Data Analysis Load Balancer

Design Document: Version: 1.0

Last saved by Chris Small

April 12, 2010

Abstract:

The project is to design a mechanism to load balance network traffic over multiple different links. The load balancer application will utilize an OpenFlow enabled switch to divide traffic by algorithm and send mirrored traffic to multiple Intrusion Detection System (IDS) devices to analyze network traffic.

Components

The load balancer will consist of a number of modular components that should work fairly independent of each other. This should allow reuse of components and allow us to utilize existing software.

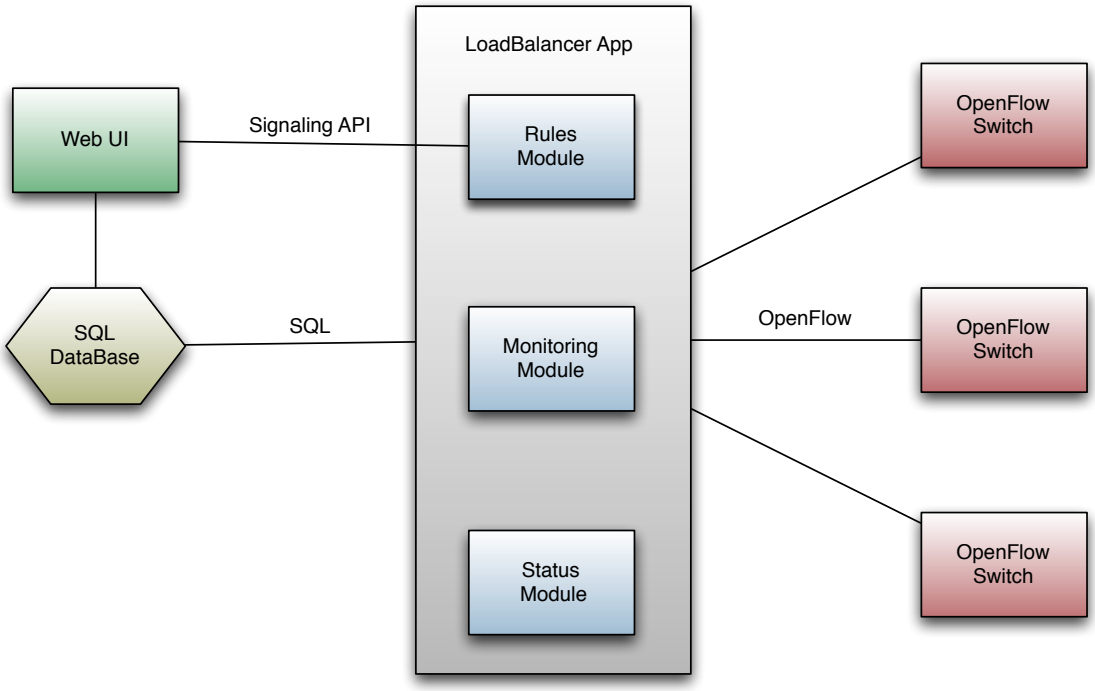
The main components are:

- Web-based User Interface
- Database/Persistent store of rules
- Rules module
- OpenFlow enable switch(es)
- Statistics collection module
- Status monitor module

It is hoped that the status and statistics module can reuse some of the code that has been developed as part of the GENI OpenFlow campus trials, SNAPP and the open-source Nagios monitoring software.

The modular design also should allow reuse of the components and modules by other OpenFlow tools. For example the rules database and module can be used by OpenFlow applications other than Load Balancing (ACL distribution, VM aware migration, etc.)

Component Diagram



Web Based UI

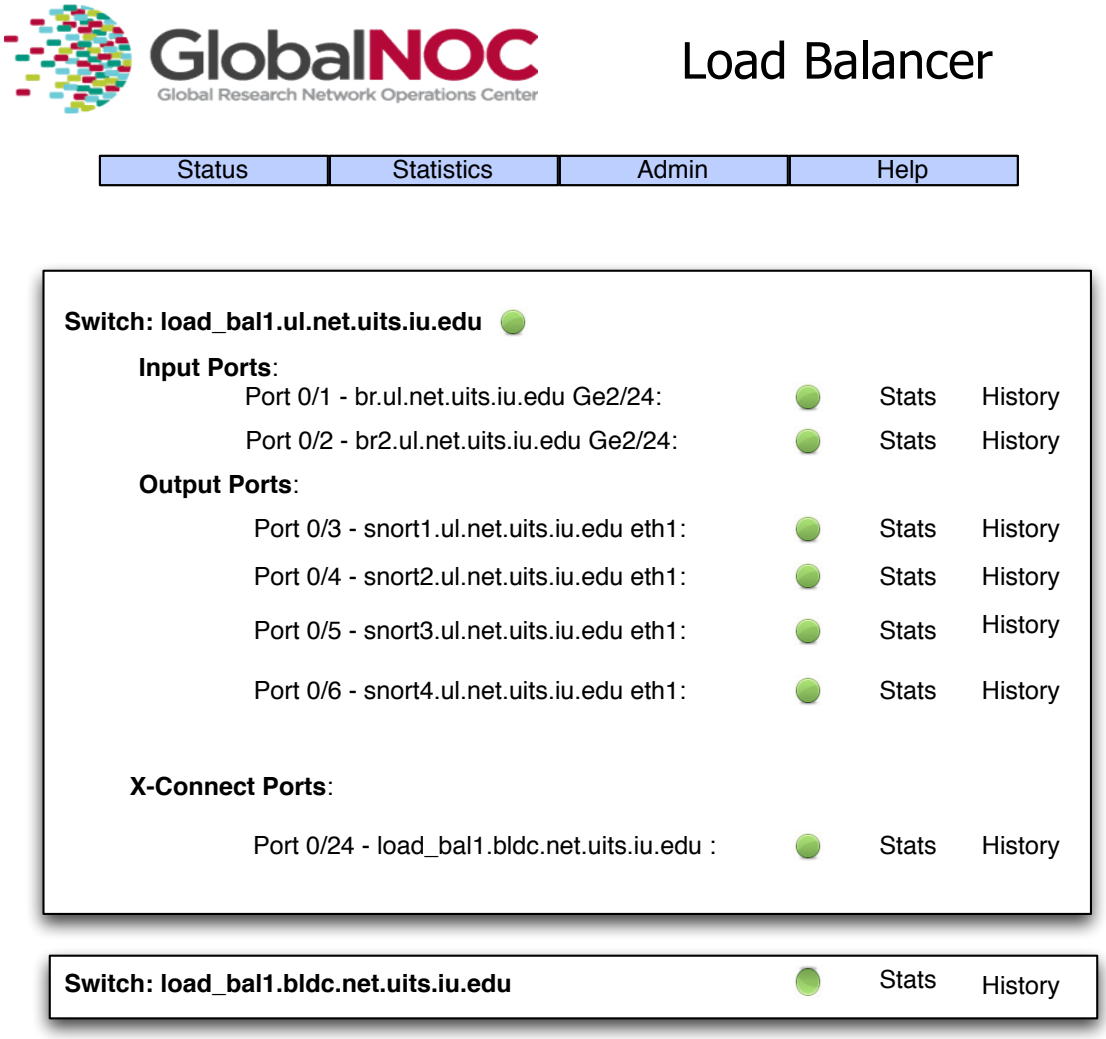
The Web based UI will be the primary interface for users to manipulate the Load Balancer rules. It will also provide a central portal to statistic and status collections.

It will have a number of "tabs" for:

- Status – The status from each port. Ideally this will include application based tests and threshold tests. Threshold tests would detect if there were no traffic flowing over an expected path. Application based tests would detect that there is an application running. For purposes of the data analysis this may be a simple check that make sure the collection program is running.
- Statistics – Group or Switch based stats. May initially just be a link to the SNAPP frontend but a consistent interface would be ideal
- Admin – Interface to define the rules to distribute the traffic
- Help – Help documentation

Groups are user-defined descriptions of a rule that describes the traffic and the input and output ports associated with it. For example, a Group may send all traffic with destination TCP port 80. In the case of source and destination IP addresses you can distinguish a hash size such as a /23 where the load balancers will divide traffic into buckets of /23 and distribute to all ports selected as part of the group.

Mockup of the status page



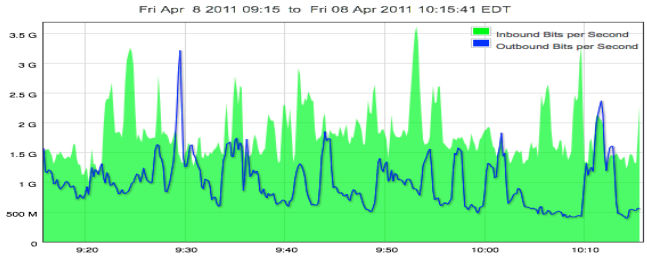
Mockup of the statistics page

GlobalNOC
Global Research Network Operations Center

Load Balancer

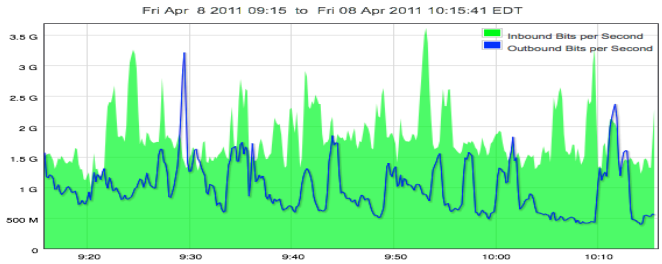
Status	Statistics	Admin	Help
	By Group		
	By Switch		

Switch: load_bal1.ul.net.uits.iu.edu

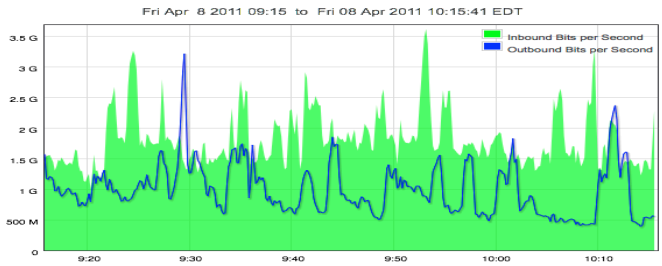


Input Ports:

Total Input:



Port 0/1 - br.ul.net.uits.iu.edu Ge2/24:

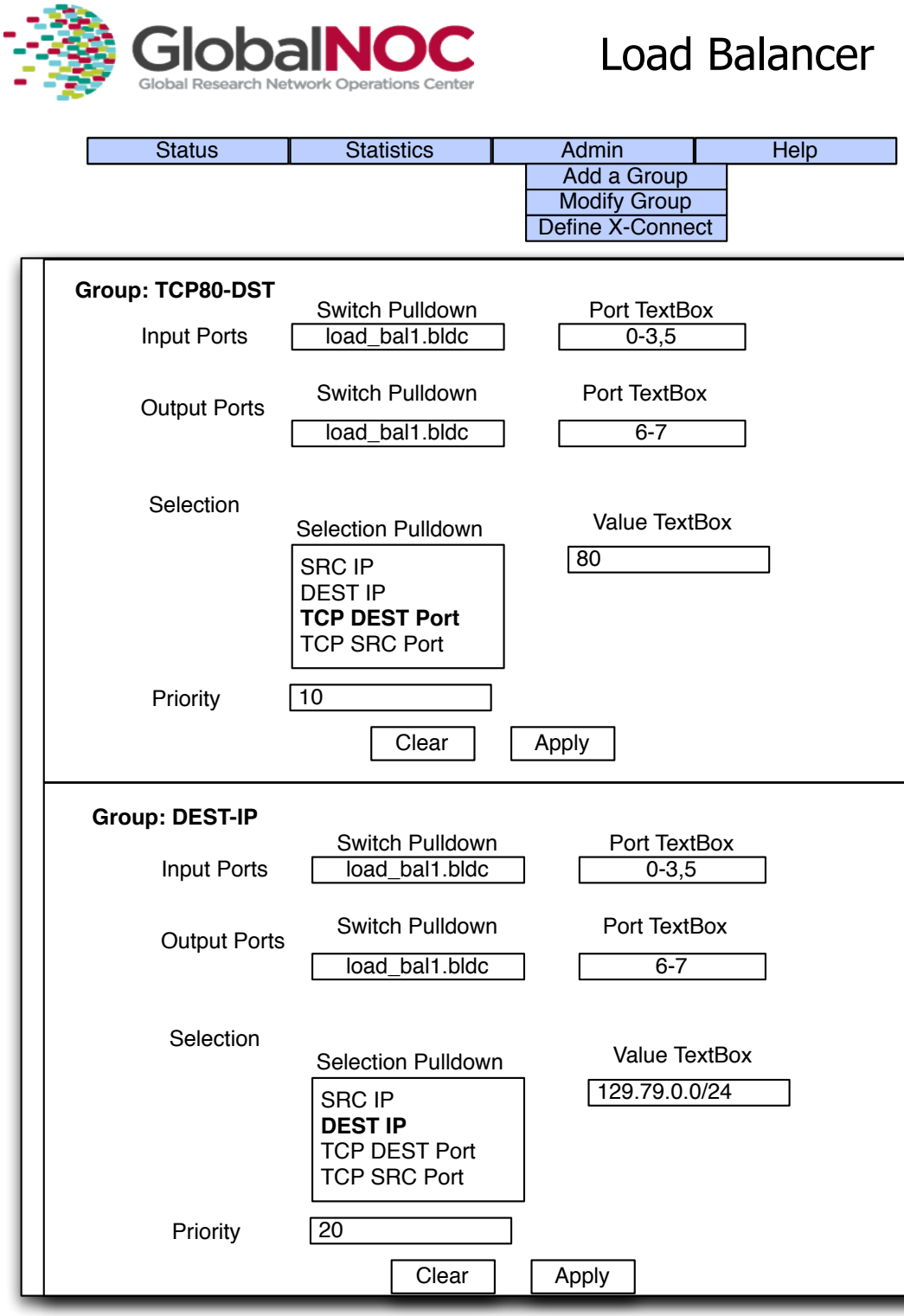


Output Ports:

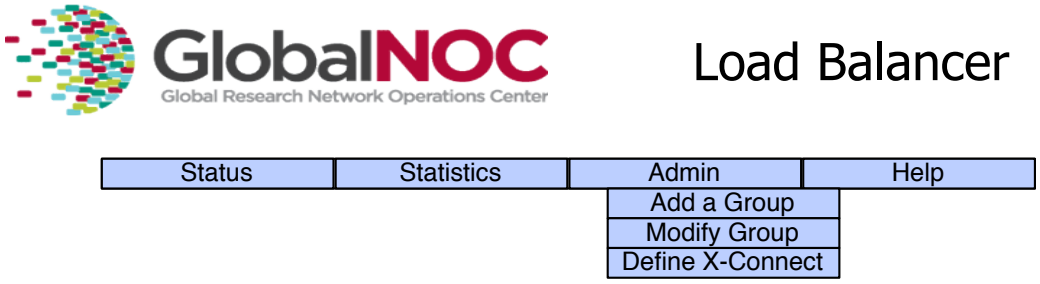
X-Connect Ports:

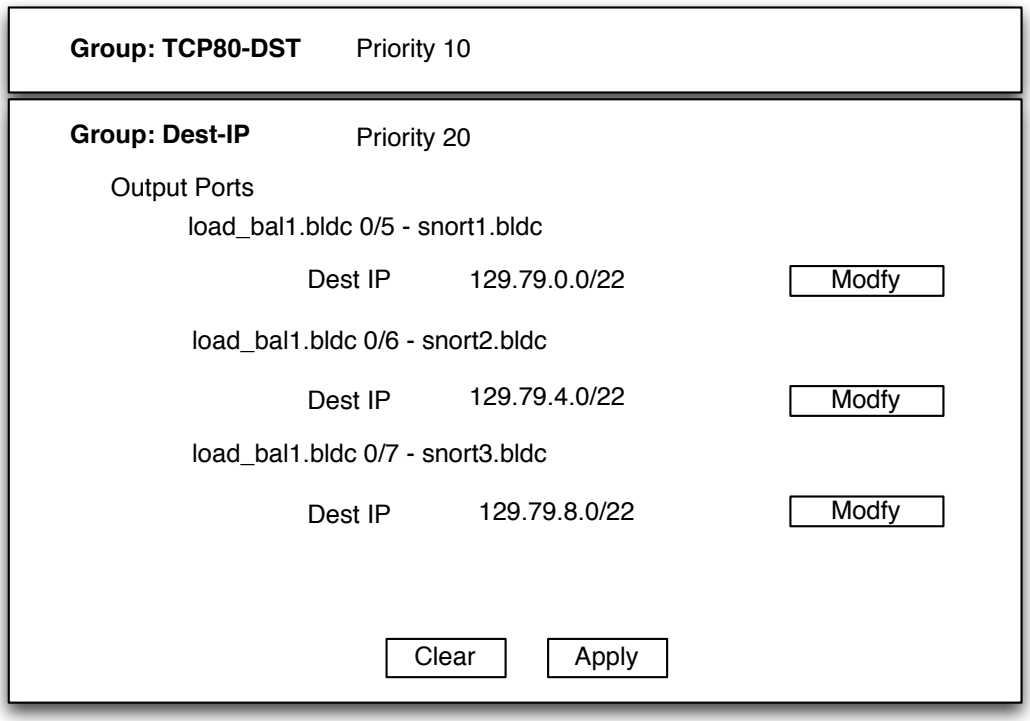
Switch: load_bal1.bldc.net.uits.iu.edu

Mockup of the admin add page



Mockup of the admin mod page





Database

The OpenFlow rules for each switch connected to the load balancer application are stored in a SQL database. This allows for easy manipulation of the rules by the user interface and simplifies the rules module. All the rules module needs to do is synchronize the rules set between the Database and the switch.

The initial tables expected are:

- node - contains switches known to the application
- port – Interfaces on each switch – description, speed
- of_rule – Each rule contained on each switch
- trunk – contains internal routing (i.e. x-connect information) – Initially only assume a L2 connection between each load balancer but eventually may contain multiple intermediate OpenFlow enabled switches in the path

A SQL database may add some additional overhead in the latency of distributing a rule to the switches in the network. However for applications where the reliability and easy of use is more important than the latency in inserting a rule a SQL database has many advantages. It will make it much easier to have check pointing and recovery. For a load balancer where the rules are pre-populated and long-lived high speed population of rules is not essential.

Load Balancing Application

The load balancer application consists of 3 parts, a rule generator, a statistics collector and a status monitor. All three modules will communicate with the switches over a shared connection. A Flowvisor[1] like mechanism with some virtualization of each module is an option but the initial work will look at a shared connection inside the application.

Rules module

The rule generator simply fetches a set of rules from the SQL database and synchronizes the rules with a remote switch. A signaling API will be provided for applications to signal the rules module to synchronize a single switch or set of rules.

Statistics

The statistics module will collect aggregate and port level statistics over the OpenFlow interface. There is the possibility it may utilize out of band mechanisms (SNMP, RANCID) to obtain or modify some information not available through the OpenFlow protocol such as the description for each port configured on the switch.

[1] Flowvisor - http://www.openflow.org/wk/index.php/FlowVisor

The IU Measurement Manager[2] statistic collection toll is expected to provide much of the code for the statistics module. It utilizes the open-source SNAPP collection infrastructure to store and provide an interface to data collected.

An example of the SNAPP OpenFlow interface is located at:

http://gmoc-db.grnoc.iu.edu/nlr-of

Monitoring

One requirement of the load balancer is to detect when there is an issue with either the input of data or with one of the data analysis nodes. The problems would be visible through the Load Balancer UI. These alarms should also be able to be passed to external programs.

The monitoring module will utilize the Nagios software tool. The Load Balancer SQL database should provide the necessary information for an automatic configuration on the Nagios configuration. This module will also leverage some existing work in the Measurement Manager application.

Initially the tests will check for:

- Thresholds on input and output traffic - Test if utilization is to high or to low
- Interface status test
- Test that collection application (bro, snort) is working

The Nagios plugin mechanism should provide plenty of flexibility for adding different application tests for application other than a data analysis app. For example, a http check could be added for web load balancing in the future.

It is also expected that the alarms will feed into set of actions that will occur if an alarm is seen. If a collection node goes down traffic could be redirected to a hot spare or the rules rewritten to redirect the traffic. This may be handled by the monitoring app utilizing some of the functions inside Nagios or a special code written. This will be controlled by configurations generated by the user UI.

Switches

In order to build the data analysis infrastructure we need to have a set of requirement s for the Switch and firmware chosen MUST:

- Support at least the OpenFlow 1.0 spec
- Allow for hardware matching of IP src and dst wildcard rules and TCP port
- Have at least 12 10Gb/sec ports
- Support SSL keys for authentication of controllers and switches

[2] Measurement Manager - http://groups.geni.net/geni/attachment/wiki/OFIU-GEC10-status/IU_OF_GEC10_poster.pdf

In addition having these features it would be advantageous that a switch allowed multicasting of frames to multiple physical ports.

Having group based hash support such as OpenFlow 1.1 option could improve the ability of the distribute traffic more easily and minimize the number of rules.

Exports

The load balancer application should export a copy of the how it distributes software to assist the data analysis software. Format and mechanism is to be determined.

Security

With the ability to redirect packet capture traffic control on the rules is very important.

The load balancing application MUST:

- Authenticate switches and controllers to each other using the OpenFlow security mechanisms
- Ensure adequate authentication on the web interface

Testing

The load balancing application needs to send a simulated or captured flow and insure traffic is complete and distributed over multiple ports. Functional tests of all major functions in the UI function properly.

Future Ideas and Work

Currently there isn't any mechanism to have affinity of a connection based on previous connections. Since data analysis is only examine the packets it is outside the scope of this project but would be desirable for a general load balancing solution. Some academic work has been done on load balancing using OpenFlow that may be utilized.[3]

The central concept of the load balancer application is to be modular to 1) allow for independent modules of modules 2) reuse of modules include existing code. Module connections should allow other applications to be easily added to the infrastructure to combine functions. For example, a Layer 2 topology application can be incorporated with the load balancing application to allow load balancing over a

[3] OpenFlow-Based Server Load Balancing Gone Wild, R. Wang, D. Butnariu, and J. Rexford, *Princeton University Hot-ICE 2011*
http://www.usenix.org/event/hotice11/tech/full_papers/Wang_Richard.pdf

large set of interconnected switches, such as in a data center or across wide area networks. Similar integration could happen with OpenFlow applications such as VM migration.

Load balancer infrastructure