FLORIDA INTERNATIONAL UNIVERSITY
Miami's public research university

Distributed Network Monitoring using NetFlow and MonALISA

Developed in Joint Collaboration between:

Dr. Xun Su – California Institute of Technology
Jose Luis Fernandez – Florida International University
Ernesto Miguel Rubi – Florida International University

NSF Award Number Number: SCI – 0231844

Table of Contents

Table of Figures

Introduction

Building on the success of previous research conducted under the National Science Foundation's Strategic Technologies for the Internet (STI): Research Experience for Undergraduates Award No. 331112 we find strong motivation to integrate the obvious need for network monitoring and understanding with a distributed platform that can be molded to fit the scalable needs of an ever expanding networking topology. Various technologies are currently widely implemented by networks of varying sizes and complexity to gain crucial understanding of traffic patterns; one such popular method is NetFlow. It is with a drive for insightful knowledge of NetFlow; its key advantages and drawbacks, that we chose it to be the tool of choice when analyzing network traffic behavior across interfaces on AMPATH's production network. It is important to note that NetFlow by itself cannot be easily parsed to produced the comprehensive data analysis which we sought during our research; however, another widely used open source tool: Flow Tools was the primary parser for NetFlow data being streamed from the multiple network elements which we chose to investigate.

Both NetFlow and Flow Tools provide the data gathering and analysis infrastructure to our project as well as many network engineers; and in fact, it is the case today that these technologies are a commonplace pairing which yields important results critical to understanding the true behavior of networks all over the world. However, a critical part to the research that we undertook was to integrate the parsed NetFlow data into a tool which would make available the information in a distributed format; incorporating a historical component as well as a real-time understanding of data being received from the network components being monitored and analyzed. Having worked closely with our peers at Caltech and CERN we quickly identified *monALISA (Monitoring Agents using a Large Integrated Services Architecture)* as a promising tool that could fit our needs due to its design architecture (JAVA / JINI based) and overall philosophy as a tool to provide monitoring information from large as well as distributed systems. MonALISA's flexibility as a tool to gather, store and distribute network data collected was crucial to the success of our investigation and it shall become apparent throughout the course of this report.

One further technology that we intended to explore was that of the National Laboratory for Applied Network Research (NLANR) PMA (Passive Monitoring Agent). There are key differences between PMA data and NetFlow data which are worthwhile of research effort. With our results we hope to provide a stable platform from which networks of varying degrees can be closely monitored, their traffic patterns clearly identified and the appropriate decisions taken to rectify issues which negatively impact performance or augment those which have a positive impact on the delivery of service to an end user.

Background

The AMericasPATH (AMPATH) network is an FIU project sponsored in part by the US National Science Foundation CISE directorate, in collaboration with Global Crossing and other telecommunications product and service providers. Using Global Crossing's terrestrial and submarine optical-fiber networks, AMPATH is interconnecting the research and education networks in South and Central America and the Caribbean to US and non-US research and education networks via Internet2's Abilene network.

The purpose of the AMPATH project is to allow participating countries to contribute to the research and development of applications for the advancement of Internet technologies. The mission of AMPATH is to serve as the pathway for Research and Education networking in the Americas and to the world and to be the International Exchange Point for Latin America and the Caribbean research and education networks. Additionally AMPATH fosters collaboration for educational outreach to underserved populations both in the US and abroad. The AMPATH pathway serves as the bridge between Central and South American National Research Networks (NRENs) and the world's research and education networks. With the multiplicity of complex networked systems and educational activities served by AMPATH's wide-ranging infrastructure a strong demand for high availability and engineering collaboration arises. It is met through the use of various monitoring agents to provide an strong factual foundation to troubleshooting. Likewise, deciphering the everyday activities of our peers is achieved with a distributed approach to data gathering and dissemination.

The MonALISA framework provides a distributed monitoring service that not only is closely integrated with our monitoring and data distribution philosophy but also acts as a dynamic service system. The goal is to provide the monitoring information from large and distributed systems in a flexible, self-describing way. This is part of a loosely coupled service architectural model to perform effective resource utilization in large, heterogeneous distributed centers. The framework can integrate existing monitoring tools and procedures to collect parameters describing computational nodes, applications and network performance. [1]

[1] http://monalisa.cacr.caltech.edu

Anatomy of an International Exchange Point

Having stated AMPATH's purpose we now define the current state of the international exchange point; having evolved through several iterations of new NREN peers as well as demonstrated its capacity to act effectively as a local facilitator of HPC network connectivity through the South Florida GigaPOP infrastructure.

Below, Figure 1 demonstrates AMPATH's current design, IP addresses as well as ASNs are omitted; we will use simple NREN names to identify our international peers. For more detailed information please visit http://mrtg.ampath.net.

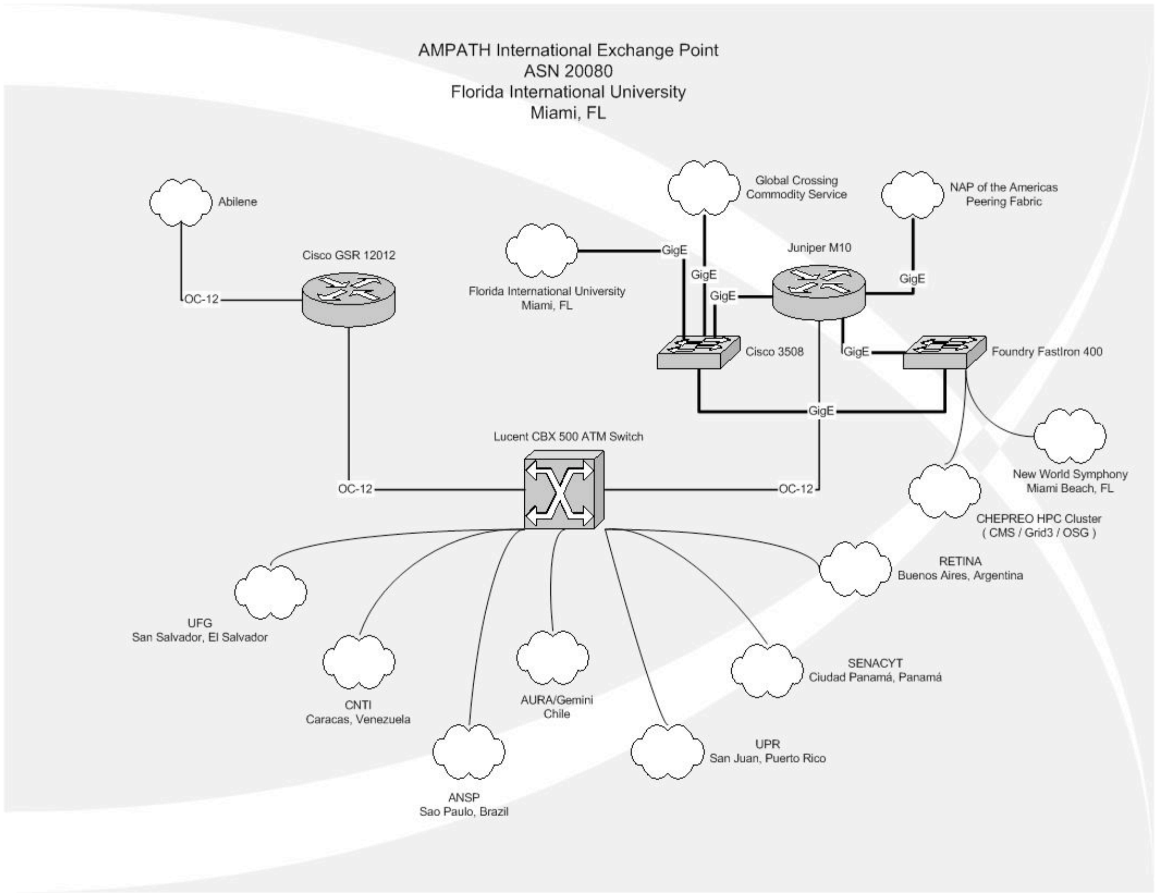

AMPATH International Exchange Point
ASN 20080
Florida International University
Miami, FL
Abilene
Cisco GSR 12012
OC-12
Global Crossing Commodity Service
NAP of the Americas Peering Fabric
Juniper M10
GigE
Florida International University Miami, FL
Cisco 3508
Foundry FastIron 400
Lucent CBX 500 ATM Switch
New World Symphony Miami Beach, FL
CHEPREO HPC Cluster (CMS / Grid3 / OSG)
RETINA Buenos Aires, Argentina
UFG San Salvador, El Salvador
CNTI Caracas, Venezuela
AURA/Gemini Chile
SENACYT Ciudad Panamá, Panamá
UPR San Juan, Puerto Rico
ANSP Sao Paulo, Brazil

Figure 1 - AMPATH in its current state

It is this network that served as the backdrop for our study. Two core routers exist; a Cisco GSR 12012 as well as a Juniper M10; both routers have NetFlow accounting enabled and are designed to export this data to a collection workstation.

NetFlow

NetFlow was originally developed by Darren Kerr and Barry Bruins at Cisco Systems in 1996 as a switching path. Today NetFlow is primarily used for network accounting. NetFlow is data collected and exported by a router. It contains information about all flows processed by that router. A flow is IP data which has the following seven identical characteristics:

- Source IP address
- Destination IP address
- Source port
- Destination port
- Layer 3 protocal type
- TOS byte
- Input logical interface

NetFlow records only unidirectional traffic inbound to any interface on the router. Even though this traffic is unidirectional NetFlow accounts for all traffic going in and out of the router by recording both transit traffic and traffic destined for the router.
.
By storing only the router's flow information and neglecting payload it becomes feasible to store large amounts of data. NetFlow data can be used to describe traffic on a network, view trends, identify DOS attacks and many other applications. Many network vendors now implement various flavors of NetFlow, all similar in achieving the main goal of recording flows through various interfaces on the network device.

We now focus on configuration details for both core routers.

Cisco GSR 12012 – NetFlow Configuration

```
ip flow-export source Loopback0
ip flow-export version 5
ip flow-export destination 131.94.191.101 2058
ip flow-sampling-mode packet-interval 100
```

Juniper M10 – NetFlow

```
forwarding-options {
    sampling {
        traceoptions {
            file sampled-trace files 4;
        }
        input {
            family inet {
                rate 100;
                run-length 4;
                max-packets-per-second 5000;
            }
        }
        output {
            cflowd 131.94.191.101 {
                port 2059;
                source-address 198.32.252.34;
                version 5;
                autonomous-system-type origin;
            }
        }
```

Examining the previous configuration we see the need to specify a source interface on the GSR router on which to export the NetFlow data gathered; similarly on the Juniper the export interface is given. It is not necessary, however, to explicitly define the router's loopback interface as the export source interface. It is strictly an arbitrary decision which interface to choose, provided the configuration on the collection mechanism (workstation) can be properly modified to accommodate for allowing the IP address on said interface sufficient access through local security measures (i.e: iptables, ipchains, any other firewall rules, etc.).

Note that on both core routers the collection workstation is the same; the collection daemon that is running at the data repository is set to listen on port 2059 and so equally, both core routers must know of this port requirement in order to successfully establish communication with the collection process running locally at the repository workstation. JunOS calls NetFlow 'cflowd' but this is very similary to the Version 5 which runs on Cisco IOS.

A main difference that we will explore in our later use-case scenario is the ASN information gathered by the Juniper M10. The use of AS numbers associated to NetFlow data makes the information much more humanly readable and provides a good deal of aggregation to quickly make sense of peering relationships and overall traffic patterns.

The NetFlow version is set to be v5 on the Cisco GSR as well as on the Juniper M10; site administrator preferences maybe different, multiple versions of NetFlow exist and in the case of choosing which NetFlow version to run at a specified router/site familiarity is more often than not a determining factor. What should be apparent is the exporter / importer relationship that is established and needed for collection of gathered data; the router configuration specifies a host on the FIU campus network (131.94.0.0/16) on which NetFlow collection will run.

A key technical note is the sampling mode which the router is running. In the GSR's case a 1-100 sampling rate is specified. This means that for every 100 packets processed by the forwarding engine or route processor there will be one packet extracted and reported to the NetFlow process running at the router. This is the lowest allowed sampling rate on our GSR running NetFlow v5 and clearly we can see that this causes limitations on the analysis of network data. It is not uncommon to have short host-to-host sessions where the overall transmission does not exceed 100 packets. It is beyond the scope of this paper to discuss NetFlow sampling algorithms but we can safely say that there is a chance that if the transmission is 100 packets or less NetFlow will not account for it. This quickly introduces a margin of error to any analysis of data flows but especially to those UDP flows which are transmitted over our core; since TCP flows with their inherent error correction ways are less prone to being ignored by the collection process.

Flow-Tools

Flow Tools is a collection of programs and libraries used to collect and process NetFlow data. These tools allow users to process stored flow data from a series of command line interfaces. Commands like flow-filter and flow-sort allow the user to filter and sort NetFlow data by IP address, port, AS number and any other parameter present in that data collected. The data is presented on the command line in a table format. However these tools do not provide a dynamic way of dynamically monitoring flow data. Through the use of MonALISA we have used NetFlow data collected and processed by Flow Tools to create a graphical medium by which to view certain characteristics of the Ampath network in a close to real time fashion.

For our analysis we implemented flow-tools version 0.66 on a dual Xeon 2.66 GHz system with a copper Gigabit Ethernet network connection to FIU’s campus network. The operating system of choice was Fedora Core 2, developed by RedHat. Below is a small startup script used to start flow-capture on the NetFlow collection workstation. Notice that the listening port is specified after the IP address of the transmitting device. A similar setup is done when the flow originates directly from a router.

```
#!/bin/sh
# description: Start Flow-Capture

case "$1" in
'start')

        su - netflow -c "/usr/local/netflow/bin/flow-capture -N0
-n288 -z6 -E1G -w /home/netflow/flows/gsr.ampath.net
0/###.###.191.101/2502"
        su - netflow -c "/usr/local/netflow/bin/flow-capture -N0
-n288 -z6 -E1G -w /home/netflow/flows/juniper.ampath.net
0/###.###.191.101/2501"
        touch /var/lock/subsys/startflows
        ;;
'stop')

        killall -9 /usr/local/netflow/bin/flow-capture
        rm -f /var/lock/subsys/startflows
        ;;
*)

echo "Usage: $0 { start | stop }"
;;

esac
exit 0
```

A summary of running processes shows the result of running the above script; which constantly listen and create/rotate files for a time span of five minutes (again, this time differential can be configured, it usually varies between five to fifteen minutes).

```
netflow   2993  0.0  0.0  4088 2004 ?        S     2004
63:04 /usr/local/netflow/bin/flow-capture -N0 -n288 -z6 -
E1G -w /home/netflow/flows/gsr.ampath.net
0/131.94.191.101/2502

netflow   3017  0.3  0.0  4404 1348 ?        S     2004
536:25 /usr/local/netflow/bin/flow-capture -N0 -n288 -z6 -
E1G -w /home/netflow/flows/juniper.ampath.net
0/131.94.191.101/2501
```

The options as to file compressions, maximum file size, number of seconds to collect for before rolling over to a new file, etc are all included in the command line arguments once flow-capture is started.

In our particular scenario[2]:

-E expire_size

Retain the maximum number of files so that the total storage is less than expire_size. The letters b,K,M,G can be used as multipliers, ie 16 Megabytes is 16M. Default to 0 (do not expire).

-n rotations

Configure the number of times flow-capture will create a new file per day. The default is 95, or every 15 minutes.

-z z_level

Configure compression level to z_level. 0 is disabled (no compression), 9 is highest compression.

-N nesting_level

Configure the nesting level for storing flow files. The default is 0.

```
-3   YYYY/YYYY-MM/YYYY-MM-DD/flow-file
-2   YYYY-MM/YYYY-MM-DD/flow-file
-1   YYYY-MM-DD/flow-file
 0   flow-file
 1   YYYY/flow-file
```

[2] Source: flow-capture manual page (user@host:man flow-capture).

2 YYYY/YYYY-MM/flow-file
3 YYYY/YYYY-MM/YYYY-MM-DD/flow-file

Having discussed the data gathering techniques dissemination now becomes a major focus of the investigation, as mentioned previously monALISA was the tool of choice. monALISA's ApMon interface affords programmers with a simplified API.

MonALISA and ApMon

ApMon is an Application Programming Interface (API) which interacts with the MonALISA service. ApMon allows any application to send parameterized monitoring information to MonALISA. The data can be sent as UDP datagrams to multiple hosts running the MonALISA service. ApMon has been implemented in C, C++, Java, Perl, Python.

Through the use of ApMon MonALISA can receive parameterized data in name/type/value sets. That ism when transmitting a data point the application specifies the name of the parameter about to be sent, the type of the parameter (string, object, integer, double) and the actual value of the parameter. For our implementation and for reasons which will be discussed later we decided to use a 64 bit real number as the type for all of our data transmissions with ApMon. This type is represented in ApMon by the constant `ApMonConst::XDR_REAL64()`. The ApMon module on the MonALISA service will then receive this data and create any needed fields on its database for new parameters or populate existing fields if a particular parameter name already exists.

The resulting data stored in MonALISA is a set of parameters and the values of those parameters over time. MonALISA then provides a interface by which to view one or multiple parameters in a real-time or historical graph.

FlowTools Integration with ApMon

NetFlow allows for the monitoring of a large number of parameters. For this project we decided to limit the parameters monitor to the following:

- UDP/TCP port destination/source traffic
- IP destination/source traffic
- IP protocol traffic
- IP Next Hop traffic
- AS destination/source traffic
- Prefix destination/source traffic

For most of these parameters we will be monitoring that total traffic in octets over a period of time. The initial period of time was 5 minutes. You can observe in the flow-capture startup script above the parameter –n288 which indicates that we want flow-capture to generate 288 files per day which results in a new file generated very 5 minutes.

Our application will pick the most resent file and use it to retrieve the desired parameters. This will result in parameterized NetFlow data being retrieved by our application reflecting 5 minute averages for each of the monitored parameters. Hence the value of each parameter will represent the total number of octets associated with that particular parameter over a 5 minute period.

We will use flow-stat to collect all of the parameters specified above. Flow-stat allows us to specify the reporting format, the sort type and sort field and whether to use symbolic names whenever possible. Below is the list of the flow-stat formatting values which were used in this project.

• 5	UDP/TCP destination port
• 6	UDP/TCP source port
• 8	Destination IP
• 9	Source IP
• 12	IP protocol
• 16	IP Next Hop
• 19	Source AS
• 20	Destination AS
• 24	Source Prefix
• 25	Destination Prefix

Below is a sample output using flow-stat. We selected formatting option 5 (UDP/TCP destination port) and sorted by octets.

```
$ flow-cat ft-v05.2005-01-19.135207-0500 | flow-stat -f 5 -S 2
#  --- ---- ---- Report Information --- --- ---
#
# Fields:    Total
# Symbols:   Disabled
# Sorting:   Descending Field 2
# Name:      UDP/TCP destination port
#
# Args:      flow-stat -f 5 -S 2
#
#
# port      flows                 octets                packets
#
80          42816                 11274635              82997
25          2877                  3770113               7623
2010        28                    2543195               1747
4662        2095                  2290946               3770
2009        28                    2091493               1413
```

As we can see for this particular 5 minute interval the HTTP port (80) was most heavily used at 11274635 octets roughly 10.75 MB

Similarly flow-filter is used further specify the particular flows to be included in the report. For this project we will be using flow filter to specify which interface to monitor and what direction of traffic we would like to be included in the report. Option –I specifies that only egress traffic from the specified interface number destined outside of the router is to be reported. Option –i indicates that ingress traffic origination from the outside of the router destined to the specified interface is to be reported.

The example below shows traffic destined outside of the router via interface 61 using reporting format 5 (destination port) and sorting by octets.

```
$flow-cat ft-v05.2005-01-19.135207-0500 | flow-filter -I 61 |
flow-stat -f 5 -S 2
#  --- ---- ---- Report Information --- --- ---
#
# Fields:    Total
# Symbols:   Disabled
# Sorting:   Descending Field 2
# Name:      UDP/TCP destination port
#
# Args:      flow-stat -f 5 -S 2
#
#
# port      flows               octets              packets
#
80        7036                2693420             14157
44191     233                 1602211             2456
25        793                 1041730             2394
46878     2                   937500              625
```

The example below shows traffic destined to the router via interface 61 using reporting format 5 and sorting by octets.

```
$flow-cat ft-v05.2005-01-19.135207-0500 | flow-filter -i 61 |
flow-stat -f 5 -S 2
#  --- ---- ---- Report Information --- --- ---
#
# Fields:    Total
# Symbols:   Disabled
# Sorting:   Descending Field 2
# Name:      UDP/TCP destination port
#
# Args:      flow-stat -f 5 -S 2
#
#
# port      flows               octets              packets
#
80        20576               5345776             37613
2010      13                  2459099             1649
2009      10                  2077746             1389
4165      20                  1927749             1380
25        316                 1245240             2429
```

The application written will parse the flow-stat report and generate a name/value pair. The name of the parameter will always be the first column of the flow-stat report. This name will be one of the following:

- A Port number (or name when applicatble)
- An IP address
- A protocol name or number
- An AS number
  or
- A Prefix (in the form of network/mask bit)

The value of the parameter will always be the total number of reported octets for that parameter. This will generate a large number of parameters which will be difficult to sort visually. ApMon provides the ability to classify the parameters submitted to MonALISA. Along with the parameter name/value pairs a cluster and a nodename can be specified. Think of each cluster and nodename as a group. A cluster is a group containing a list of nodenames. Each nodename in turn is a group containing a list of parameters. When submitting data to MonALISA we will define each router interface as a cluster and the description of the parameters being measured as a nodename within that cluster. For example the “NWS Internet2” interface in the Juniper router will be consider a cluster (named Junipter – NWS Internet2) and it will contain the following nodenames: “Egress Destination IP”, “Egress Destination AS”, “Egress Source Ports UDP/TCP”, etc. Lastly these individual nodenames will contain the actual parameter name/values pairs.

In order to give the user the flexibility to specify what values to monitor and how to classify them in MonALISA a configuration files was create for our application. Below is a description of the allowed settings in the configuration file.

Setting Name	Description	Example
cluster_name:	The description of the interface	GSR-StarLight VLAN
node_name:	The description of the parameter being monitored	Ingress Destination Ports
flow_file_directory:	Path to the directory containing the NetFlow files	/home/netflow/flows/gsr/
flow_stat_options:	Options passed directly to flow-stat	-f 5 –S 2 -n
flow_filter_options:	Options passed directly to flow-filter	-i 60
value_param:	The column number from the flow-stat report which will be used as the value for the parameter. Where the first column number is 0.	2

max_params:	The maximum number of name/value pairs which will be processes and transmitted to MonALISA for this nodename.	15
next	End of the nodename's configuration.	

The sample provided above would look like this in the actual configuration file:

```
cluster_name:GSR-StarLight VLAN
node_name:Ingress Destination Ports
flow_file_directory:/home/netflow/flows/gsr/
flow_stat_options:-f 5 -S 2 -n
flow_filter_options:-i 60
value_param:2
max_params:15
next
```

Our application would parse this information and generate the following report:

```
$flow-cat ft-v05.2005-01-19.224516-0500 | flow-filter -i 60 |
flow-stat -f 5 -S 2 -n
#  --- ---- ---- Report Information --- --- ---
#
# Fields:    Total
# Symbols:   Enabled
# Sorting:   Descending Field 2
# Name:      UDP/TCP destination port
#
# Args:      flow-stat -f 5 -S 2 -n
#
#
# port      flows                octets               packets
#
57893       1                    913876               1106
32808       1                    730500               487
6881        52                   210301               283
32920       1                    151132               101
32918       1                    85132                57
1144        3                    84854                61
http        39                   68171                106
1279        1                    67500                45
eDonkey-20  67                   64742                107
3113        2                    63044                49
33498       3                    58552                40
```

The application will parse the generated report and retrieve the parameters name and values as specified in the configuration. For this example the parameter names will be the port numbers and the values will be the octets as specified by the configuration file (value_param:2). The following data would then be sent to MonALISA:

Cluster	Nodename	Parameter Name	Parameter Value
GSR-StartLight VLAN	Ingress Destination Ports	57893	913876
GSR-StartLight VLAN	Ingress Destination Ports	32808	151132
...	...	...	...
GSR-StartLight VLAN	Ingress Destination Ports	http	68171
...	...	...	...

Implementation and Execution

We chose Perl as the development language for its ease of use and its ability to compile at runtime. In the overall the application will check to see if an instance of it is already running and if so quit. The application will then read through the configuration file and gather all the specified parameters. After the parameter name/value pairs for each nodename are retrieved they are transmitted to the MonALISA service hosts as specified in the ApMon.conf file. The application will proceed to each of the cluster/nodenames specified in the configuration file. Cron was used to execute this application every 5 minutes.

MonALISA Customizations

Since this application generated a large number of parameters in a short period of time a number of modifications were made to the ml.properties file for the default farm created. This file is usually located in /home/monalisa/MonaLisa/Service/myFarm/. Below are the parameters which were changed and a description of each change:

```
lia.Monitor.use_epgsqldb=true
lia.Monitor.jdbcDriverString=com.mckoi.JDBCDriver
```

These parameters tell MonALISA to use the embedded PGSQL database as opposed to the MYSQL database. After working with the MonALISA development team they suggested to use the PGSQL database as it was faster and more stable for our configuration.

```
lia.Monitor.Store.TransparentStoreFast.web_writes = 3
```

Specifies that 3 tables will be created for historical data.

```
lia.Monitor.Store.TransparentStoreFast.writer_0.total_time=43200
lia.Monitor.Store.TransparentStoreFast.writer_0.table_name=monitor_s_12
hour
lia.Monitor.Store.TransparentStoreFast.writer_0.writemode=1
```

Defines the first table to maintain data for 12 hours. The data stored in this table written in `writemode` 1 which will write every value received by MonALISA.

```
lia.Monitor.Store.TransparentStoreFast.writer_1.total_time=10800
lia.Monitor.Store.TransparentStoreFast.writer_1.table_name=monitor_s_e_3hour
lia.Monitor.Store.TransparentStoreFast.writer_1.writemode=2
```

Defines the second table which will maintain data for 3 hours. This table will use `writemode` 2 which will store all data included objects which can not be displayed with the MonALISA client.

```
lia.Monitor.Store.TransparentStoreFast.writer_2.total_time=16070400
lia.Monitor.Store.TransparentStoreFast.writer_2.samples=4464
lia.Monitor.Store.TransparentStoreFast.writer_2.table_name=monitor_s_6months
lia.Monitor.Store.TransparentStoreFast.writer_2.writemode=0
```

This last table will store sampled stat for 186 days approximately 6 months. `Writemode` 0 specifies that the data will be sampled.

Use Case – Monitoring Peer Traffic

Having developed the monitoring utility we can best appreciate the practical implications of such a tool with an example. We choose to now take a closer look at a peer, the NWS Internet2 Gigabit Ethernet link that carries all HPC traffic to and from New World Symphony; a post-doctorate institution at Miami Beach that has AMPATH be its upstream provider of Internet2 traffic as well as commodity internet.

First, we are required to start monALISA locally; this requires Java to be installed at the workstation which will run the client side of monALISA. Having accessed the monALISA website and followed the specified instructions to download the Java client we are left with a MonAlisa.jnlp file which runs when executed if Java is properly installed and on the specified user path.

Once monALISA start the screen presented is a global view of all running sites / farms. Having previously specified that the myFarm.conf file on the monALISA directory was set to “test’ we can deduce that the local farm, the monALISA service itself belongs to this group. We must select it in order to view our FIU farm.

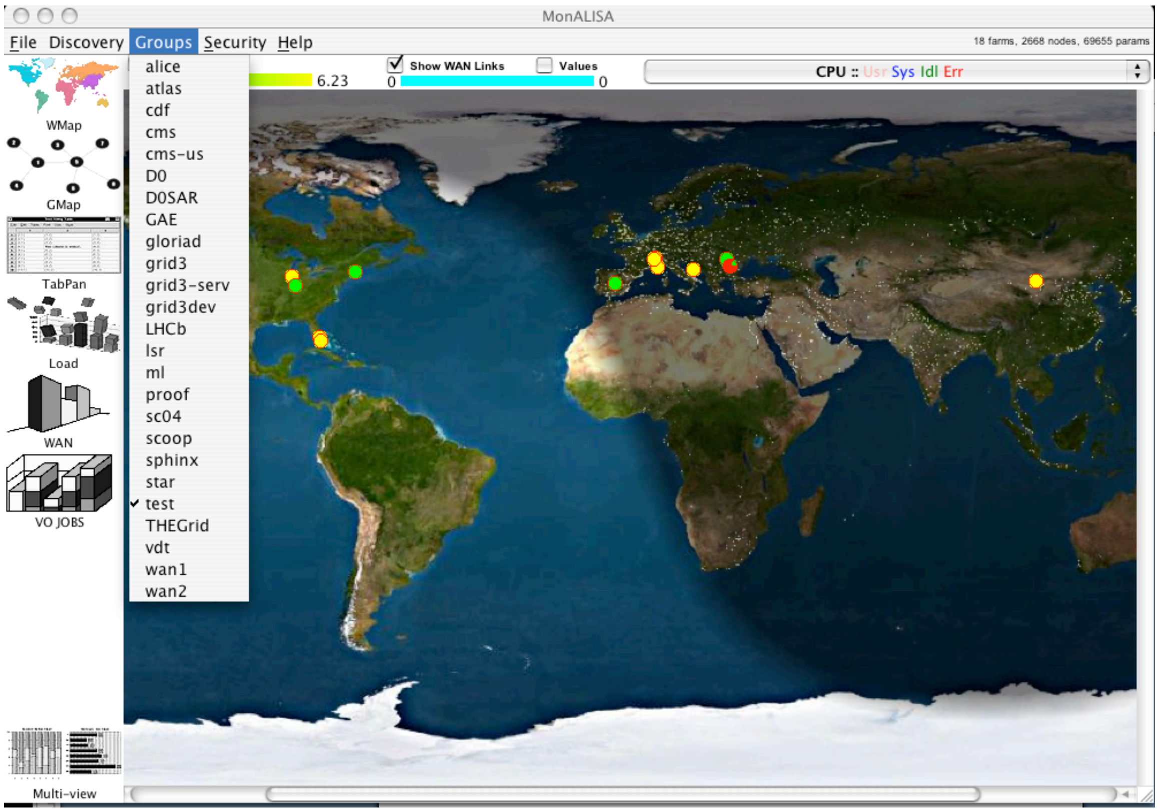
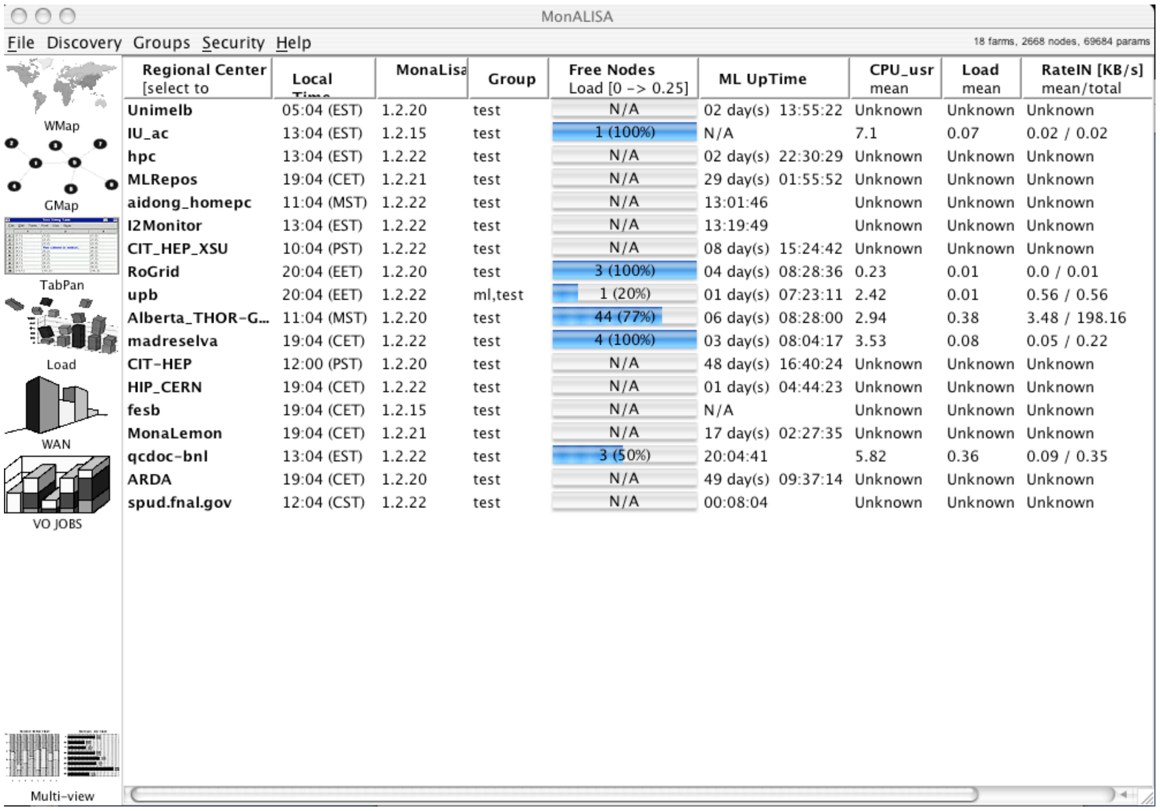


Figure 2 - the monALISA tool startup screen with 'test' group defined.

Having now properly established our working group under monALISA we choose a more detailed view from the left menu presented. The "TabPan" view gives detailed information about the monALISA farm but also serves as the starting point to access parameters that are unique to the site.


MonALISA
File Discovery Groups Security Help
18 farms, 2668 nodes, 69684 params
WMap
GMap
TabPan
Load
WAN
VO JOBS
Multi-view

Regional Center [select to	Local Time	MonaLisa	Group	Free Nodes Load [0 -> 0.25]	ML UpTime	CPU_usr mean	Load mean	RateIN [KB/s] mean/total
Unimelb	05:04 (EST)	1.2.20	test	N/A	02 day(s) 13:55:22	Unknown	Unknown	Unknown
IU_ac	13:04 (EST)	1.2.15	test	1 (100%)	N/A	7.1	0.07	0.02 / 0.02
hpc	13:04 (EST)	1.2.22	test	N/A	02 day(s) 22:30:29	Unknown	Unknown	Unknown
MLRepos	19:04 (CET)	1.2.21	test	N/A	29 day(s) 01:55:52	Unknown	Unknown	Unknown
aidong_homepc	11:04 (MST)	1.2.22	test	N/A	13:01:46	Unknown	Unknown	Unknown
I2Monitor	13:04 (EST)	1.2.22	test	N/A	13:19:49	Unknown	Unknown	Unknown
CIT_HEP_XSU	10:04 (PST)	1.2.22	test	N/A	08 day(s) 15:24:42	Unknown	Unknown	Unknown
RoGrid	20:04 (EET)	1.2.20	test	3 (100%)	04 day(s) 08:28:36	0.23	0.01	0.0 / 0.01
upb	20:04 (EET)	1.2.22	ml,test	1 (20%)	01 day(s) 07:23:11	2.42	0.01	0.56 / 0.56
Alberta_THOR-G...	11:04 (MST)	1.2.20	test	44 (77%)	06 day(s) 08:28:00	2.94	0.38	3.48 / 198.16
madreselva	19:04 (CET)	1.2.22	test	4 (100%)	03 day(s) 08:04:17	3.53	0.08	0.05 / 0.22
CIT-HEP	12:00 (PST)	1.2.20	test	N/A	48 day(s) 16:40:24	Unknown	Unknown	Unknown
HIP_CERN	19:04 (CET)	1.2.22	test	N/A	01 day(s) 04:44:23	Unknown	Unknown	Unknown
fesb	19:04 (CET)	1.2.15	test	N/A	N/A	Unknown	Unknown	Unknown
MonaLemon	19:04 (CET)	1.2.21	test	N/A	17 day(s) 02:27:35	Unknown	Unknown	Unknown
qcdoc-bnl	13:04 (EST)	1.2.22	test	3 (50%)	20:04:41	5.82	0.36	0.09 / 0.35
ARDA	19:04 (CET)	1.2.20	test	N/A	49 day(s) 09:37:14	Unknown	Unknown	Unknown
spud.fnal.gov	12:04 (CST)	1.2.22	test	N/A	00:08:04	Unknown	Unknown	Unknown

Figure 3 - monALISA farm detail

We can see here our farm specific options which we previously configured on the myFarm.conf file. The monALISA process that pertains to our particular study is titled I2Monitor and we are presented with a snapshot of the current state of the system.

Some important parameters to note are:

MonALISA version: 1.2.22
Group: test
ML Uptime (Uptime for the MLD process – monALISA itself)
Rate IN/OUT in KB/s

Choosing the I2Monitor farm, we are then presented with the custom detail which we have chosen to integrate into our analysis.

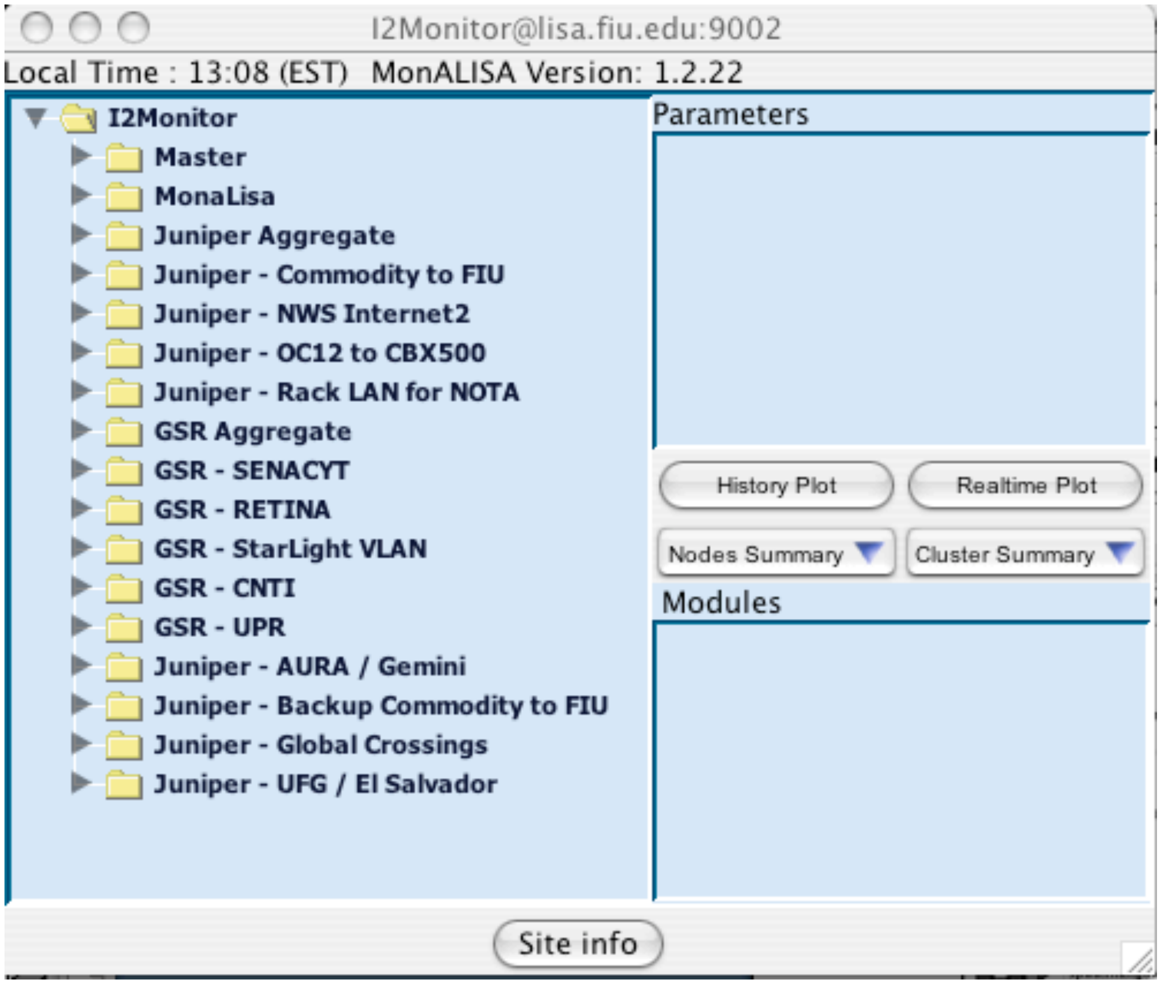
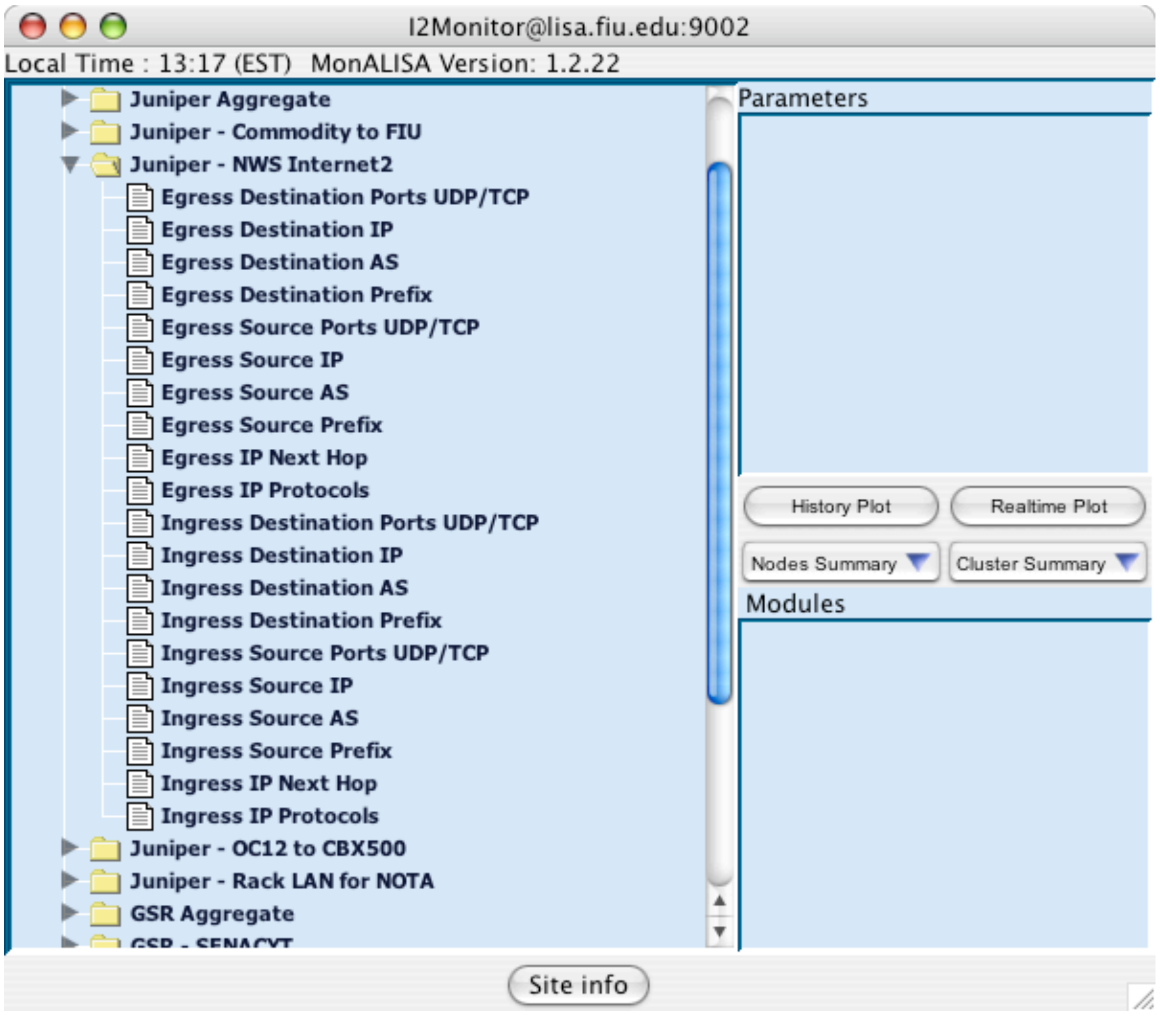


Figure 4 - Farm specific data

It is evident which parameters we have specified to be monitored. As was described earlier, two core routers are monitored along with corresponding interfaces of interest. There is also a Master and MonALISA folder; these folders contain information not of interest to our research but it is important when determining the status or troubleshooting apparent issues with the monALISA farm or service itself.

We can appreciate the simplified approach of having a folder under which are stored the corresponding NetFlow parameters in an already parsed way; ready to be analyzed. We will; in this case analyze the Juniper – NWS Internet2.



I2Monitor@lisa.fiu.edu:9002

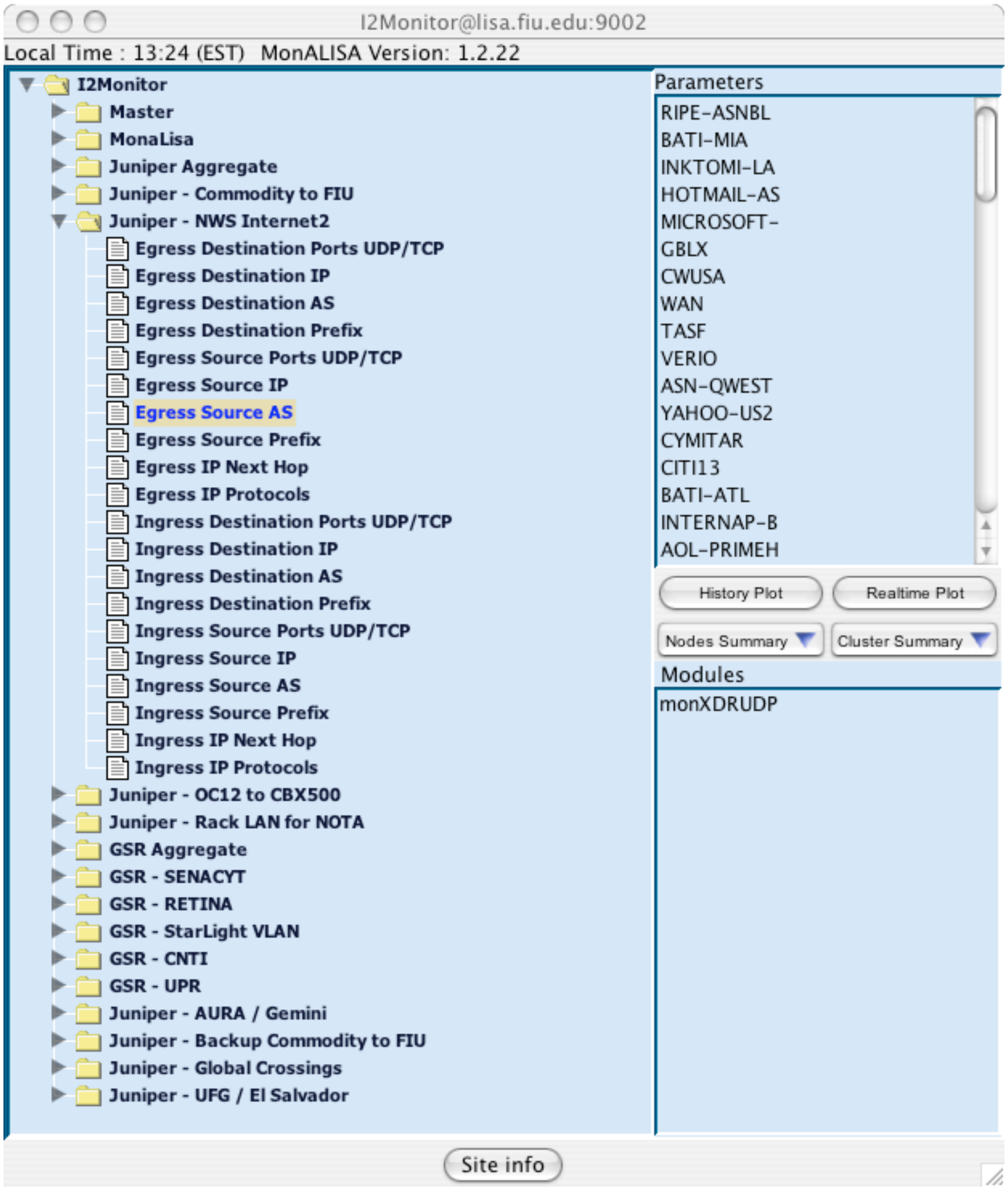
Local Time : 13:17 (EST) MonALISA Version: 1.2.22

- Juniper Aggregate
- Juniper - Commodity to FIU
- Juniper - NWS Internet2
  - Egress Destination Ports UDP/TCP
  - Egress Destination IP
  - Egress Destination AS
  - Egress Destination Prefix
  - Egress Source Ports UDP/TCP
  - Egress Source IP
  - Egress Source AS
  - Egress Source Prefix
  - Egress IP Next Hop
  - Egress IP Protocols
  - Ingress Destination Ports UDP/TCP
  - Ingress Destination IP
  - Ingress Destination AS
  - Ingress Destination Prefix
  - Ingress Source Ports UDP/TCP
  - Ingress Source IP
  - Ingress Source AS
  - Ingress Source Prefix
  - Ingress IP Next Hop
  - Ingress IP Protocols
- Juniper - OC12 to CBX500
- Juniper - Rack LAN for NOTA
- GSR Aggregate

Figure 5 - Parameters studied

As we can see, we are monitoring quite a lot of data. Having made a pre-determined decision about which parameters we would like to monitor we can now view them by simply choosing the intuitive name which we've assigned.

Choosing Egress Source AS we view the current set of stored AS numbers belonging to flows which are leaving our Juniper router; destined to the New World Symphony network.



I2Monitor@lisa.fiu.edu:9002

Local Time : 13:24 (EST) MonALISA Version: 1.2.22

- I2Monitor
  - Master
  - MonaLisa
  - Juniper Aggregate
  - Juniper - Commodity to FIU
  - Juniper - NWS Internet2
    - Egress Destination Ports UDP/TCP
    - Egress Destination IP
    - Egress Destination AS
    - Egress Destination Prefix
    - Egress Source Ports UDP/TCP
    - Egress Source IP
    - Egress Source AS
    - Egress Source Prefix
    - Egress IP Next Hop
    - Egress IP Protocols
    - Ingress Destination Ports UDP/TCP
    - Ingress Destination IP
    - Ingress Destination AS
    - Ingress Destination Prefix
    - Ingress Source Ports UDP/TCP
    - Ingress Source IP
    - Ingress Source AS
    - Ingress Source Prefix
    - Ingress IP Next Hop
    - Ingress IP Protocols
  - Juniper - OC12 to CBX500
  - Juniper - Rack LAN for NOTA
  - GSR Aggregate
  - GSR - SENACYT
  - GSR - RETINA
  - GSR - StarLight VLAN
  - GSR - CNTI
  - GSR - UPR
  - Juniper - AURA / Gemini
  - Juniper - Backup Commodity to FIU
  - Juniper - Global Crossings
  - Juniper - UFG / El Salvador

Parameters

RIPE-ASNBL
BATI-MIA
INKTOMI-LA
HOTMAIL-AS
MICROSOFT-
GBLX
CWUSA
WAN
TASF
VERIO
ASN-QWEST
YAHOO-US2
CYMITAR
CITI13
BATI-ATL
INTERNAP-B
AOL-PRIMEH

Modules

monXDRUDP

Figure 6 - ASNs traversing AMPATH towards NWS

Having been presented with the parameters corresponding to the AS numbers traversing our router destined to NWS we have a clear top-level view of flows of the network and can delve deeper into this data by showing a realtime plot or history plot of the AS data gathered.

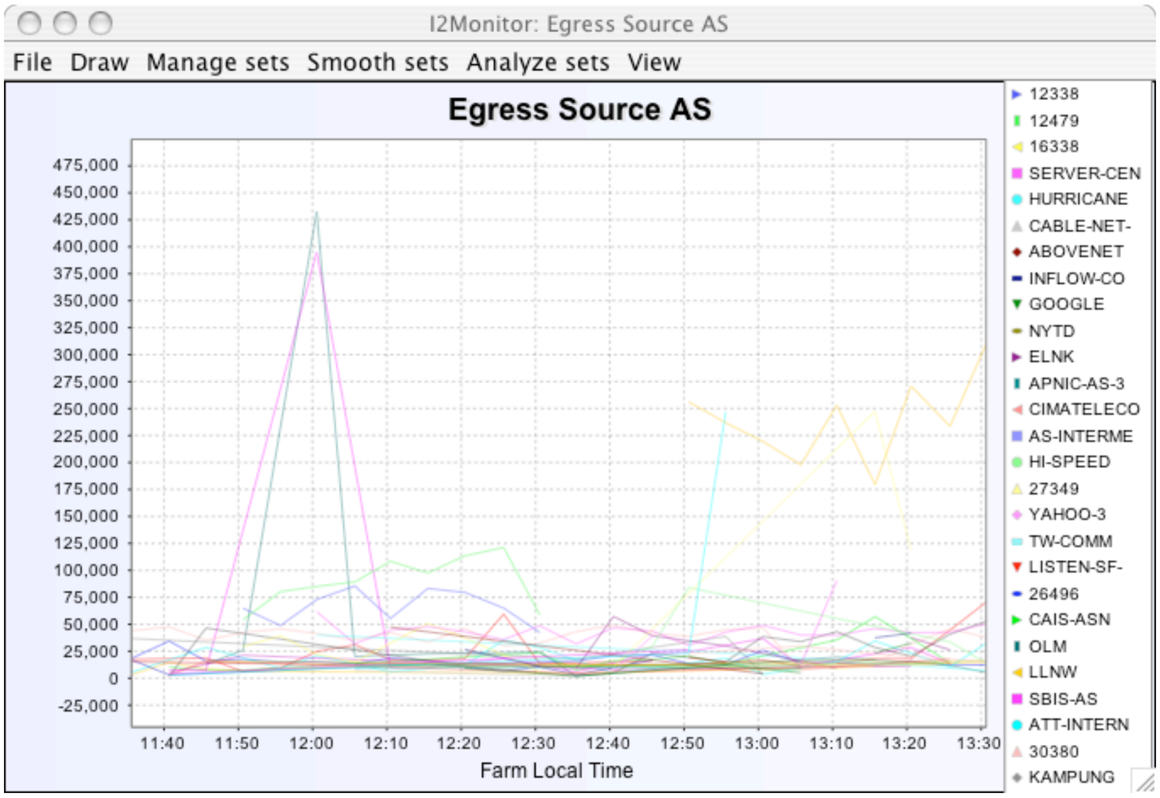
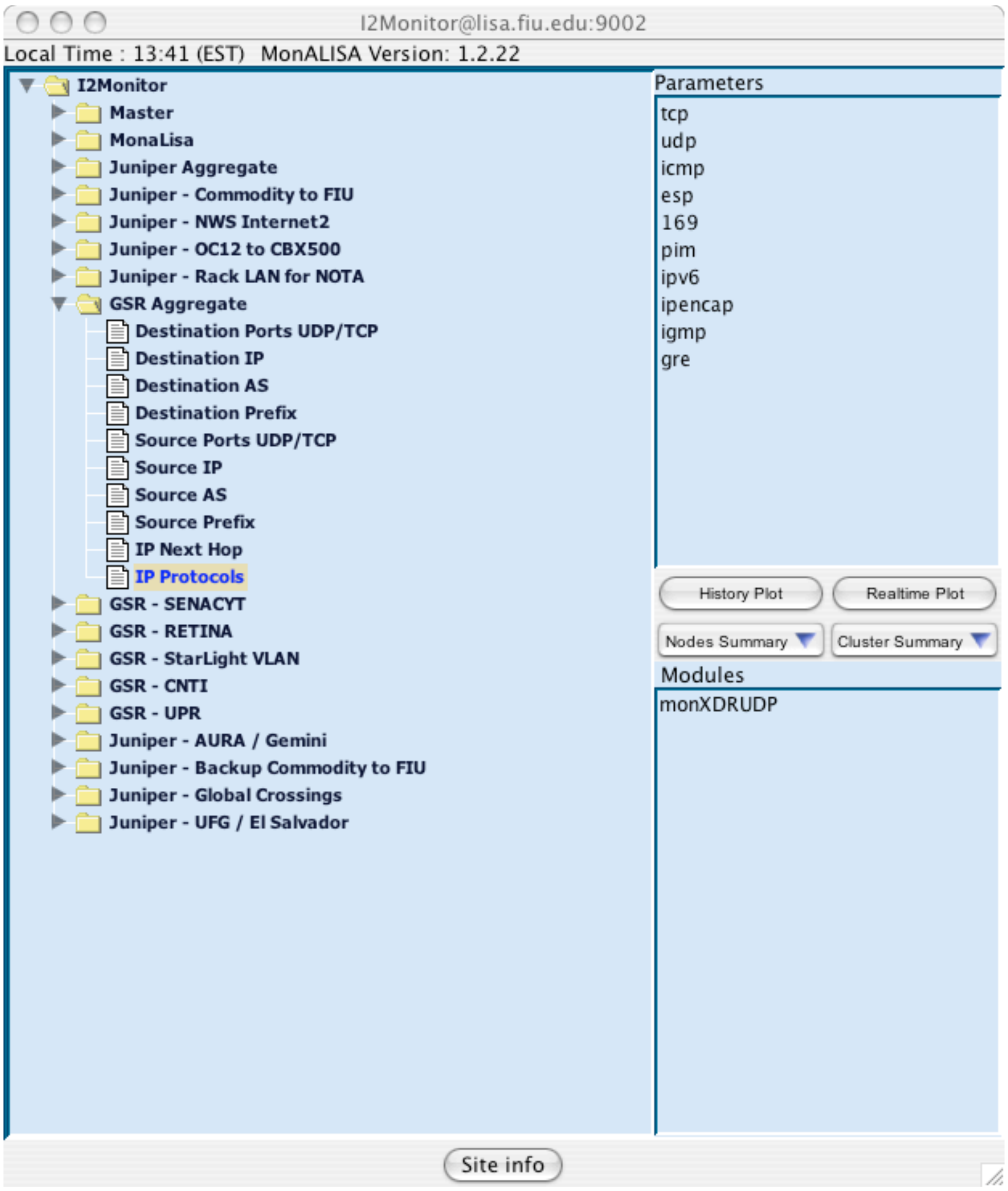


Figure 7 - ASN traffic destined to NWS

By choosing all AS Parameters shown with Ctrl+A and then choosing History plot we can get a default two hour window of ASN data. We previously chose to pass parameter / value pairs which contained the parameter ASN in this case along with data which was chosen to be numbers of octects corresponding to the parameter.

We see from the legend that the maximum utilization of approximately 450 Kbps occurred at around noon by the GBLX ASN, corresponding to AMPATH's commodity provider, Global Crossings.

Other valuable information can be easily extracted from the NetFlow parameters which we've specified; we now take a look at another feature which is helpful in network analysis. We show now the GSR aggregate data but more specifically the protocols which are traversing the GSR router; the main provider of Internet2 to AMPATH IXP peers.



I2Monitor@lisa.fiu.edu:9002

Local Time : 13:41 (EST) MonALISA Version: 1.2.22

- I2Monitor
  - Master
  - MonaLisa
  - Juniper Aggregate
  - Juniper - Commodity to FIU
  - Juniper - NWS Internet2
  - Juniper - OC12 to CBX500
  - Juniper - Rack LAN for NOTA
  - GSR Aggregate
    - Destination Ports UDP/TCP
    - Destination IP
    - Destination AS
    - Destination Prefix
    - Source Ports UDP/TCP
    - Source IP
    - Source AS
    - Source Prefix
    - IP Next Hop
    - IP Protocols
  - GSR - SENACYT
  - GSR - RETINA
  - GSR - StarLight VLAN
  - GSR - CNTI
  - GSR - UPR
  - Juniper - AURA / Gemini
  - Juniper - Backup Commodity to FIU
  - Juniper - Global Crossings
  - Juniper - UFG / El Salvador

Parameters

tcp
udp
icmp
esp
169
pim
ipv6
ipencap
igmp
gre

Modules

monXDRUDP

Figure 8 - Router Aggregate NetFlow data

Having followed the same procedure we now drill down through the GSR Aggregate data to show the IP Protocols which are currently being recorded by the NetFlow process on the router.

A historical look can show the recent or long-term utilization of IPv6; and we can visualize it by means of a history plot which follows:

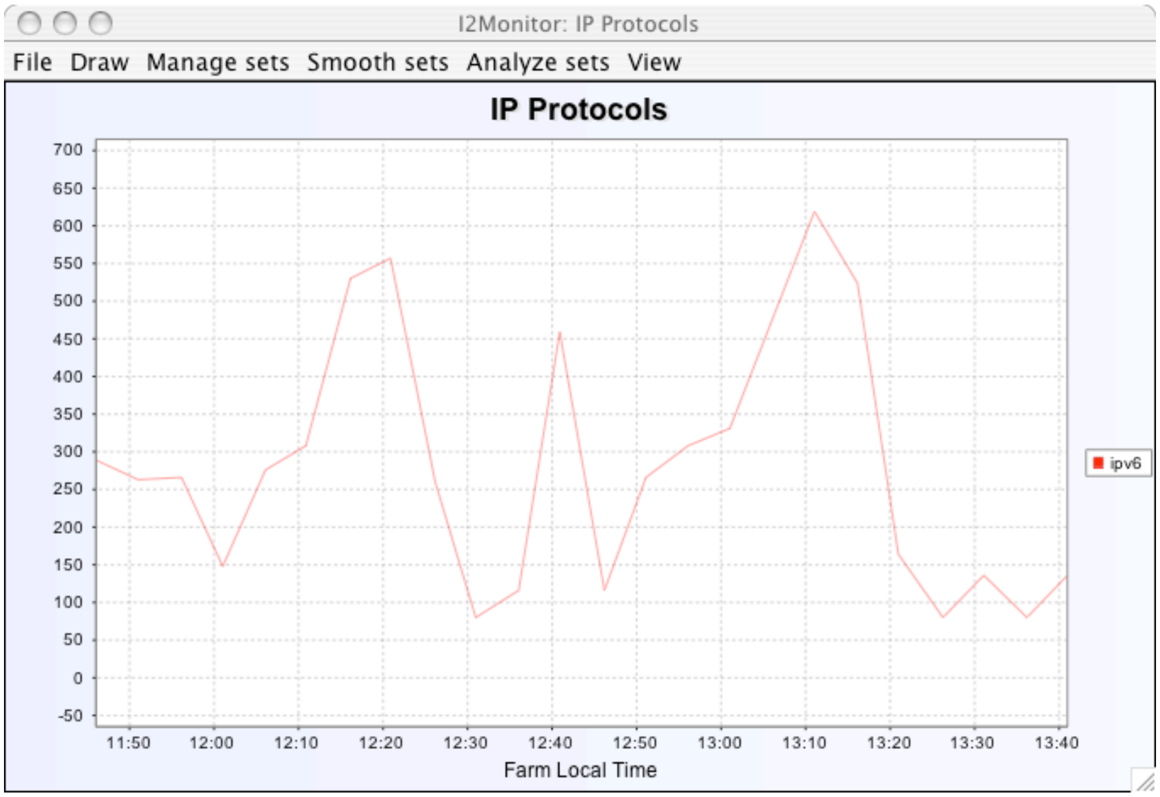


Figure 9 - Two hour snapshot of IPv6 data traversing AMPATH

We can quickly note that IPv6 data has not been a significant load on AMPATH during the time period specified. A longer look at this data can be easily accessed by using the Analyze Sets option from the menu and observing the desired time period.

These are some use cases which can be useful in demonstrating a simple yet powerful tool that integrates flexibility, scalability along with an easy of use for end users. We invite all who read this document to access our monALISA farm by following the procedures specified above and exploring with sufficient time the intricacies of the AMPATH exchange point as well as other peer networks.

Recommendations / Next Steps

NetFlow data contains a rich amount of network information, which has a variety of applications. Through the use of the distributed monitoring environment provided by MonALISA and the reporting flexibility embedded in the FlowTools API it was possible to encapsulate and summarize this data in a cohesive, comprehensive format. In doing so we were able to create dynamic and real time views of the network traffic gaining a clear understanding of its behavior.

Having seen the feasibility and usefulness of such a tool it is important to understand the benefit to the research networking community at large; more specifically in terms of scalability concerns which may arise from the distributed deployment of tools such as this one. During the course of our research a persistent concern from researchers as well as engineers involved with monitoring technologies was the ability to have reliable data from which to draw conclusions. As mentioned previously NetFlow data is sampled and much of the sampling is arbitrary; not a strict requirement that is enforced by either our tool or a standard sampling rate that is given across routing/switching platforms in the industry.

Other questions that were raised during the course of the research conducted included issues with the length of time of a NetFlow capture; exactness of time-stamped flow data as well as parameter specificity and flexibility.

It is these and other issues that the Cisco University Research Program (URP) grant awarded to CIARA seeks to address with the help of a larger window of time and a the support of senior researchers and staff. The aim is to address the monitoring needs and issues raised during this REU research; using the tool developed as a foundation to a more advanced reliable and sophisticated user friendly distributed application. During the course of the next year we intend to be active in the cross-examination of NetFlow data with more detailed packet traces(provided by Passive Monitoring tool developed by NLANR / SDSC). With the Cisco URP we hope to correlate these two sets of results and drawing conclusions as to the reliability of sampled data and also the time-stamp issues which need be addressed.

A great deal of interest has been shown in the research community in this tool and the potential that it demonstrates in becoming a barometer for network utilization as well as the understanding of traffic traversing research links. In particular, the CHEPREO Grid3 cluster; a critical research tool and infrastructure component of the CHEPREO grant will serve as a testbed for the deployment of our newly minted technology; so that we may better understand and report the exact utilization patterns of researchers using the CHEPREO grid technology.