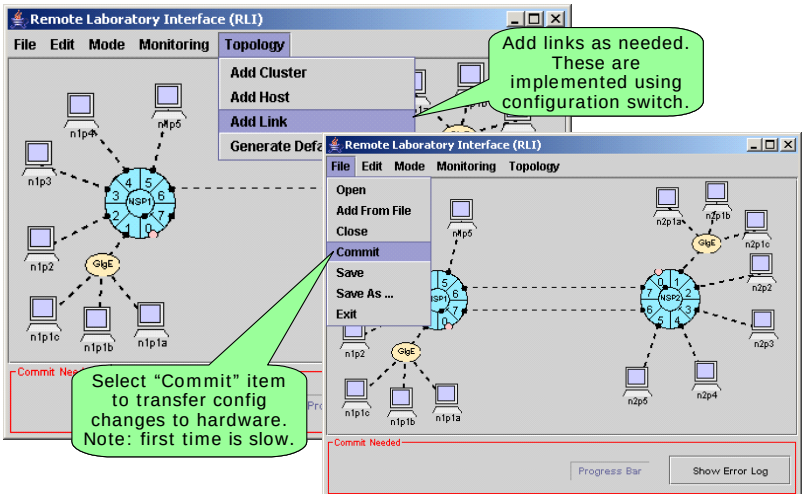
The Open Network Laboratory
a resource for networking research[1]

John Dehart, Fred Kuhns, Jyoti Parwatikar, Jonathan Turner and Ken Wong
Washington University, St. Louis
{jdd,fredk,jp,jst,kenw}@wustl.edu

Abstract: The Open Network Laboratory is a resource designed to enable experimental evaluation of advanced networking concepts in a realistic operating environment. The laboratory is built around a set of open-source, extensible, high performance routers, which can be accessed by remote users through a Remote Laboratory Interface (RLI). The RLI allows users to configure their own private testbed within the ONL infrastructure, run applications and monitor those running applications using built-in data gathering mechanisms. Support for data visualization and real-time remote display is provided. The RLI allows users to install and configure software *plugins* that run in the routers' embedded processors. The routers are architecturally similar to high performance commercial routers, enabling researchers to evaluate their ideas in a more realistic context than that provided by PC-based routers. This demonstration will provide an introduction to ONL, with an emphasis on how users can quickly get started setting up and running their own experiments.

1.*Building and configuring a network*

The Remote Lab Interface allows remote users to create and configure a network topology through an intuitive graphical user interface. The figure below shows a typical session. The basic building block is a *cluster* which consists of a router together with several directly connected hosts and a gigabit Ethernet subnet with three attached hosts. The router in a cluster starts with four uncommitted ports that can be used either to connect to other routers or to additional hosts. The figure shows a configuration using two clusters, with a pair of links joining the two routers.



Once a base configuration has been specified, a user can request allocation of the testbed resources needed to implement that configuration by selecting the **Commit** option from the **File** menu. Assuming the required resources are available, they will be allocated and initialized (this initialization includes rebooting of all the hosts, loading the FPGA bit files that implement the routers' basic packet handling features and rebooting the embedded processors at each router port).

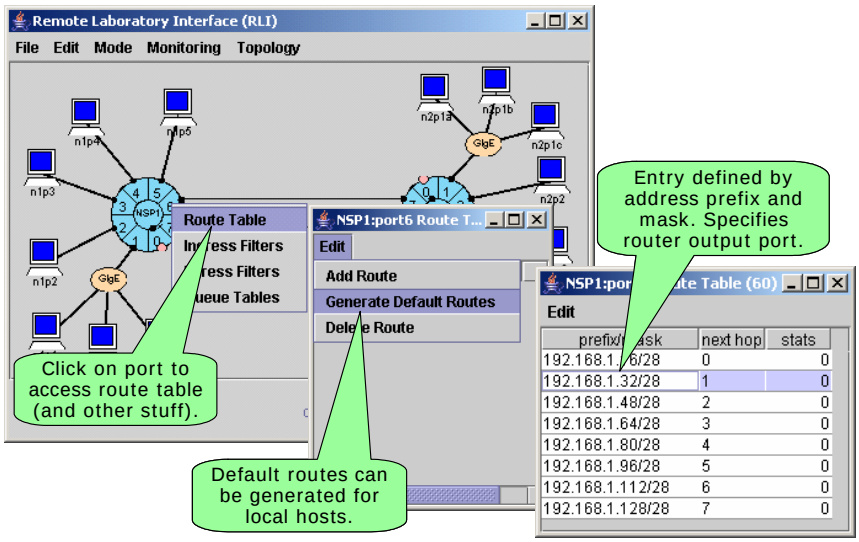
2.*Configuring routes, filters and queues*

Each port of the router has a routing table that can be configured through the RLI. The RLI will generate a set of default routes on request, allowing any host in the configured network to

[1] This project is supported by the National Science Foundation, grant #230826.

communicate with any other host. These routes can then be modified or deleted and new routes can be added. To transfer configuration changes to the hardware, the user *commits* the changes to the hardware by selecting **Commit** from the **File** menu.

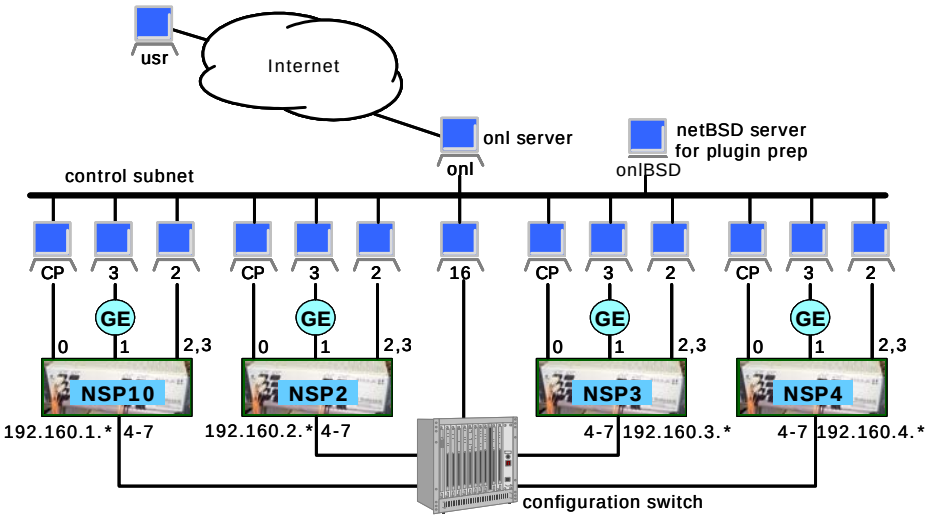
Each router port also includes an exact match flow table and a general filter table. A flow table entry will match packets with specified source and destination addresses, protocol and port numbers (port numbers are checked only for TCP and UDP). A general match filter allows address prefixes, protocol wild cards and port number ranges as well. The number of general match filters is limited to 32 per port. On the egress side of the router, flow table entries and general filter table entries can be mapped to a specified link queue. Link queues are serviced by a weighted DRR packet scheduler, with configurable weights and queue discard thresholds.



3. *ONL testbed equipment*

The ONL testbed includes four experimental routers plus 40 general purpose PCs that serve as hosts in the experimental networks and control processors for the routers. A diagram of the testbed equipment is shown below. Users interact with the testbed through the RLI, which is implemented as a standalone Java application. The RLI communicates with the testbed through an SSH connection to the ONL server, which relays control messages to other testbed components.

The experimental routers are referred to as Network Services Processors (NSP). Each NSP has a total of eight ports, three of which are connected directly to hosts (one of which acts as the NSP's Control Processor (CP)) and one of which connects to a gigabit Ethernet switch with three more hosts. The remaining four ports connect to a Configuration Switch, which is used to configure experimental topologies linking NSPs to one another or to additional hosts.

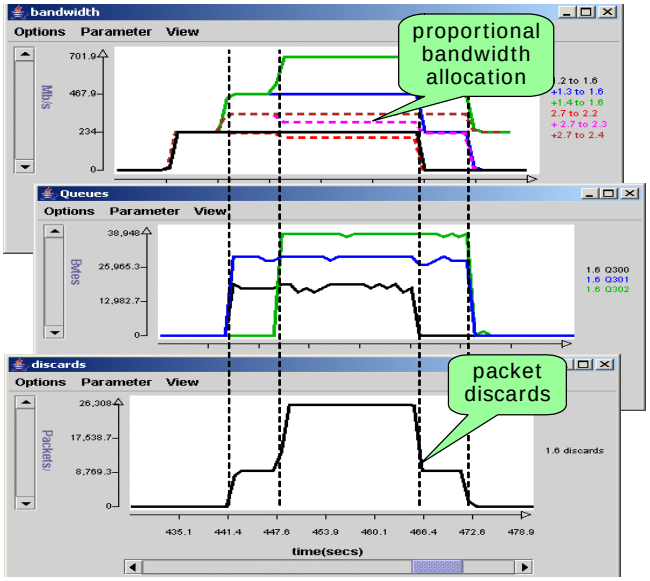


4. *Monitoring traffic flows in the network*

The RLI includes extensive support for traffic monitoring and data visualization, enabling users to get a clear picture of what is happening "below the surface" and allowing them to deliver compelling demonstrations. The figure at right shows real-time displays obtained for three UDP

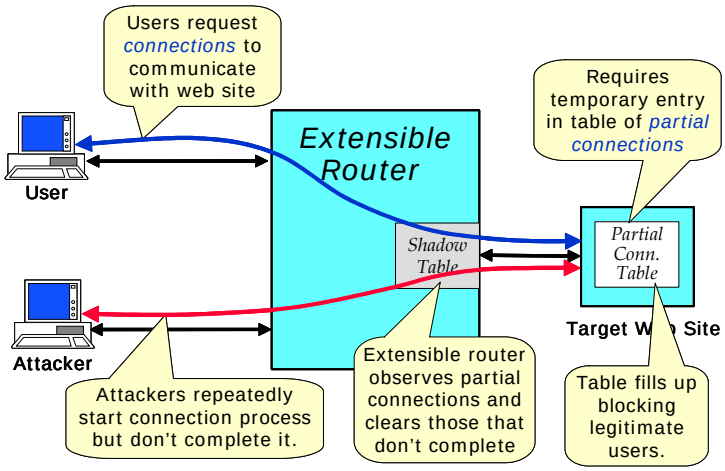
flows passing through a bottleneck link. The top chart shows the incremental bandwidth usage by the three flows (solid lines for bandwidth entering the bottleneck link, dashed lines for bandwidth exiting the bottleneck link), the middle chart shows the number of bytes queued for each flow and the bottom chart shows the rate at which packets are discarded at the bottleneck link. The data sources used to produce these charts are implemented in the NSP hardware. The monitoring subsystem polls the hardware counters periodically to obtain the raw data used to produce the displays.

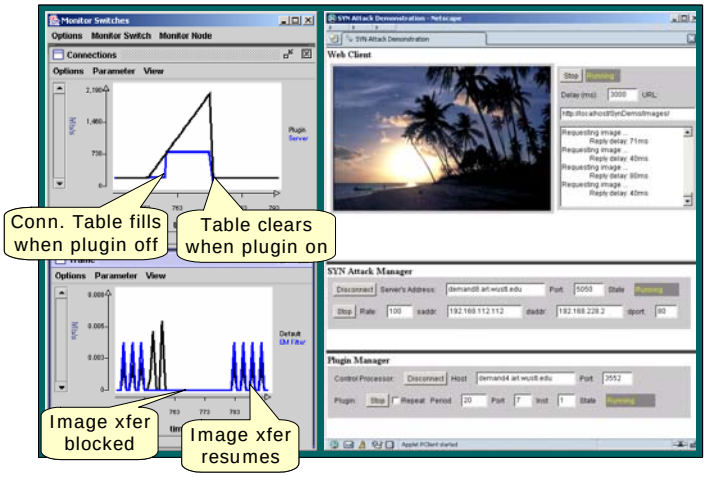
The NSP hardware provides a wide variety of different counters that can be monitored and displayed. In addition, it supports data monitoring in the connected hosts and within the NSPs' embedded processors.



5. *Extending router functionality with plugins*

Each port of every NSP includes an embedded processor subsystem, called the Smart Port Card, that can host software plugins for implementing specialized packet processing functions. The figure below shows an example application used to protect a vulnerable web site from a simple denial of service attack. In this scenario, the attacker creates partially open TCP connections, but never completes the connections, causing the table of partial connections in the target web site to fill up, blocking new connection requests from legitimate users. The SPC in the router port connecting to the target hosts a software plugin that maintains a shadow table of partially open connections. When it detects connections that have remained in the partially open state for too long, it clears them by sending a packet that appears to be from the connection's originator, clearing the connection. In the demonstration, the attack mitigation code is periodically enabled and disabled, allowing us to observe a rapid increase in the number of entries in both the target's table of partial connections and the shadow table. The bandwidth display at bottom also shows that image transfers are blocked while attack mitigation is disabled (each peak corresponds to an image transfer).





For more information see: www.onl.arl.wustl.edu.