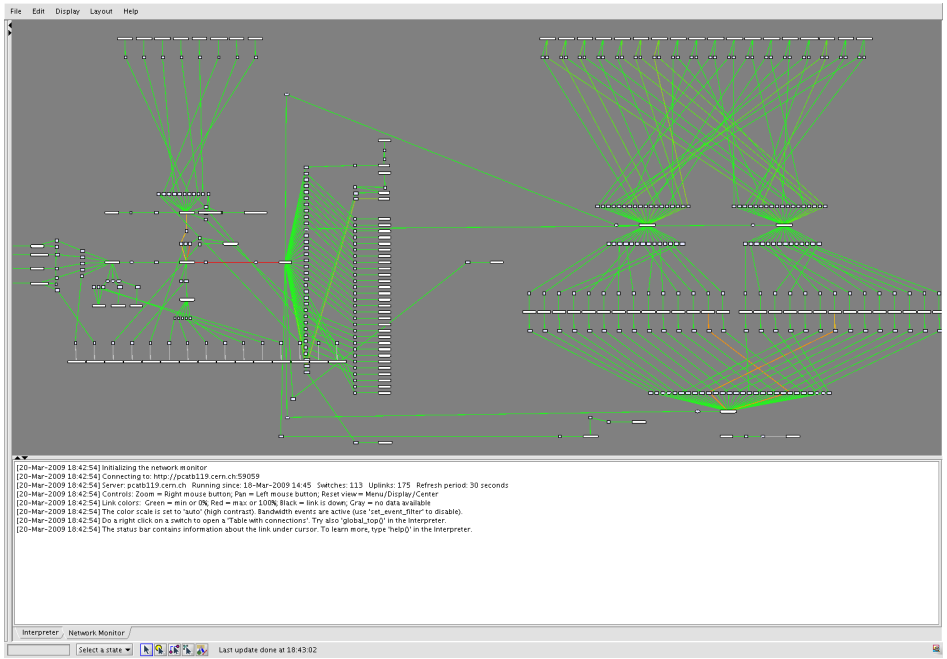
Monitoring Individual Traffic Flows in the ATLAS TDAQ Network

R.Sjoen, S.Stancu, M.Ciobotaru, S.M.Batraneanu, L.Leahu, B.Martin, A.Al-Shabibi

March 21, 2009

Outline

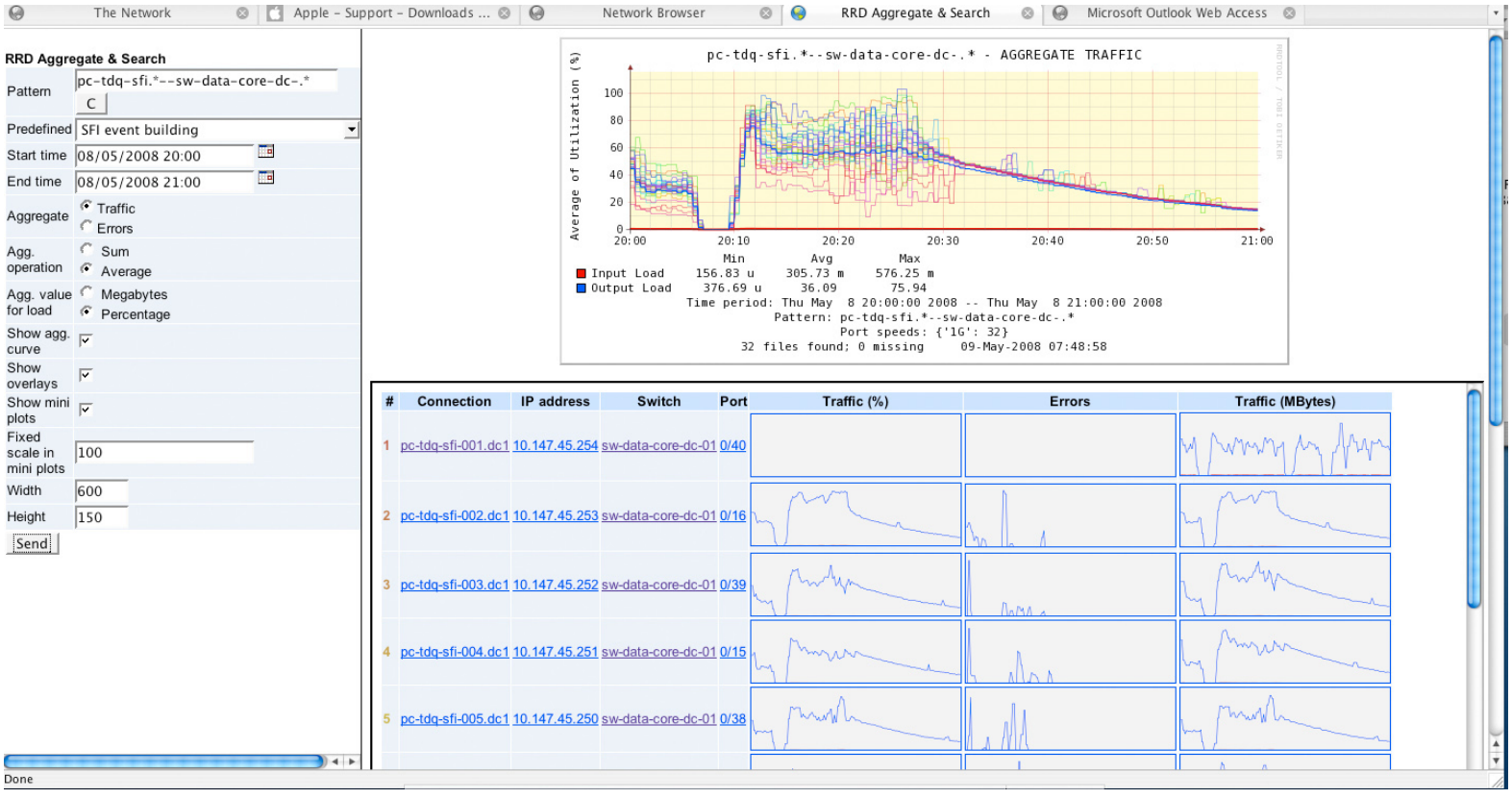
Introduction



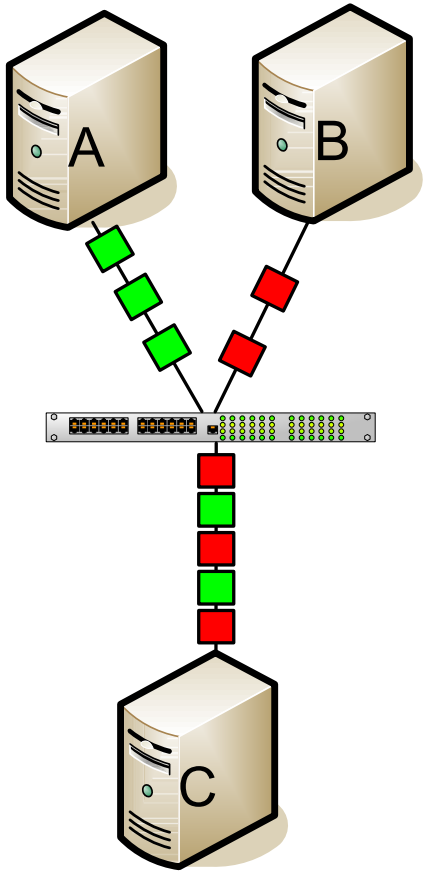
- 5 multi-blade chassis devices
- 200 edge switches
- 2000 processors
- Well known set of applications

- Classical SNMP-based monitoring to provide statistics on aggregate traffic
- Difficult to monitor, troubleshoot and quantify single traffic flows

Monitoring tools - Aggregate browser



What are Traffic flows?



- A set of packets belonging to a single conversation between two logical entities
- Traffic flows can be defined on each layer of the TCP/IP model, we define 4 types of conversations
  - An Ethernet conversation is between two interfaces within the same broadcast domain
  - An IP conversation is between hosts, but it can also span several networks
  - A TCP or UDP conversation is a conversation between two sockets on different hosts

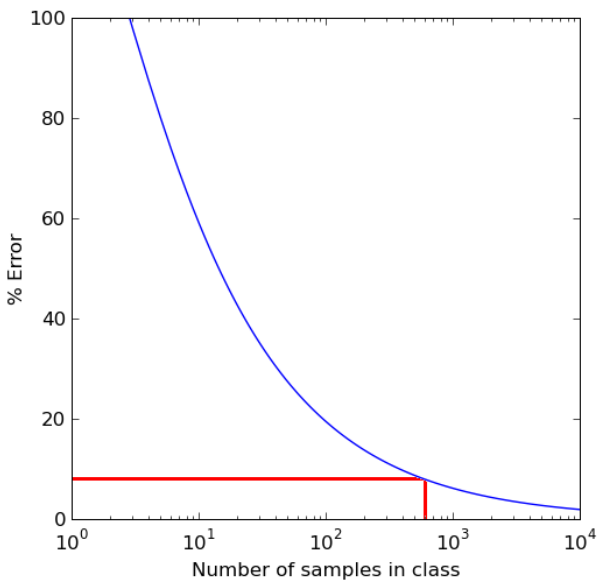
Statistical sampling

- Can't look at everything, even though we would like to
- We examine only a subset of the packets that traverse the network
- Enough to make assumptions about the traffic

Sampling a gigabit ethernet link

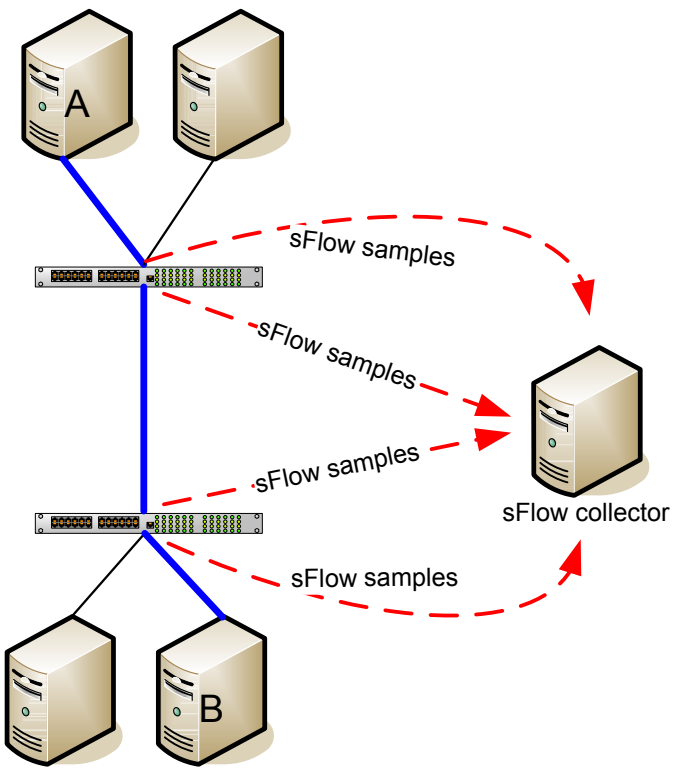
- Frames per second between 80k and 1.2m.
- Sampling rate of 512 gets between 150 and 2300 samples per second.

Accuracy of statistical sampling



- 1 million packets
- 2000 of these are sampled
- 500 of the sampled packets belong to traffic flow A
- We can with 91% accuracy assume that 25% of all the packets belong to flow A
- Accuracy not dependent on total number of frames, but on the number of samples
- We can increase accuracy on short and low rate traffic flows by increasing the sampling rate

Statistical sampling with sFlow



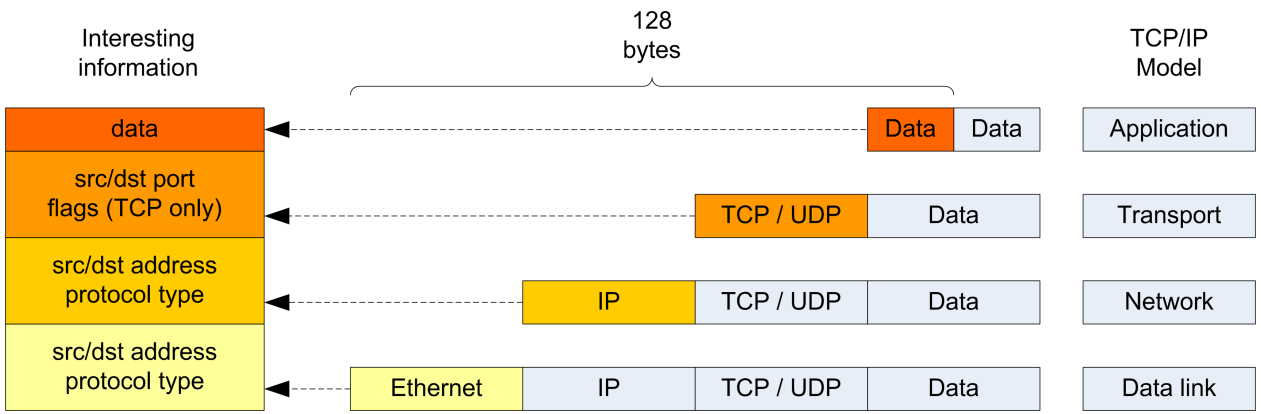
- Mechanism to capture traffic data in switched or routed networks
- Applicable for high speed networks
- Implemented in hardware
- UDP for transport layer communication
- Supported by multiple vendors, including HP and Force10

An sFlow agent running on a port delivers flow samples (on average 1 out of N packets) to a collector

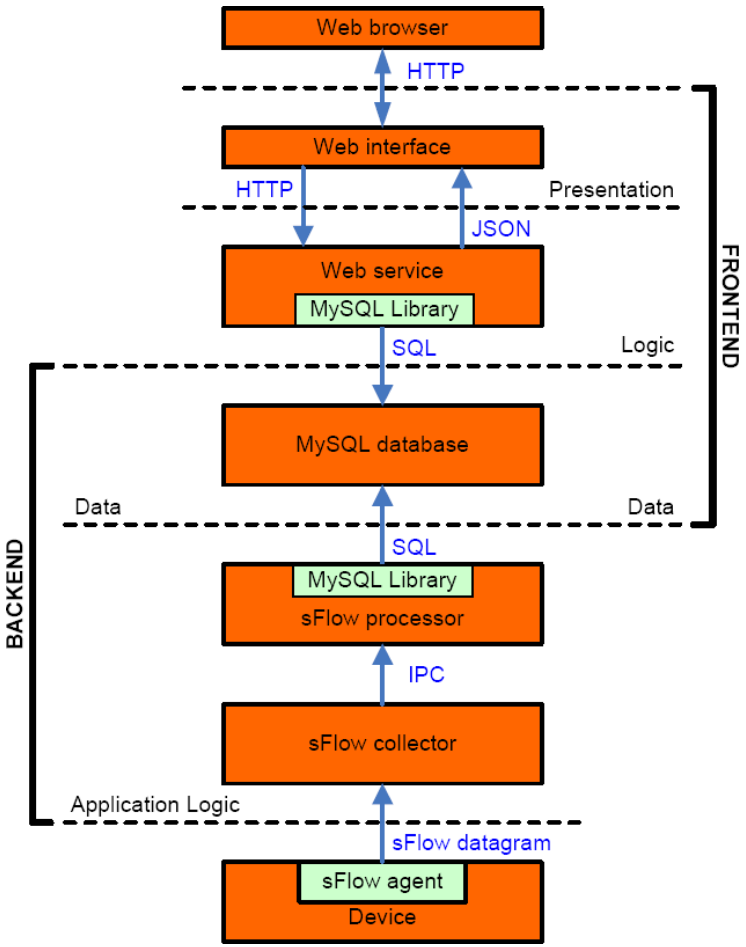
What information can we get ?

Any information contained in the packet header up to the first 128 bytes, for most packets this also includes a part of the payload.

- Conversations between hosts
- Conversations between applications
- Protocol distributions on each layer
- Distribution of packets with certain flags or properties



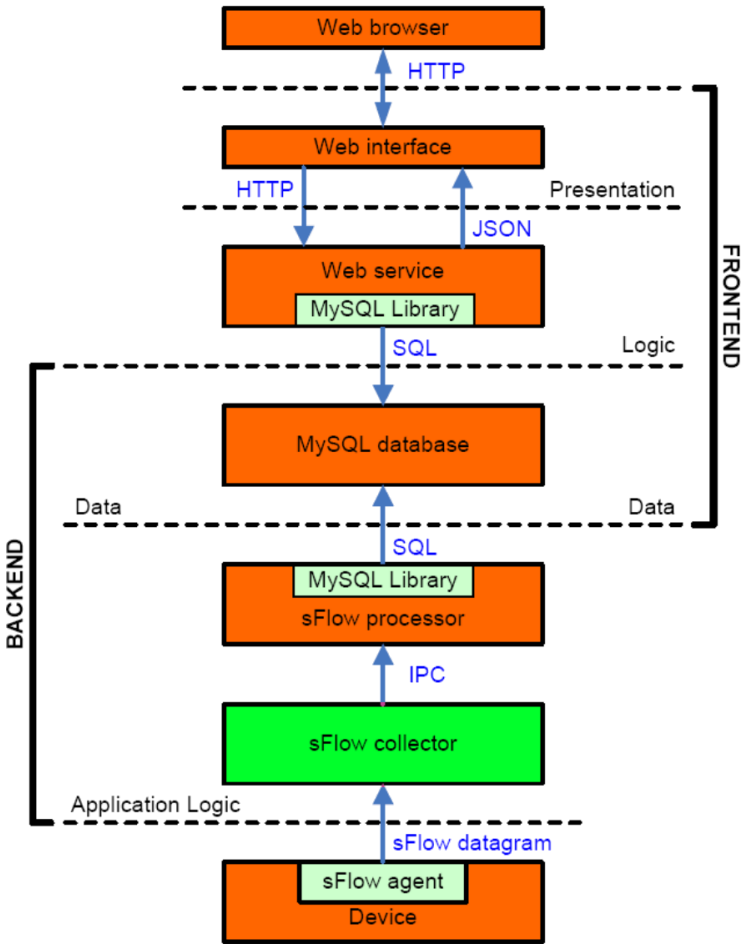
System architecture components overview



- sFlow agents
- sFlow collector
- sFlow processor
- MySQL database
- Web service
- Web interface

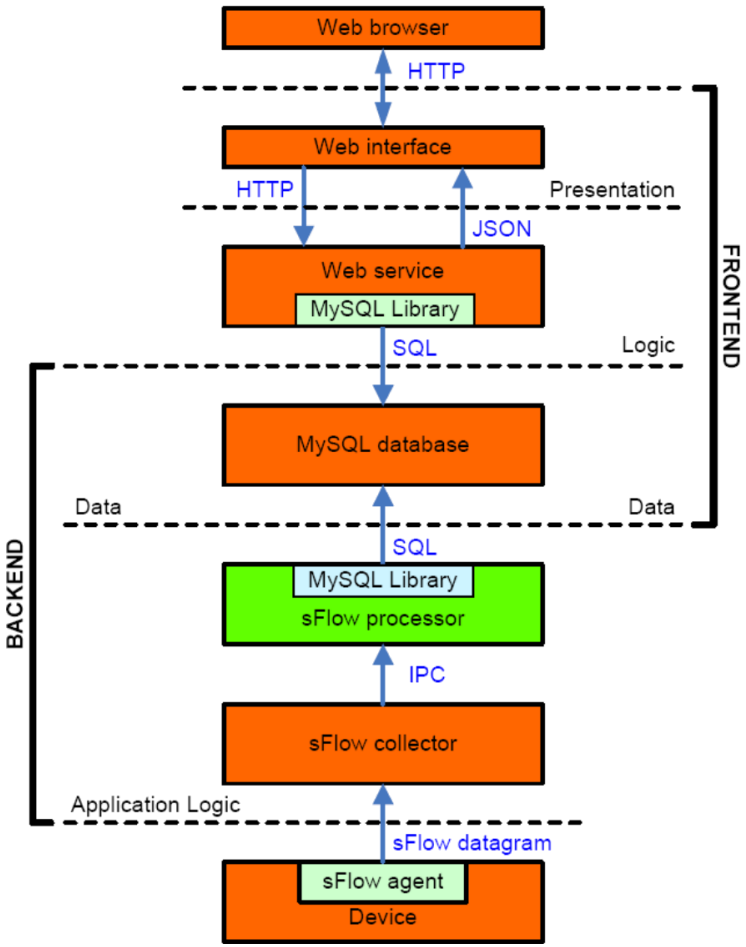
The system has been developed with modularity in mind

The sFlow collector



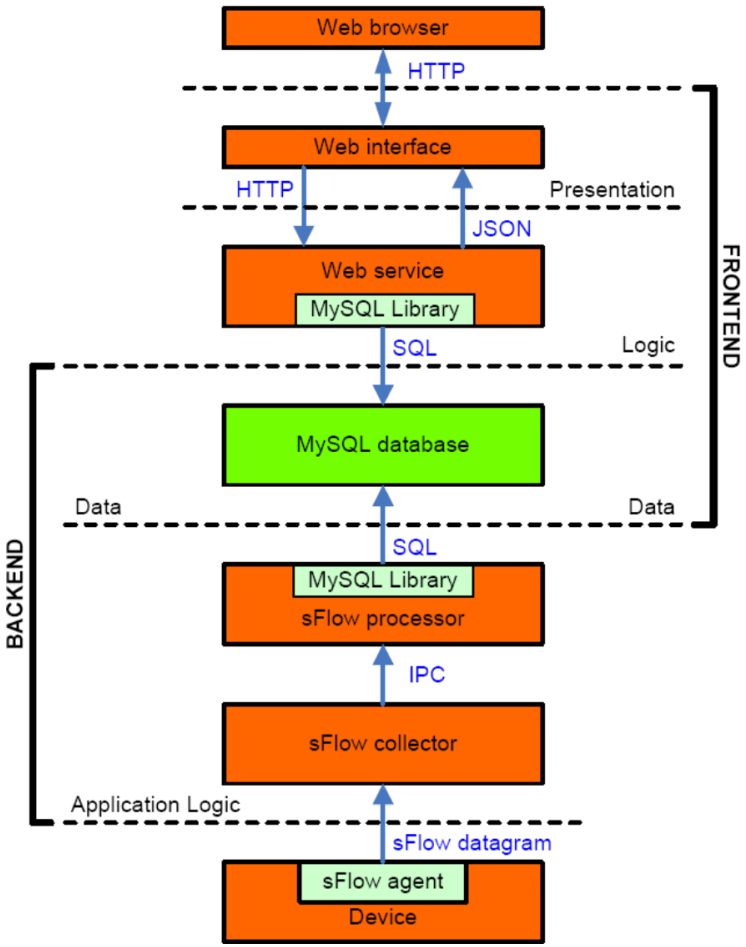
- Implemented in C
- Optimized for speed
- Stores samples in temporary files

The sFlow processor



- Implemented in C
- Responsible for extracting information from the files that the collector stores
- Interacts with the data storage system

The MySQL database



Optimized for high speed continuous inserts

- Using bulk inserts
- Disabling instant flushing of transactions
- Increasing the size of the buffer pool and key buffer

Sample based storage

The first implementation of our collector used this method

Pros

- Easy to implement
- Fast and simple storing

Cons

- A lot of work required to process and display the data to the user
- Data rate directly proportional to the sampling rate
- Results in sacrificing accuracy for space, reducing the accuracy with regards to short-lived traffic flows

Conversation based storage

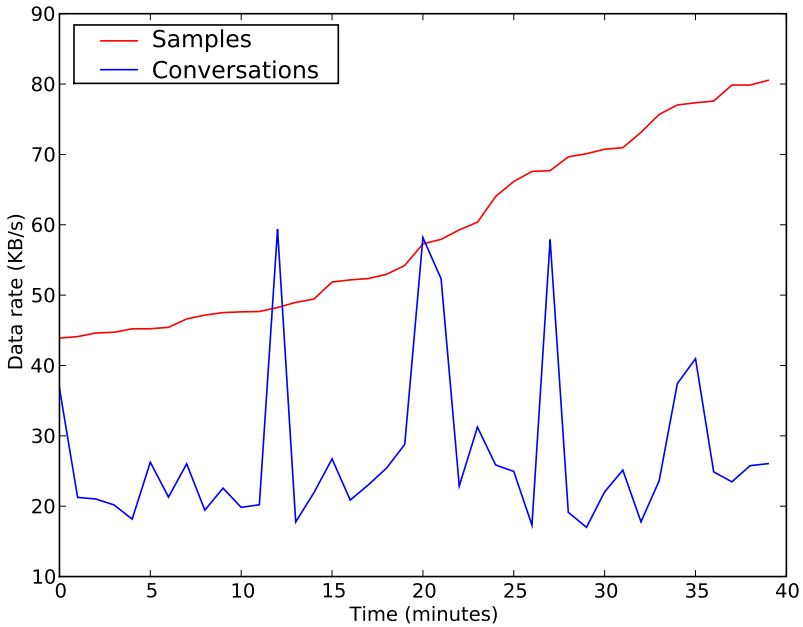
Pros

- Storage size is dependent on the number of conversations
- Increasing the sampling rate to capture short-lived traffic flows has a minimal impact on data rate

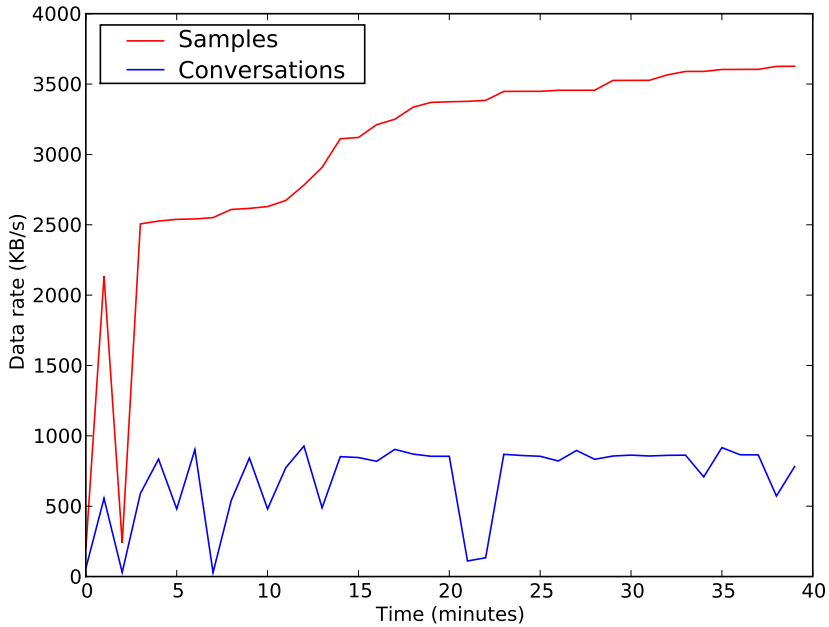
Cons

- More processing to do before storing, because we need to extract all the information we need
- Requires a more advanced data structure for storage than the sample based storage.

Storage rate - samples vs conversations

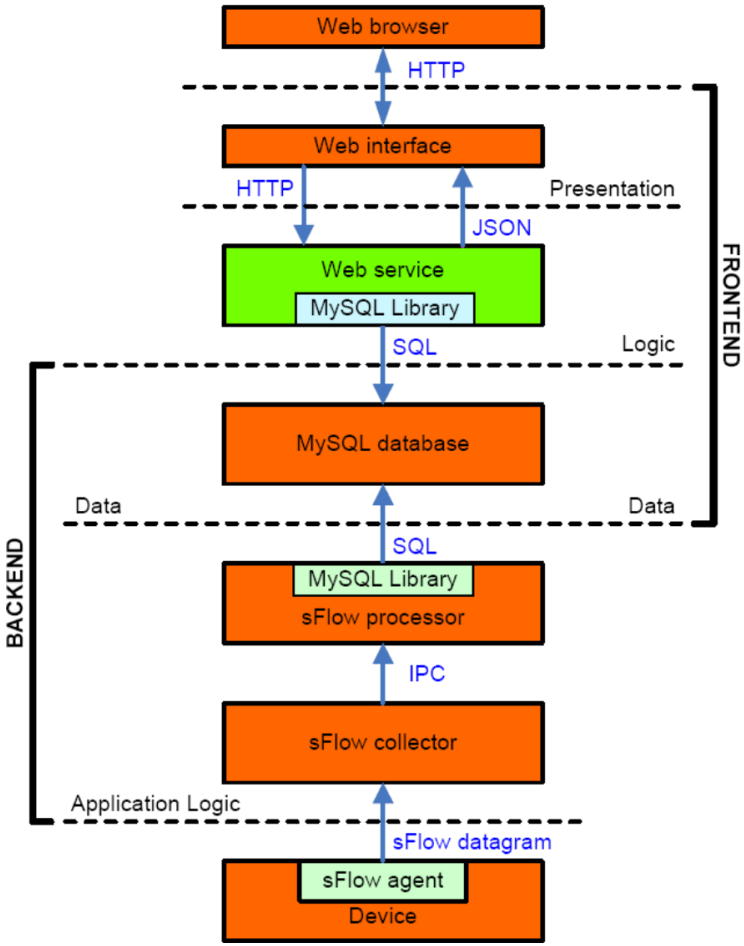


Almost idle network



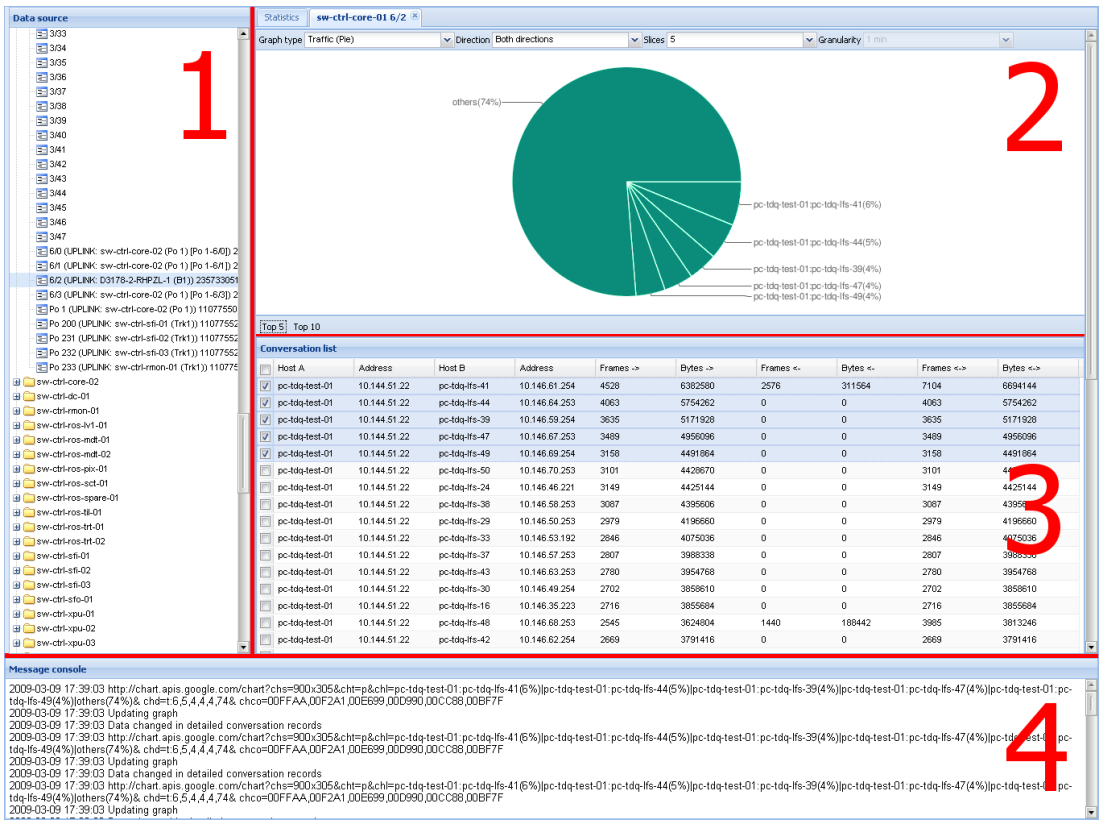
Busy network

Web service



- Written in Python
- Runs as a web service under Apache
- Exports API to access the data
- Works with JSON (JavaScript Object Notation)
- Prepares the data to be displayed, resolving hostnames, service names etc

Web interface



- Early stage development
- Implemented using the Google Web Toolkit
- Powerful cross-browser compatible AJAX interface
- Uses Google Charts API for visualizing data

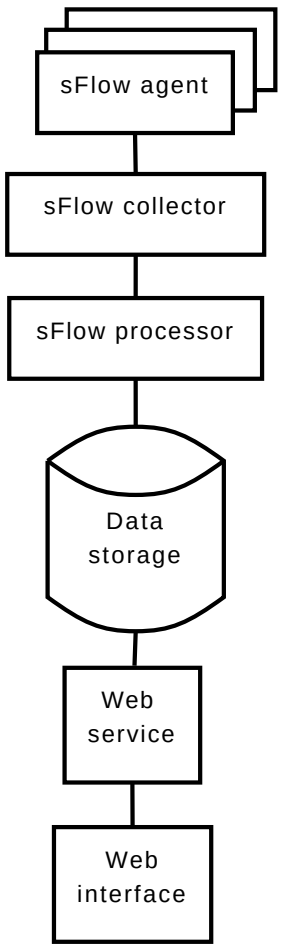
Distributing the system

The Atlas TDAQ network only partially populated

- We are currently operating with approximately 40% of the total number of nodes
- New nodes will have an increased number of cores, leading to more processes running on each node
- More processes leads to more conversations per node

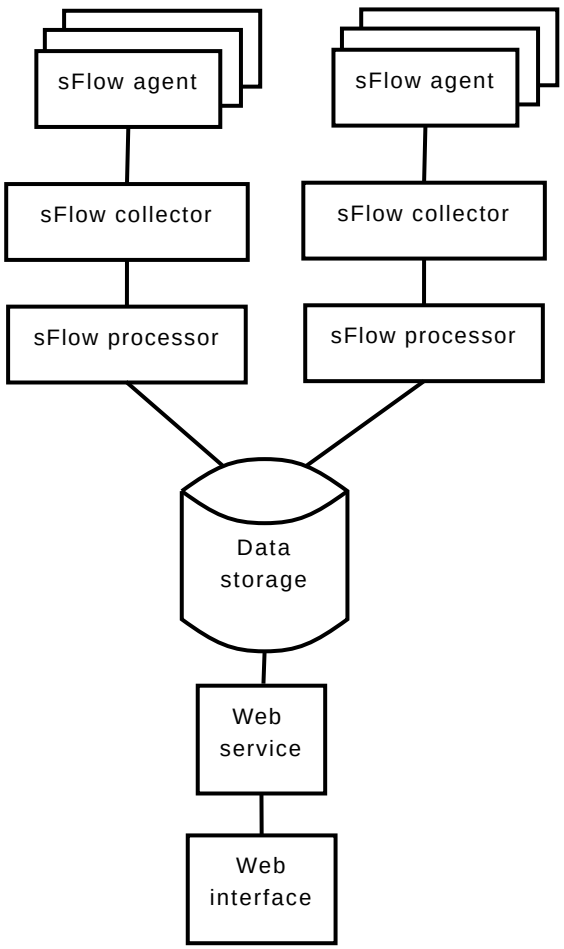
Because of this we have examined different ways to distribute the system

Non-distributed



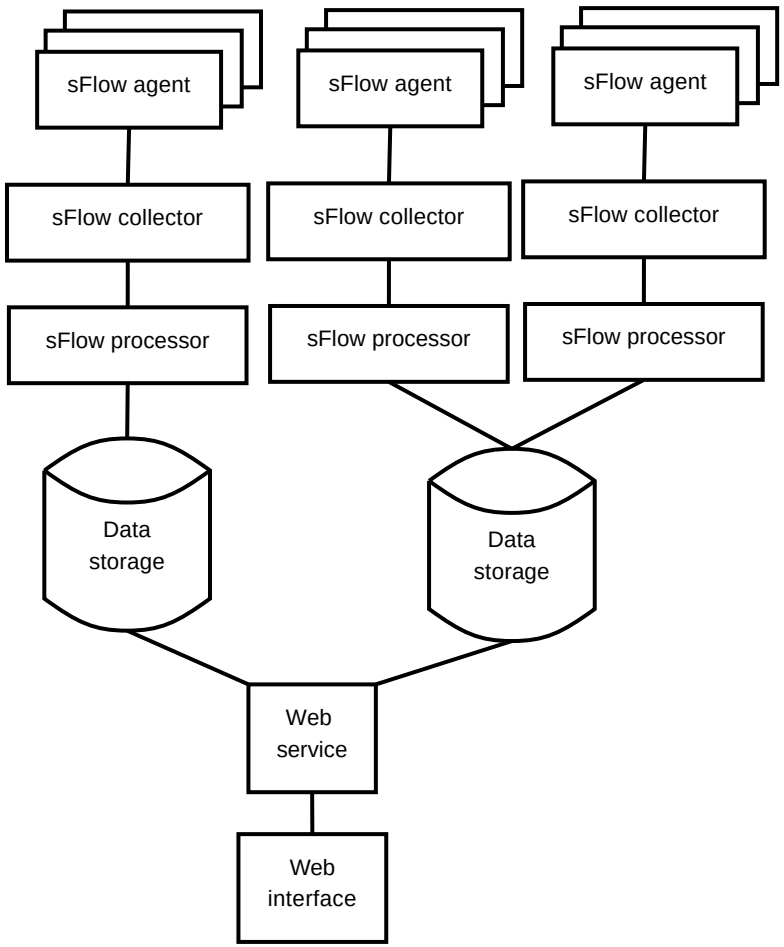
- Running prototype
- Easy to configure and manage
- Fast and simple
- Highly dependent on the system it is running on
- Bottleneck is mainly between the sFlow processor and the data storage service
- sFlow data needs to be aggregated into a single host, consuming bandwidth

Partially distributed



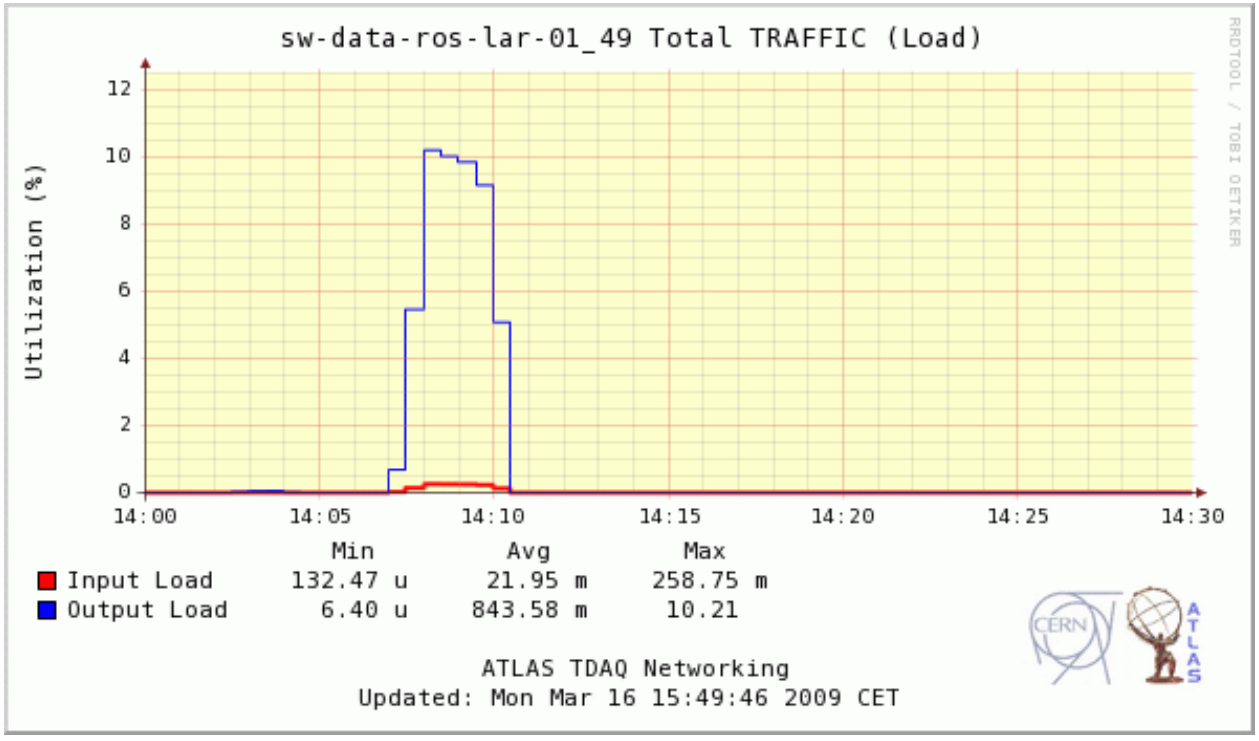
- Optimal for high sampling rates
- Distribute the bandwidth usage consumed by sFlow data by not having to aggregate all the sFlow traffic to the same collector
- Splitting the work required to process the samples
- Centralized storage

Fully distributed



- Each collector has its own database
- Provides the same advantages as the partially distributed model
- Faster response time when communicating with the web service
- Requires extra application logic in the web service

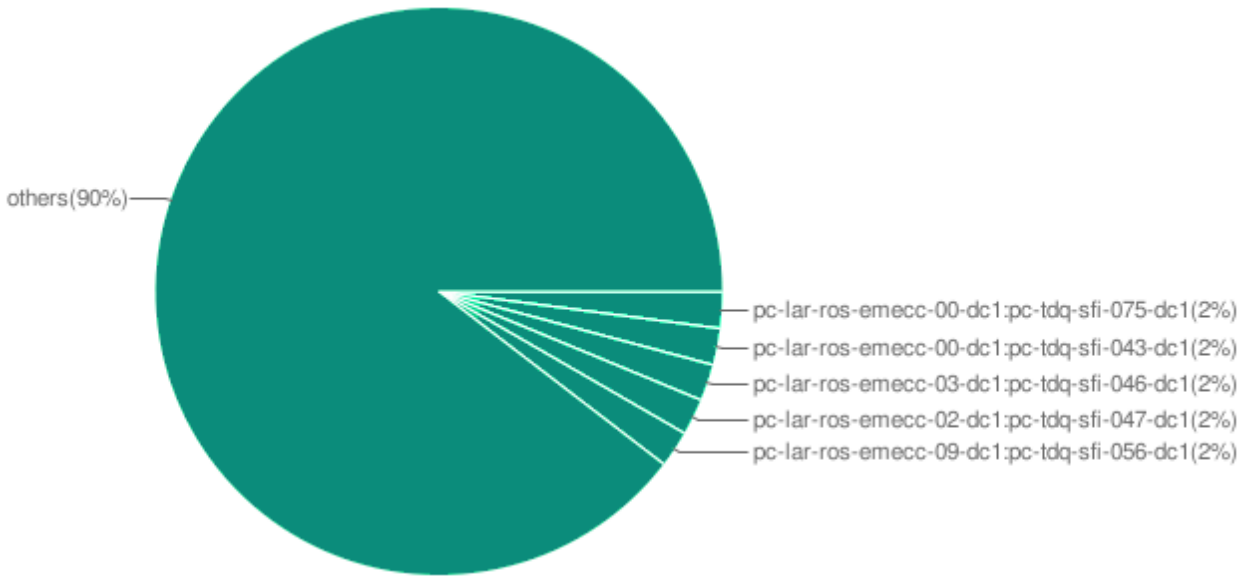
An unknown spike in traffic



A 3-minute interval with a burst consuming 10% of the egress link

Using sFlow to display the bandwidth usage

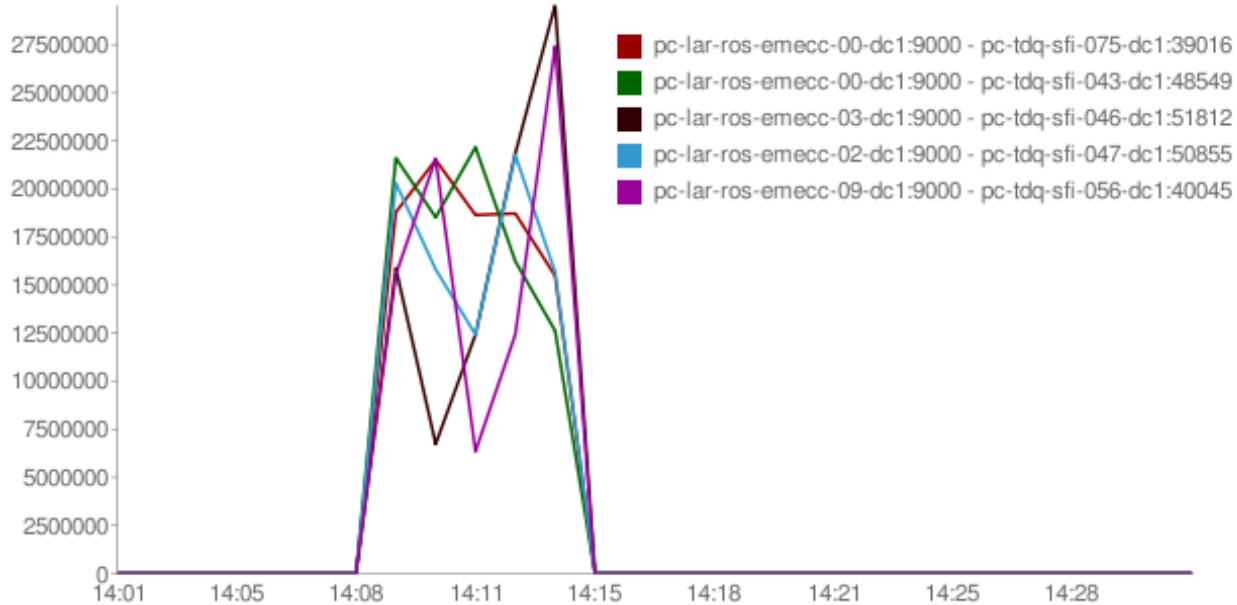
We are able to determine which hosts are contributing to the traffic, the chart shows the top 5 hosts.



We can see that the top 5 hosts are using approximately equal shares, we can suspect that the same application is running on all the hosts

Using sFlow to determine which applications

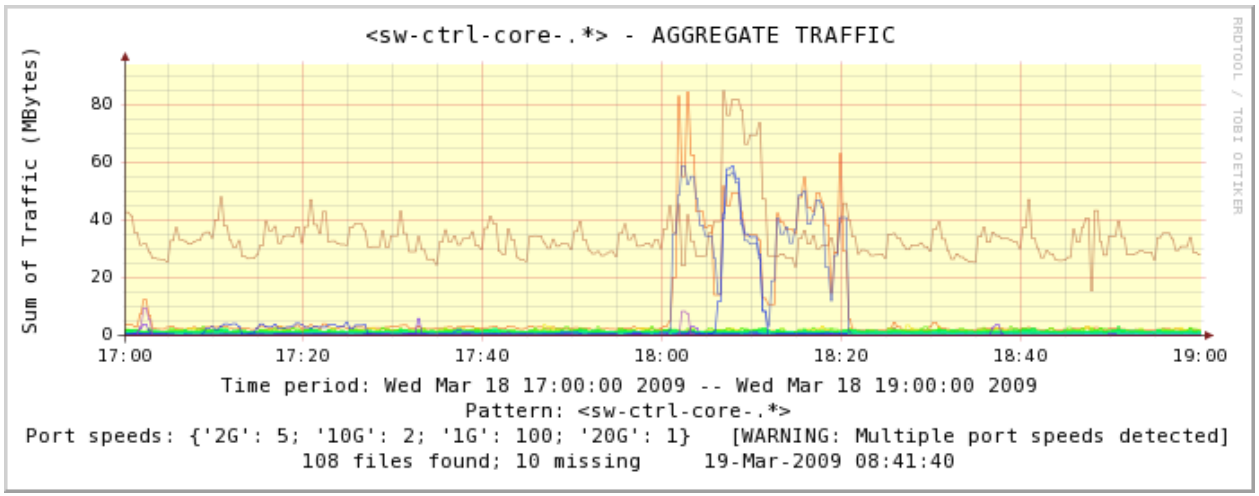
By looking at the port numbers we can determine the top 5 applications on these hosts which are generating the traffic



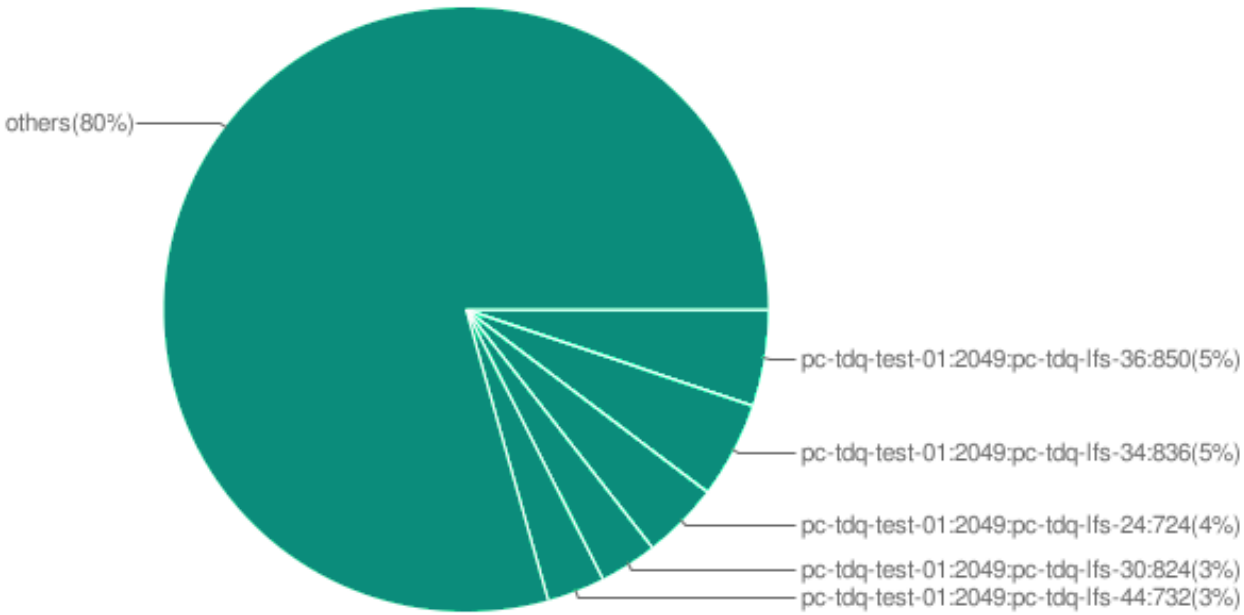
In this example we can see that the service port is 9000, which corresponds to a well known application in our network

Some unidentified control traffic

While looking for anomalous control traffic we notice a continuous traffic load on a port that has nothing to do with data taking.



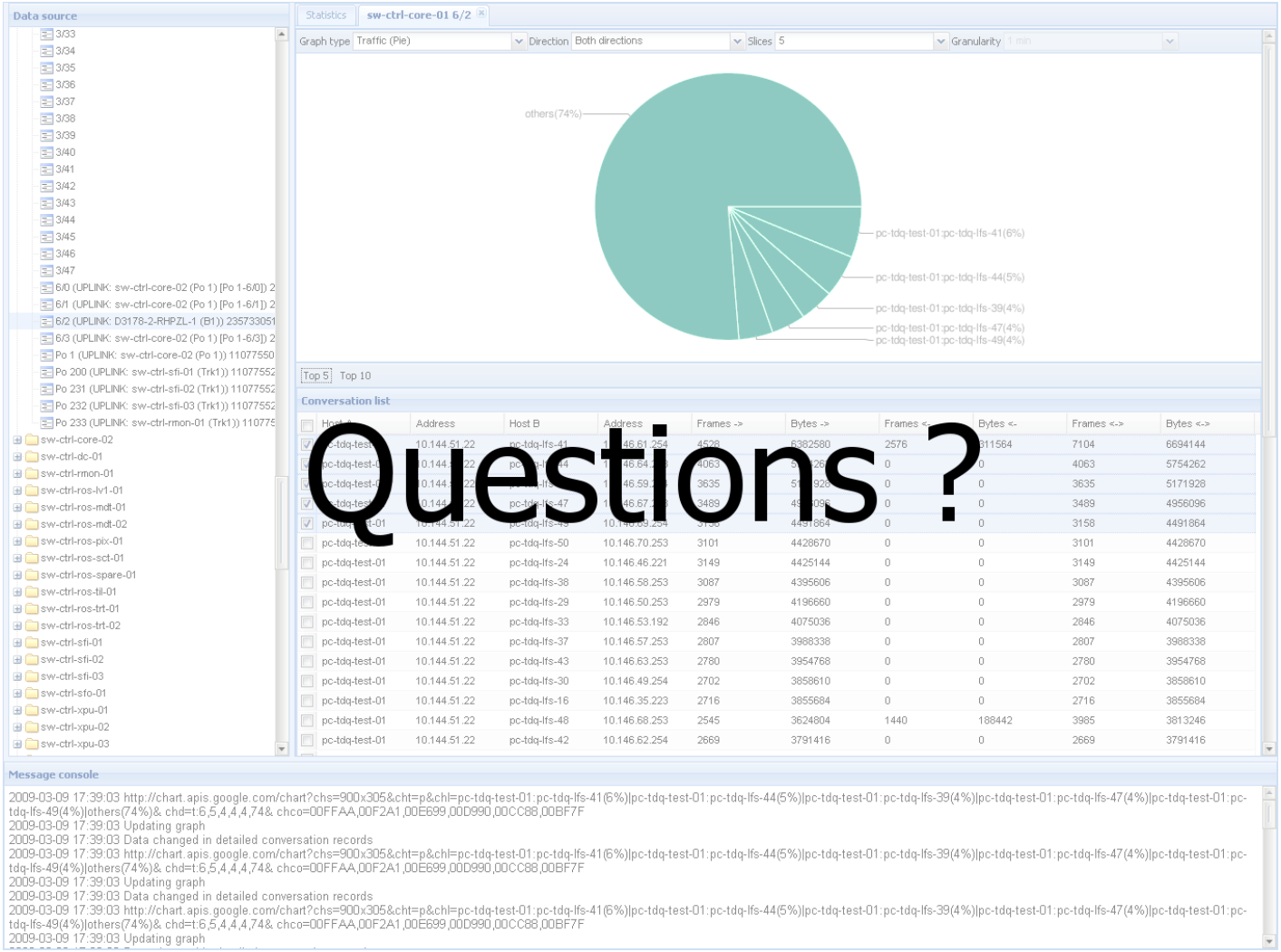
Using sFlow to display the bandwidth usage



We are able to identify the machine generating the traffic as some diagnostic test which is being run by the system administrators.

Conclusion

- Taking it one step further compared to classical SNMP based monitoring
- By gaining a deeper knowledge about the traffic flows we have the ability to identify unknown traffic patterns
- When a problem is detected diagnostics can be performed immediately without having to reproduce the problem
- An intuitive interface allow non-expert users to easily obtain the desired information
- Historical analysis of events only limited by storage space


Data source
Statistics
sw-ctrl-core-01 6/2
Graph type
Traffic (Pie)
Direction
Both directions
Slices
5
Granularity
1 min
others(74%)
pc-tdq-test-01:pc-tdq-lfs-41(6%)
pc-tdq-test-01:pc-tdq-lfs-44(5%)
pc-tdq-test-01:pc-tdq-lfs-39(4%)
pc-tdq-test-01:pc-tdq-lfs-47(4%)
pc-tdq-test-01:pc-tdq-lfs-49(4%)
Top 5
Top 10
Conversation list
Message console
Questions?