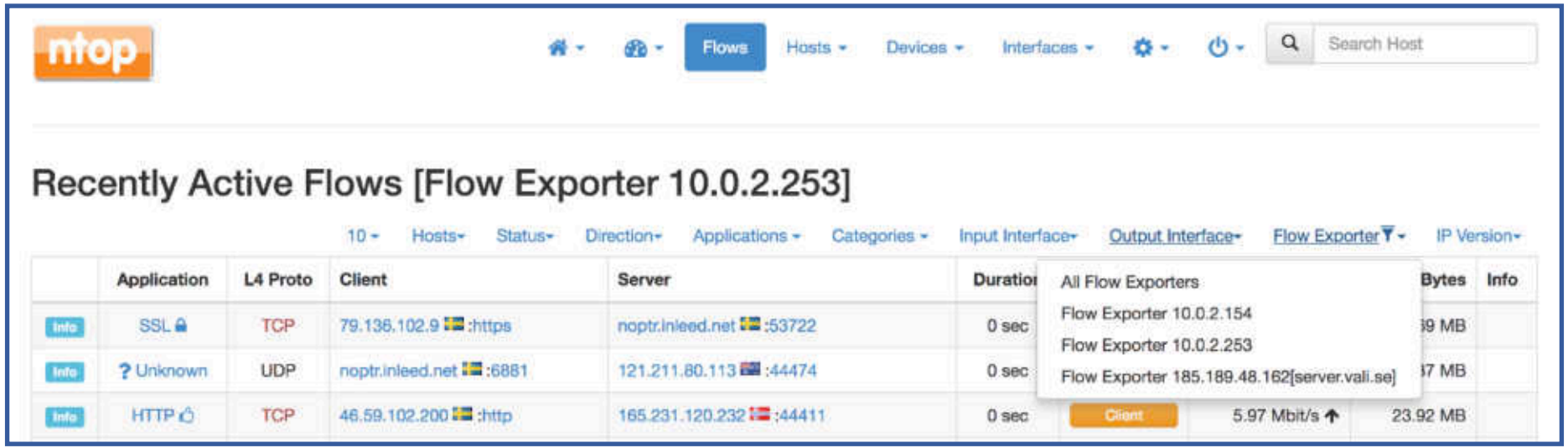
sFlow Collectors [2/4]

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



sFlow Collectors [3/4]

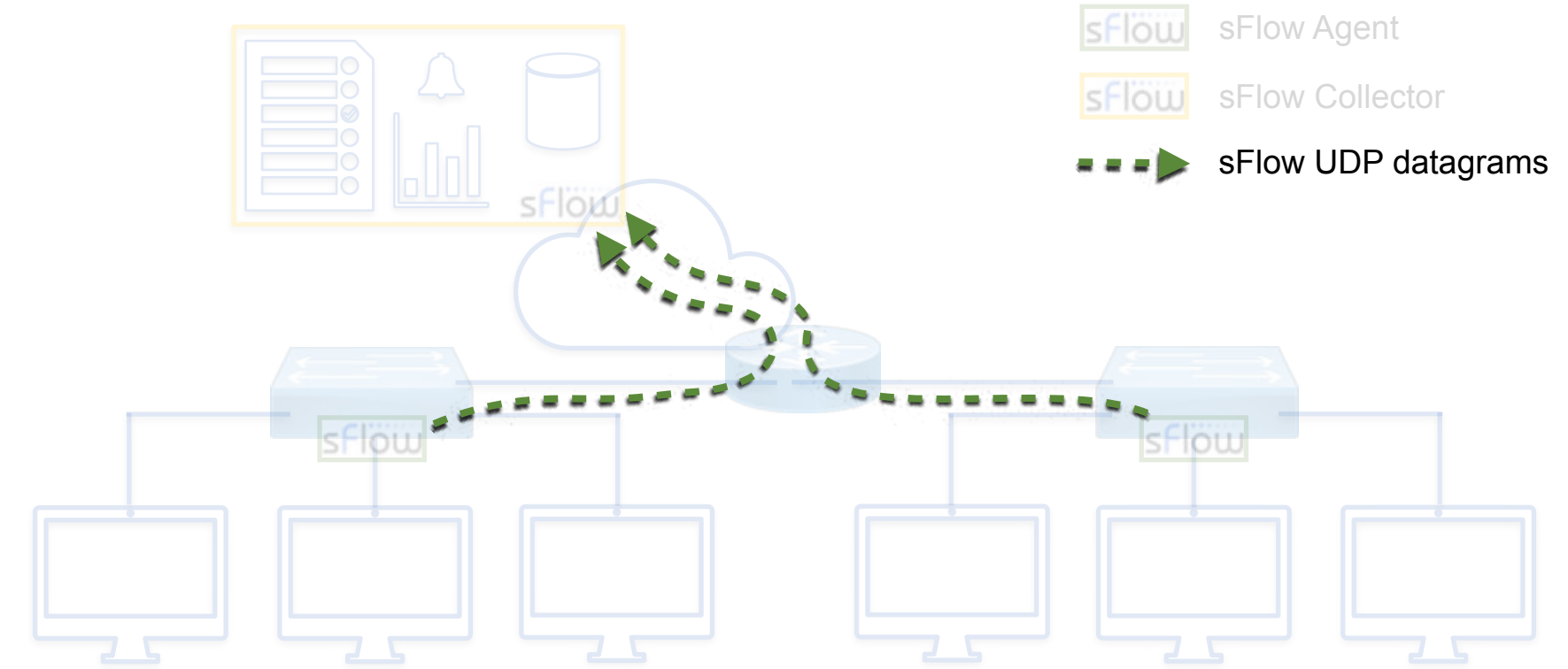
- ntopng
  - Graphical tool to generate live and historical statistics on sources and destinations, network conversations (who talks to whom), and network interfaces
  - Facilitates the correlation of sources and destinations with the physical ports they are using



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]

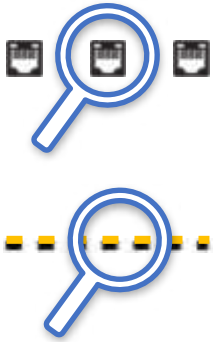
- sFlow UDP datagrams are periodically and unsolicitedly sent by each agent to one or more collectors
- Collectors don't need to discover new agents
- Reduced workload
  - Collectors don't have to generate reqs and match reqs/resps
  - Agents don't have to parse and process reqs

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

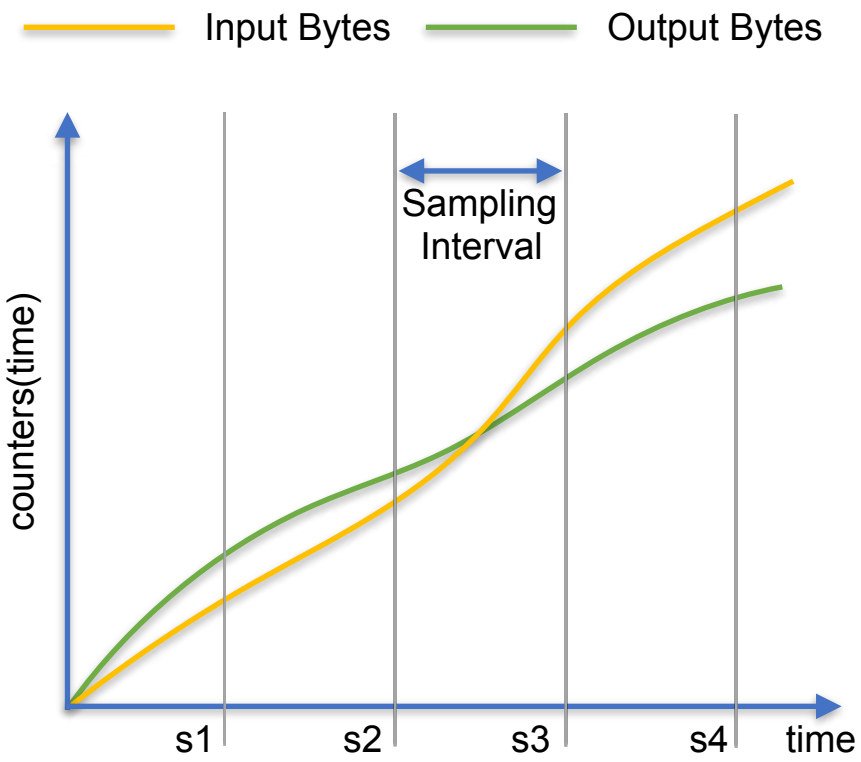
- Two different sampling processes in sFlow
- Counters Sampling
  - Produce Counter Samples
- Statistical Packets Sampling
  - Produce Flow Samples



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]



- Δ = Sampling Interval
- sX = Xth counter sample
- s1 = counters(Δ)
- s2 = counters(2Δ)
- …
- sN = counters(NΔ)

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

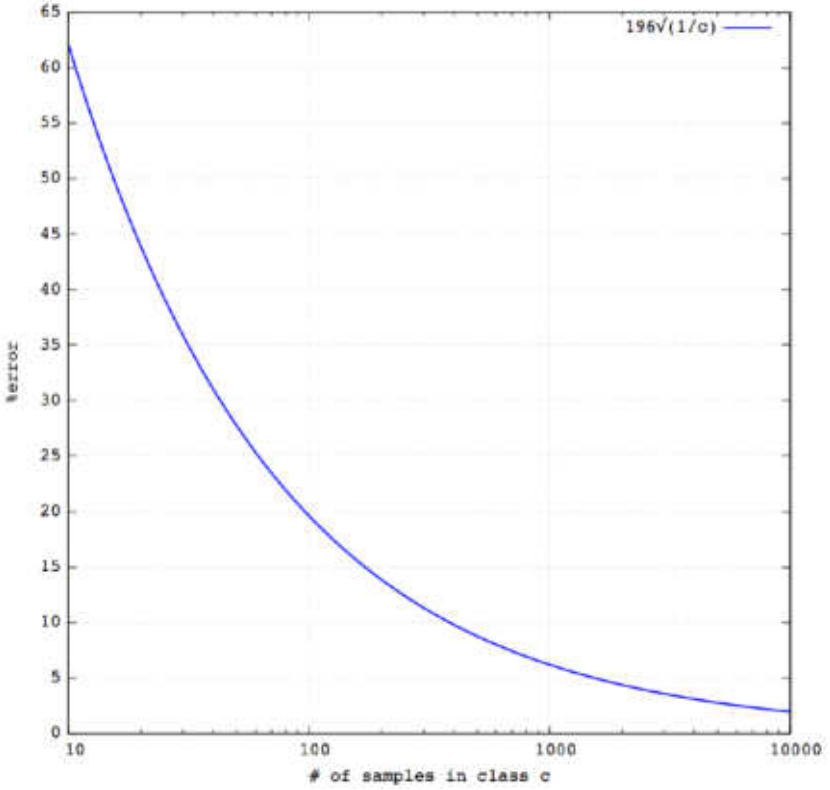
Calculating the Confidence

- 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

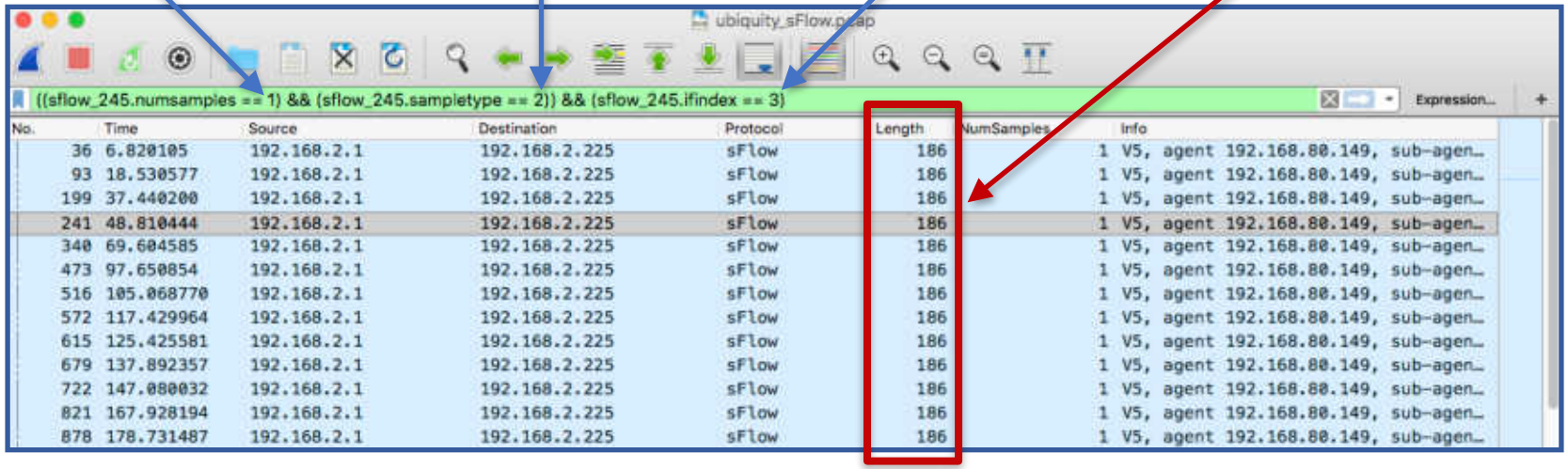
sFlow vs SNMP: sFlow Traffic

One sample per packet

Counter samples

Only for interface with id 3

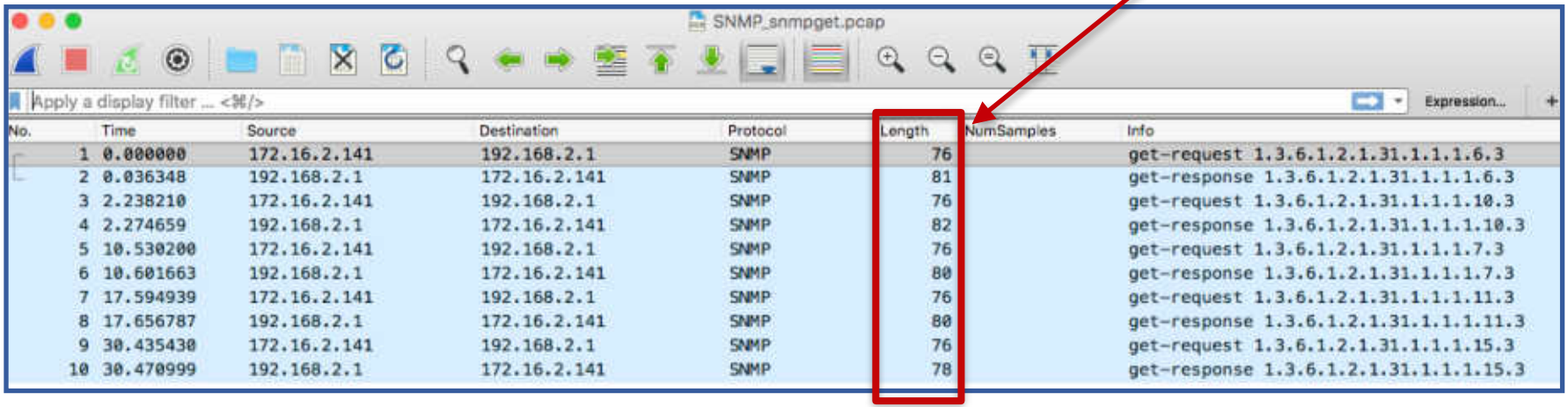
186-Byte packets



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
```

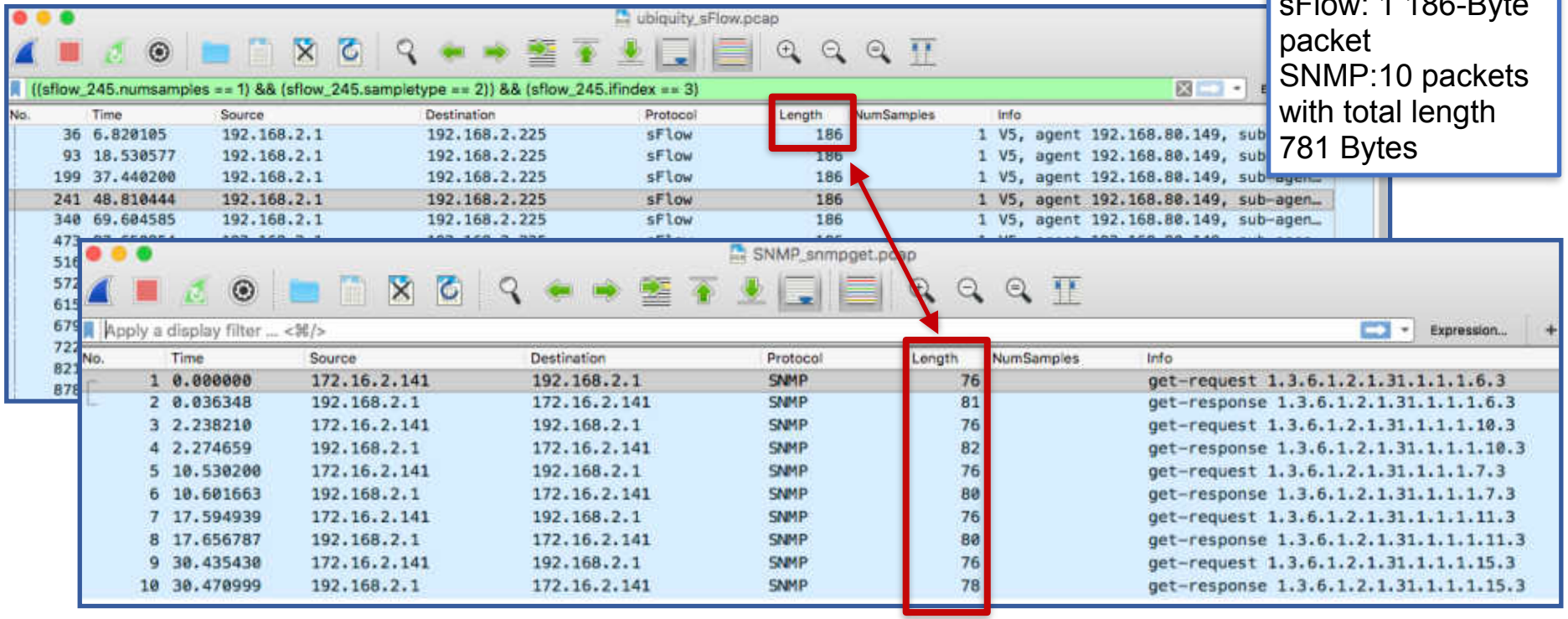
781 Bytes



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

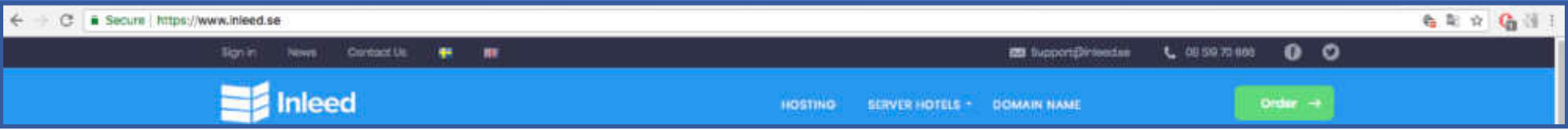
sFlow: 1 186-Byte packet
SNMP:10 packets with total length 781 Bytes



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

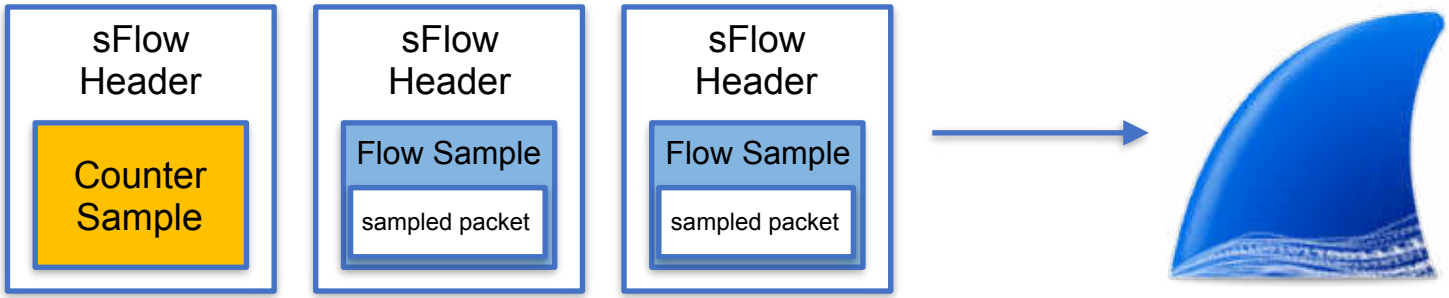
DEMOs



- 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

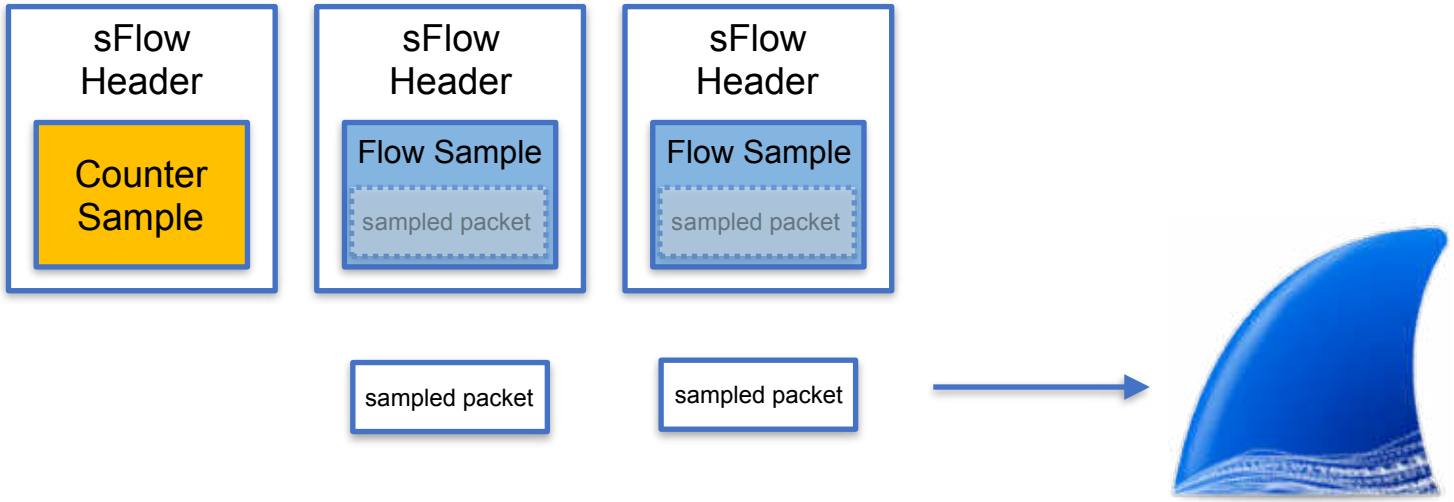
```
ssh root@<remote-host> "tcpdump -s0 -nnei ens3 -w - 'port 6343'" \
| wireshark -k -i -
```

pipe the ssh stdout…

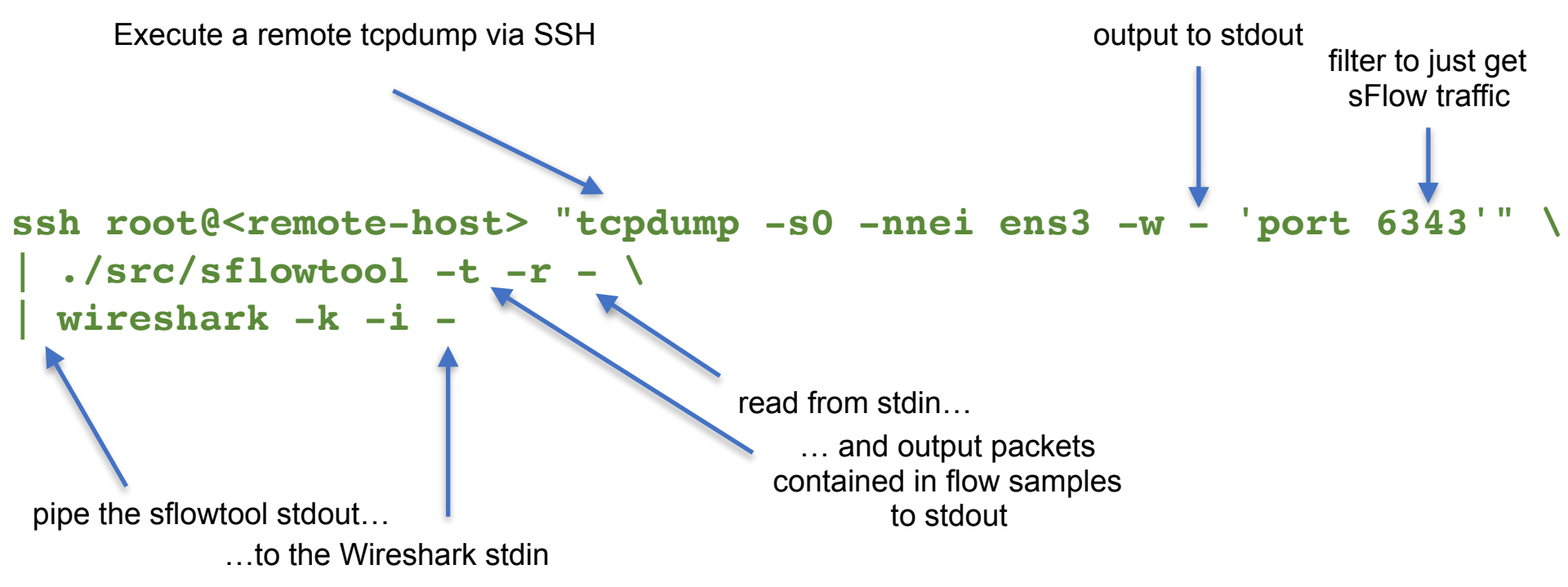
…to the Wireshark stdin

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

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

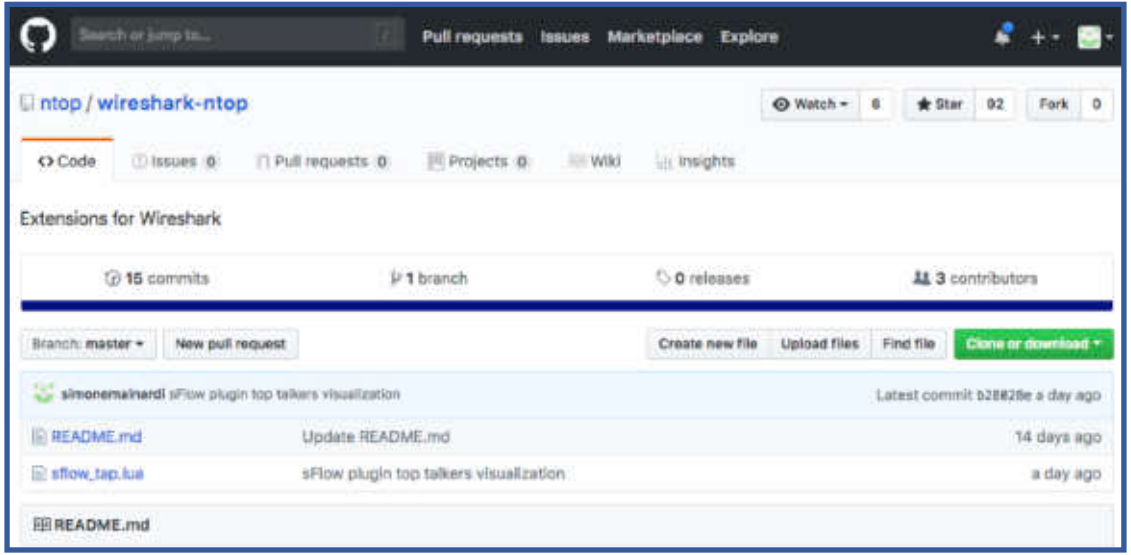


DEMO #2: Wireshark + sFlow Sampled Packets [2/2]

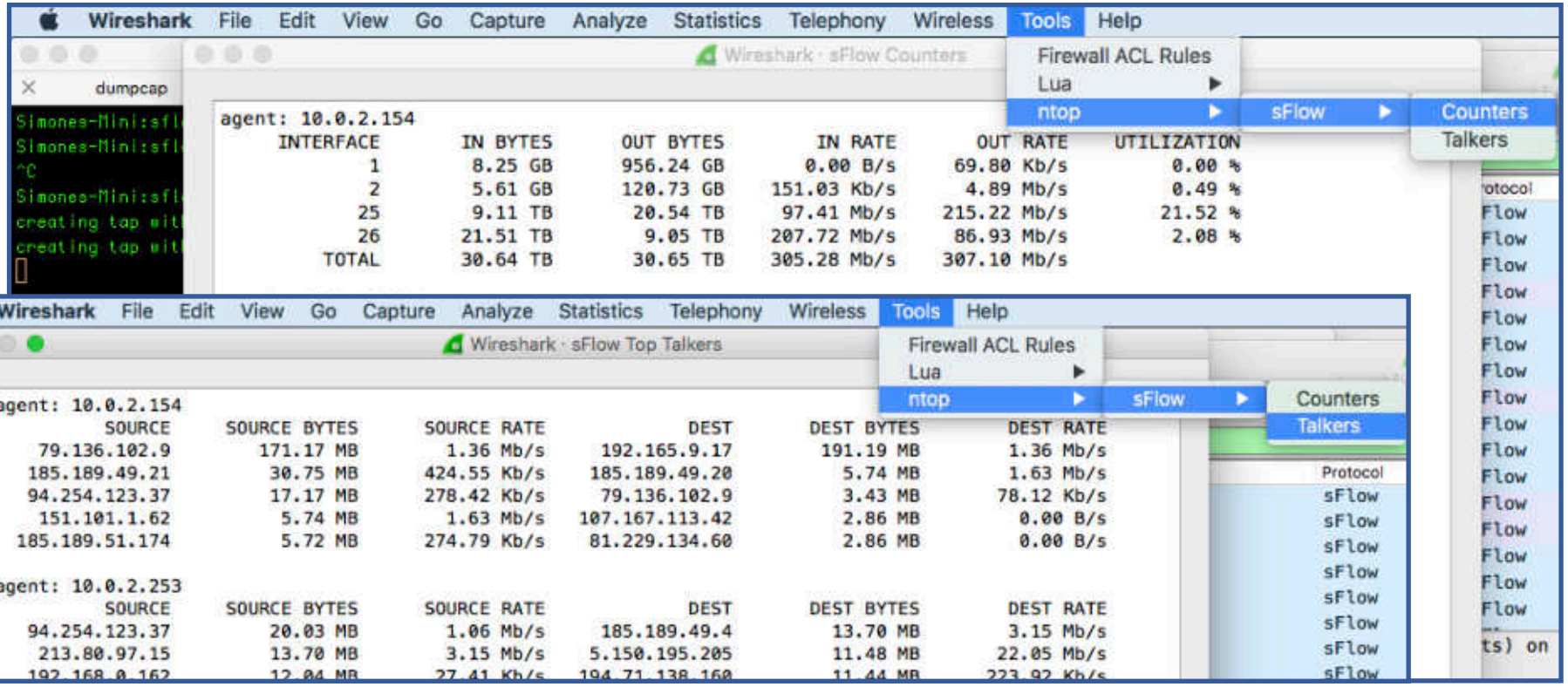


DEMO #3: Wireshark as an sFlow Collector [1/2]

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



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



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 / @simonemainardi

Appendix

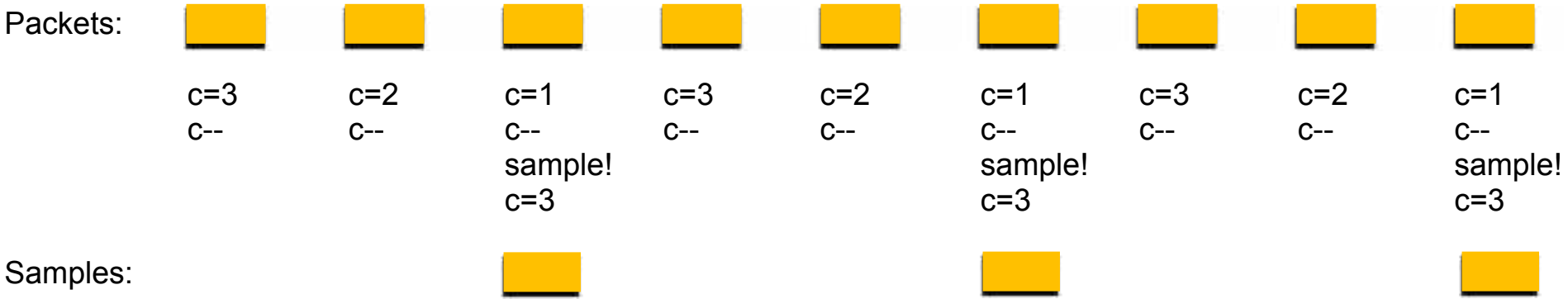
- Effects of lost sFlow packets
- Packet Sampling:
  - Strategies
  - Formulas
  - Statistical Background
- Demonstration screenshots

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

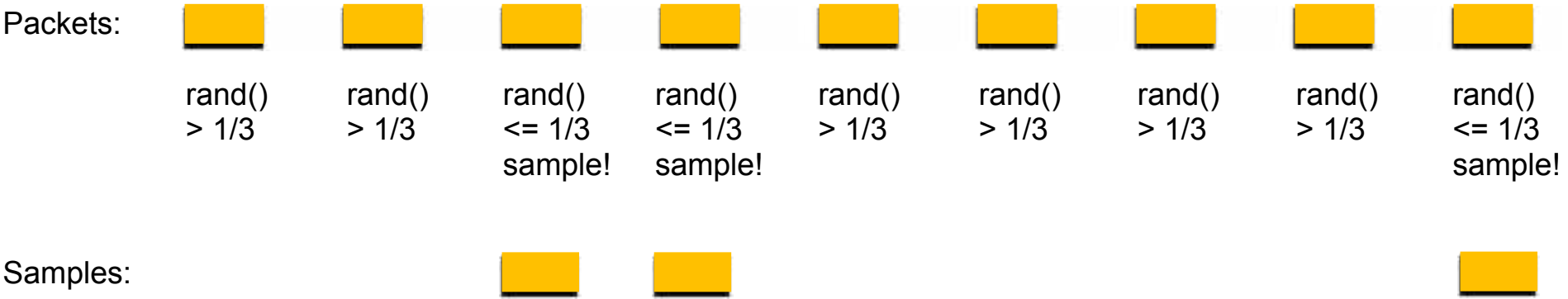
Packets Sampling Strategies [1/2]

- 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 <= r <= 1
  - Sample if r <= 1/N
- Synchronization with periodic traffic patterns is prevented with randomness
- Example with N=3, rand() = 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 = 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, 100,00 + 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]

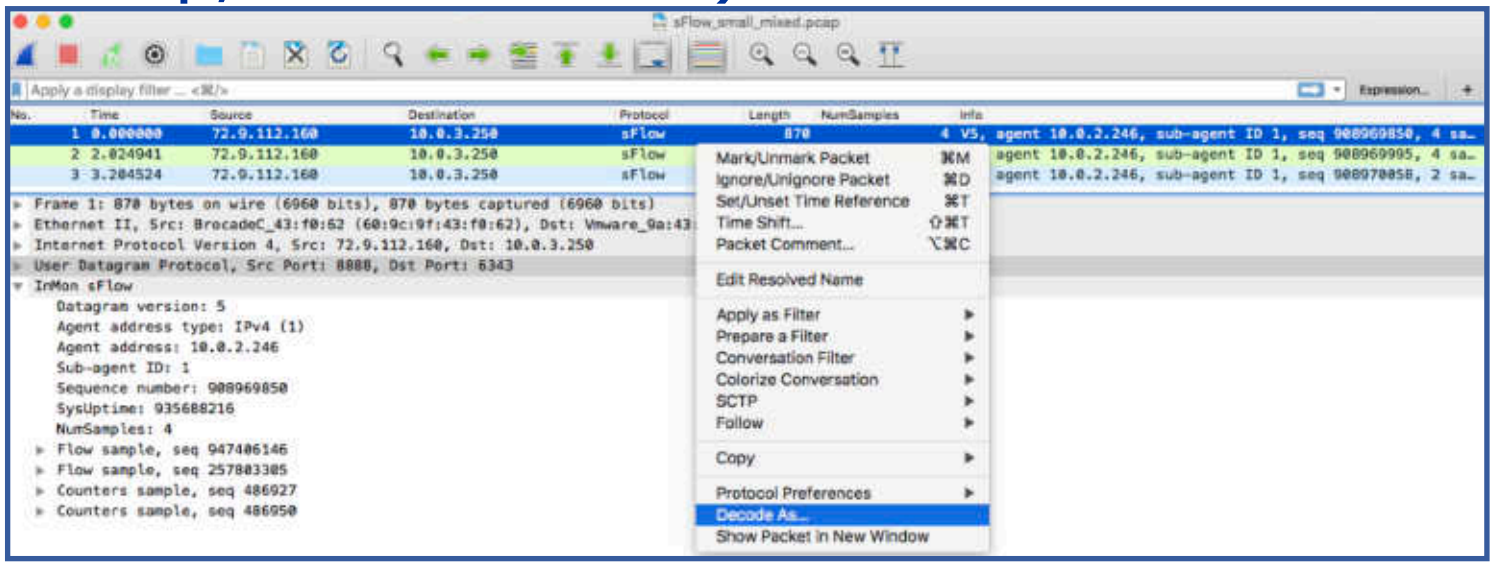
- The estimate of the percentage error %error
  %error = √(1 / c)
- In the HTTP example
  - %error = 196 * √(1 / c)
    = 196 * √(1 / 1,000)
    = 6.20 %
- The largest likely error is 6.20 %
- Note: %error formula given is an approximation and only works well when n >> c

Statistical Background

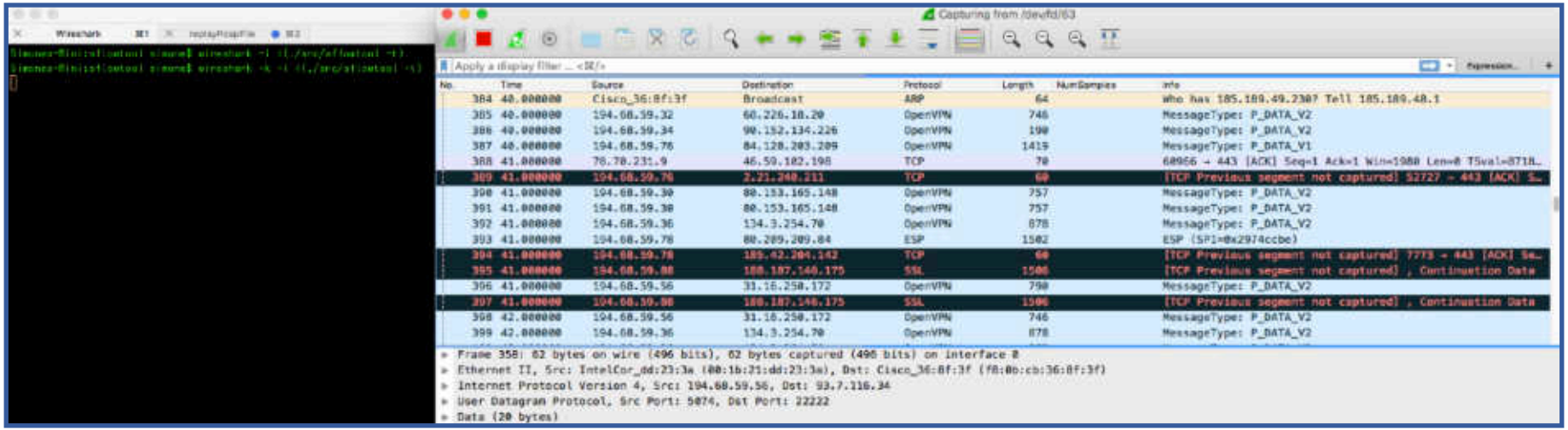
- Assumption is that packet sampling can be modeled by the binomial distribution
- Prove that measured statistics can be used to accurately estimate the parameters of the actual theoretical binomial distribution
- Use the central limit theorem to compute the confidence intervals of a normal curve

DEMO: Wireshark + sFlow Traffic

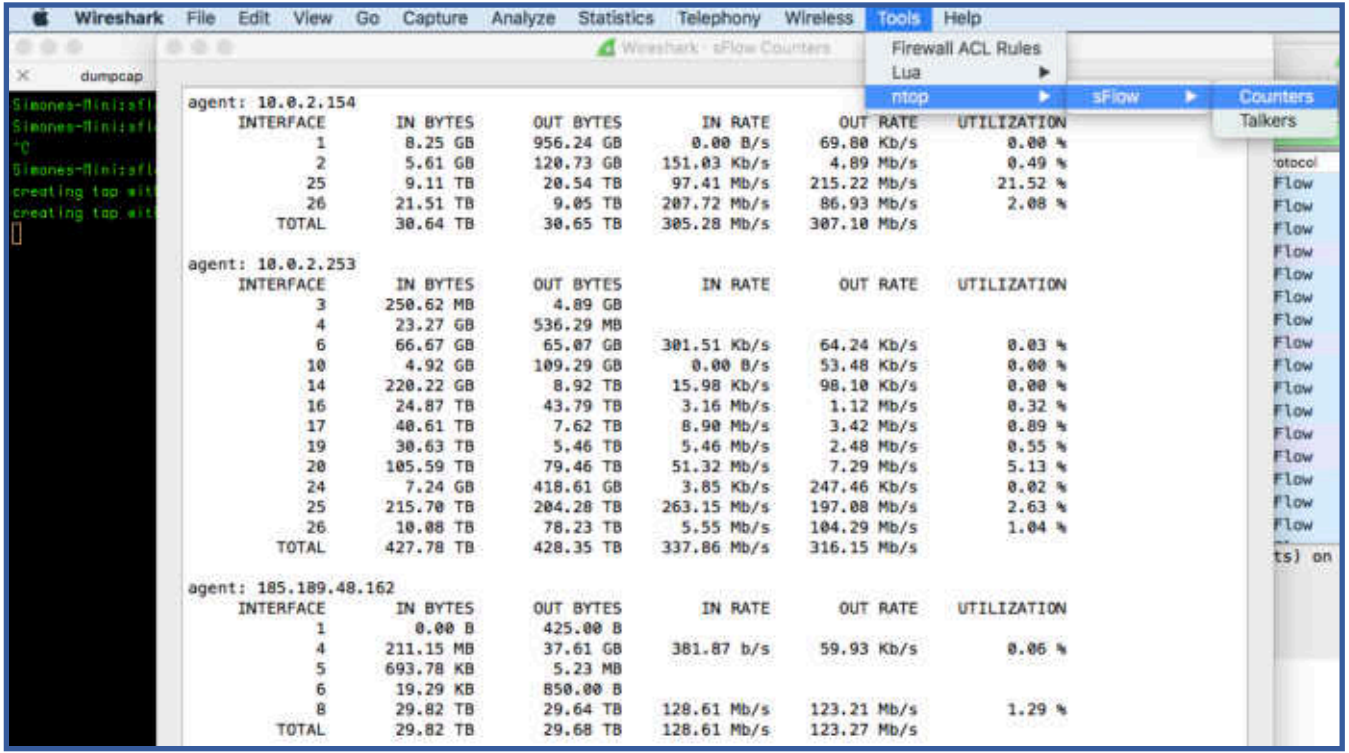
- Simply feed Wireshark with sFlow traffic (pcap, extcap, live interfaces)



DEMO: Wireshark + sFlow Flow Samples



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



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