Remote Network Labs: An On-Demand Network Cloud for Configuration Testing

Huan Liu

Accenture Technology Labs
50 W. San Fernando St., Suite 1200
San Jose, CA 95113
huan.liu@accenture.com

Dan Orban

Accenture Technology Labs
50 W. San Fernando St., Suite 1200
San Jose, CA 95113
dan.orban@accenture.com

ABSTRACT

Network equipment is difficult to configure correctly. To minimize configuration errors, network administrators typically build a smaller scale test lab replicating the production network and test out their configuration changes before rolling out the changes to production. Unfortunately, building a test lab is expensive and the test equipment is rarely utilized. In this paper, we present Remote Network Labs, which is aimed at leveraging the expensive network equipment more efficiently and reducing the cost of building a test lab. Similar to a server cloud such as Amazon EC2, a user could request network equipment remotely and connect them through a GUI or web services interface. The network equipment is geographically distributed, allowing us to reuse test equipment anywhere. Beyond saving costs, Remote Network Labs brings about many additional benefits, including the ability to fully automate network configuration testing.

Categories and Subject Descriptors

C.2.3 [Network Operations]: Network Management; C.2.1 [Computer Communication Networks]: Network Architecture and Design

General Terms

Design, Experimentation, Management

Keywords

Network Cloud, Test Labs, IP Tunnels, Configuration Testing

1. INTRODUCTION

It is well known that networks are hard to configure correctly. It is reported that most network outages are caused by operator errors in configuration, rather than equipment failures [6][14]. A recent study [19] on firewalls, a particular

type of router used frequently in enterprise networks, shows that there are on average 7.17 to 9.63 configuration errors, even in firewalls used in production. Another study [12] concludes that 3 out of 4 BGP prefix advertisements are results of misconfiguration.

This difficulty can be attributed to several causes. First, routers traditionally have a very primitive CLI (Command Line Interface) human interface. It is not only easy to make mistakes, but it is also easy to lose track of the global configuration. Second, there are many firmware versions for a router (Cisco is well known for the many versions of IOS), and each behaves slightly different. A design may work on paper, but it may not on routers with a particular version of the firmware. Third, configuration is done locally on one router at a time with no knowledge of the overall network, and a simple change at one router may have undesired interactions with the rest of the network.

To make sure the network configuration could be as correct as possible, network administrators typically have to validate the configuration changes before rolling them out to the production network. There are several approaches to validate the configuration change.

One approach is to use a router simulator to simulate the configuration. Commercial tools, such as those from Router-Sim and OPNET, have simulation models for popular router platforms. The first drawback of a simulator is that the simulation model cannot capture all aspects of a real router, and sometimes it cannot even simulate the complete command set [17]. In addition, router vendors, such as Cisco, frequently release special versions of their firmware (e.g., IOS) for specific customers to fix their reported bugs. The simulation model would not be able to capture all those subtle details. The second drawback is that one can only create a limited number of simulation models, yet, there are a large number of network devices. It is difficult to create one simulation model for each possible network device.

Another approach is to use a router emulator such as Dynamips[7]. Dynamips acts as a hypervisor and it can boot up any Cisco IOS for a set of router platforms. Even though it can accurately capture the behavior of the control plane software, it still has a couple of limitations. First, the interface modules are simulated and only a limited set of interface modules are supported. Second, the emulator only supports a limited set of Cisco routers, a much smaller portion of all available network devices.

Because of the limitations on simulators and emulators, most network administrators take a different approach. They build a smaller scale test network that mimics the real pro-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WREN'09, August 21, 2009, Barcelona, Spain.

Copyright 2009 ACM 978-1-60558-443-0/09/08 ...\$10.00.

duction network as close as possible. When they make a configuration change, they first test out the changes in the test lab to make sure that everything works properly, then they roll out the changes to the production network.

Using a test lab has its drawbacks. It is very expensive to build. High end enterprise routers could run up to millions of dollars, yet these test routers are only used during testing and they are not used again until the configuration has to change again. In addition, it is time consuming to wire up routers, since someone has to physically feed cables through the rack space. It is also very easy to make mistakes such that the physical network ends up to be different from what is designed on paper.

Because the routers have to be physically co-located, it is very difficult to share the test equipment. Accenture builds enterprise networks for a large number of clients, but we cannot share the test equipment across projects because moving them from project to project is both time consuming and costly.

In this paper, we present Remote Network Labs (RNL), which is designed to solve the problems associated with building a physical test lab. It consists of a set of network equipment that are geographically distributed throughout the Internet, possibly behind firewalls. In addition, it presents a web user interface and web services APIs (under development) which allow end users (e.g., a network administrator) to access remotely. RNL is essentially a network cloud, because similar to a server cloud, such as Amazon EC2, end users can request routers on-demand to construct a test lab. We use the term “router” loosely in this paper, it can refer to any network equipment, such as a firewall or a traffic generator, that is part of a test lab.

2. THE DESIGN AND IMPLEMENTATION OF REMOTE NETWORK LABS

RNL’s architecture is shown in Fig. 1. It consists of a collection of routers that are scattered across the world. Even though some may be co-located in the same physical lab space, there is no physical constraint on where the routers are as long as they can be connected to the Internet.

There is a general purpose PC sitting in front of every router. There are many network interface adapters on this PC and each port on the router is connected to a dedicated network interface adapter. The PC is responsible for capturing all packets from the corresponding router port and sending all packets destined to the port. The PC is also responsible for communicating with the back-end server, e.g., netlabs.accenture.com. The communication includes reporting on what routers are available, what ports are available, as well as sending/receiving packets to/from the back-end server and the router ports. The PC always initiates the connection to the back-end server, so that, even if the routers are sitting behind a corporate firewall, they can still be connected to the RNL.

The central back-end server at netlabs.accenture.com is responsible for coordinating all communications in RNL. It has two roles: web server and route server. The web server is responsible for communicating with a user’s browser during a design session where the user specifies the network topology and router configurations. The route server is responsible for routing packets from one router port to another based on the user design.

Figure 1: Remote Network Labs’ architecture

RNL is aimed at configuration testing, whereas other experimental network facilities are aimed at new routing software protocol and algorithm evaluation. Because of its unique goal, we have adopted a very different architecture. There are several key features of RNL which set it apart.

Real Routers: RNL uses real routers so that the users could perform realistic configuration testing. RNL even allows users to program different versions of the firmware onto test equipment, for example, to test the behavior under the many different versions of IOS. Facilities such as Emulab[18], PlanetLab[15], ONL[11] and VINI[3] cannot run arbitrary router software, thus cannot simulate a commercial router’s behavior.

Using real routers sets RNL apart from other experimental facilities, such as Emulab, ONL, planetLab and VINI, which use programmable router nodes to facilitate routing algorithm and protocol evaluation. Using real routers allows us to exactly mimic the production network, accurately reproducing the behavior of an exact router platform with any specific router firmware.

Distributed Network Equipment: RNL is aimed at testing configuration in an enterprise network. Because there are many types of enterprise routers and because we need a few routers of each type to construct a meaningful lab, the cost of purchasing an exhaustive list of equipment is prohibitive. In addition, enterprise routing equipment evolves quickly, thus it is also costly to keep the lab up to date. In order to be cost effective and still be useful, we have adopted a distributed architecture. Although the bulk of the test equipment (i.e., the commonly used) is located in a couple of central data centers, the users could also setup their own equipment at their site. This distributed architecture allows users to leverage the common equipment in a central location, yet still have the flexibility to accommodate special needs.

The distributed nature of RNL sets it apart from a central

facility such as WAIL[2]. A single central facility could limit the number of routers available. For example, WAIL has 50 IP routers and switches and 100 end hosts. Whereas, we envision RNL to evolve to include hundreds of routers.

Virtual connection: Having a large repository of routers is not enough. We have to allow them to be flexibly reconnected to support any topology the users want. To mimic a physical lab as much as possible, we must emulate a physical connection as close as possible, i.e., we have to capture all layer 2 and above interactions. For example, an Ethernet switch will exchange BPDU messages with neighboring switches during its topology discovery. We have to capture and replay these messages as if the two switches are directly connected.

There are two well supported techniques to create a virtual connection, unfortunately, neither fits our needs. A layer 2 virtual connection, such as VLAN tagging, cannot move packet beyond a single layer 2 domain. A layer 3 virtual connection, such as VPN, tunnels packets at the IP layer, so layer 2 information is lost. To overcome these limitations, we designed our own solution. We use a PC, where a dedicated interface card is connected to each port. The interface card accurately emulate the interactions at layer 1. Our software on each PC captures the full packet information from layer 2 and up, and it delivers the complete packet to the other end of the connection.

Programmable interface: Although we currently only support a web user interface, we are developing a web services interface which will allow a test to be fully automated. The web services interface will support everything that is doable in the web interface through a mouse, including router reservation and connecting router ports. In addition, it will also support packet generation and packet capture in and out of any router port. With these capabilities, a network administrator could fully automate configuration testing, from topology setup, applying configuration, testing, to topology tear down. Similar to the nightly unit test process often used in software development, a network administrator could automatically test any configuration changes nightly and read the log file in the morning to determine whether the change could be rolled out to the production network.

The following sections describe each component of the architecture in more detail. To facilitate the discussion of the capabilities provided to the users, we describe how a user would setup a network topology on the user interface and how a user would connect a new router to the labs.

2.1 Web user interface and web server

A screen shot of the current web user interface is shown in Fig. 2. The left hand column is our router inventory and it lists all routers that are currently connected to RNL and are available. The right hand pane shows the design space, representing the virtual test lab. It is initially empty. The users could drag and drop any router from the inventory to the design plane as they build the test lab. At this point, it is only a design, i.e., the physical routers behind them are not connected in any way yet.

Each router has a picture representation, typically a picture of the back-panel showing the various ports that could be connected to physically. Initially the picture is shown in the left column inventory indicating that the router is not

used in the new design. When the picture is dragged to the design plane, the router is removed from the inventory, since there is only one physical instance for each router listed. To connect one router to another, the user first click on a port on the first router, then drag the line to another port on the second router and the two ports are connected in the topology design. The users can save their topology design, load previous designs or start multiple simultaneous design sessions. The design data is stored in the web server, but the users could export the data to their local drive if desired.

Remote Network Labs

Powered by Accenture Technology Labs

Figure 2: RNL's web user interface

When the users are ready to start their test, they first have to reserve the routers. Since there is only one instance of each router shown in the inventory and since this is a shared facility, some or all of the routers used in a design could be used by other users. The reserve button on the user interface would bring up a calendar similar to that in Microsoft Outlook, which lists all routers used in the current design and, for each router, its current schedule. The users could select the next free period for all routers and make a reservation.

When it is time for a user's reservation, she can deploy the topology design which automatically connects the corresponding router ports according to the user design. Similarly, when the reservation expires, the router connections could be torn down when the next user deploys her test lab design. The web user interface also implements VT100 terminal emulation. If available and if the reservation is valid, the users could directly login to the console port of the router from the browser.

When a user with a valid reservation saves a design, the user interface also attempts to save the router configuration by dumping the configuration file from its console port. This currently only works for certain routers (such as all Cisco ones) that the user interface has a built-in knowledge about how to dump the configuration. We are looking into more generic mechanism to support all routers. If a router configuration is saved, when the users deploy the design, the configuration file is loaded automatically. For other un-

supported routers, the users have to manually save/restore each router's configuration, and they have to make sure that the correct configuration file is loaded on each router of the design.

We plan to support router firmware loading from the user interface in the future. Currently, the users have to login to each router to flash the correct firmware version that they want to test. Although we do provide a standard firmware for each router, it is the users' job to make sure that the correct firmware version is loaded on the router, since it could have been changed by the previous user.

2.2 Router interface

There is a piece of software running on each PC sitting in front of a router. We refer to it as the Router Interface Software (RIS). It has two jobs: capturing the physical configuration information and route packets to/from the router ports and the back-end server.

Each PC has a large number of network interfaces (either PCI-based or USB-based), one for each router port it connects to. A lab manager – the person responsible for putting a physical router into the RNL environment – must first define the physical mapping between the network interface and the router port as shown in the screen shot in Fig. 3. Although we refer to the person defining the mapping as the lab manager in this section, the lab manager could actually be an end-user of RNL. For example, the lab manager could be a network administrator who needs to connect to a specialized piece of network equipment that is only available to her.

Each PC could be connected to multiple routers. For each router, the lab manager has to specify a description and an image file. The description is used in the web interface to inform the users on what kind of equipment it is and the image is used on the web interface as the picture representation of the router.

The lab manager could connect the serial console port on the router to one of the serial ports on the PC, so that the web users can login to the console directly. Once the lab manager specifies which COM port the console port is connected to, RIS can send/receive information to/from the console port.

For each router port that the PC is connected to, the lab manager must specify three things:

1. A description of what the port is. The description is shown on the web interface when users hover their mouse over the port region on the router picture.
2. The network interface adapter the router port is connected to. The lab manager can simply select one from the drop down list.
3. A rectangular area on the router image that corresponds to the port. When the users hover their mouse over this area in the web user interface, the port description will pop up and the users can click on the area to connect to the port. The lab manager can define the active region by simply drawing a rectangle on the router image.

In addition to the port interfaces, the lab manager has to specify which interface is the Internet interface. All communication with the route server is through the Internet interface. The route server is default to be `netlabs.accenture.com`,

but to support future changes and other deployments of RNL outside of Accenture, this server address could be specified by the lab manager.

Once all configurations are specified, the lab manager can save the current configuration, then click the “Join Labs” button to connect to the route server. The details of the interface mapping as well as the router description and image are submitted in a configuration file, so that the defined router would show up on the web user interface. The route server will assign a unique id to each router and a unique id to each port, which uniquely identifies the port when communicating with the route server. To support routers behind corporate firewalls, RIS initiates and maintains a TCP connection to the route server in order to send/receive packets.

After joining labs, RIS goes into the packet forwarding mode. We use the *libpcap* library to capture the raw packet including the layer 2 header. We capture all packets coming from the port, wrap the complete packet in an IP packet which includes the port's and router's unique id and sends the packet to the route server. It also receives packets from the route server. When a packet arrives, it unwraps the packet to find the unique router and port id, then delivers the packet to the correct port.

2.3 Route server

The route server is responsible for keeping track of all available routers in RNL, some of which (those specialized equipment defined by users) could come and go at any time. It is also responsible for routing packets between the router ports. When the users deploy a test lab, a routing matrix is built in the route server corresponding to the users' design. Although several test labs could be deployed at the same time either by the same or by a different user, the routers used in each deployed test lab have to be mutually exclusive; therefore, their contribution to the routing matrix should not overlap.

The packet flow is shown in Fig. 4. When a packet is sent from a router port, RIS captures the packet, wraps it inside an Internet packet with the unique router and port id, and sends it to the route server. The route server unwraps the packet to find the router and port id. Then it looks up the routing matrix to determine which destination router and port the source router port is connected to. Next, it looks up the TCP session associated with the destination router. Lastly, the route server wraps the captured packet, along with the destination router id and port id, inside an Internet packet, and sends it to the RIS sitting in front of the destination router. RIS unwraps the packet and sends it to the destination port.

The RISs essentially build an Internet tunnel through the route server to simulate a virtual wire. Since we capture and replay the entire layer 2 packet and since the network interface card follows the same layer 1 protocol, we can accurately emulate a physical wire between the two ports. From a router's stand point, it cannot tell the difference between our virtual connection from a real physical connection except by the added delay.

To support rich testing capabilities, we are adding traffic capturing and traffic generation modules in the route server. With a web services API, the users can generate arbitrary packets and send them to any router port. Similarly, the users can specify which router port to monitor and be able to capture all packets to and from that port.

Figure 3: Defining network interfaces mapping to router ports in Router Interface software.

3. USE CASES

In this section, we describe some use cases enabled by RNL, including some new use cases that we did not envision when we started the project.

3.1 Configuration testing

RNL is designed to ease configuration testing. It can support layer 3 configuration – a topic well covered by the literature, as well as layer 2 configuration – a space where little research work has addressed. As an example, let us consider a typical enterprise network. To provide resilience, a failover mechanism is often used. Unfortunately, it is difficult to configure failover correctly. Most administrators experiment with configuration settings in a test lab for many iterations before they can set the configuration correctly.

Fig. 5 shows one RNL setup to allow experimentation with the failover mechanism. Two Cisco Catalyst 6500 series switches with a Firewall Services Module (FWSM) are used to provide switch redundancy. They are interconnected on VLAN 10 and 11 so that they can monitor each other for health. The two switches are connected to the intranet, as well as the Internet through a router. Server S1 is connected to the router in order to intercept all traffic going to the Internet, and server S2 is connected to the two switches to send/receive intranet traffic. A few servers are provided in the RNL router inventory which the users could use to setup this lab, but the users could also add additional servers to RNL just as if they are adding a router.

The user has access to the switches' console port, so that she can experiment with configuration settings. She can also shutdown one switch or disable all of its links to simulate a switch failure and observe whether the failover mechanism is triggered. The user also has access to the console for server s1 and s2, so that she can send probe packets and observe whether traffic is routed correctly.

As discussed in Catalyst 6500's configuration manual [4], configuring failover is not trivial. For example, the manual

Figure 4: How packets are routed

Figure 5: A RNL set up to experiment with configuration on the failover mechanism

states that a switch software that supports BPDUs forwarding should be used and that the user must configure the FWSM to allow BPDUs. Both steps could be easily missed during the first pass. Using RNL, we can not only accurately capture the end result of a configuration, but we can also capture transient behaviors. For example, a loop may occur if the switches are configured incorrectly and when both FWSMs are discovering the presence of the other module at the same time. Such a transient behavior is difficult to capture using simulation or static analysis techniques [20].

3.2 Test automation

RNL helps automate network testing in a couple of aspects. First, RNL offers the ability to fully automate the setup and tear-down of any topology. Although the initial release only supported a browser interface, we are working to expose a set of web services interfaces to allow one to programmatically reserve equipment, setup topology and deploy.

Second, RNL offers better testing capability than what was available before. In a physical testing environment, visibility is limited. To observe what is going on in the test, we have to find a free port on a router and connect a traffic generator to capture the packets received. This limitation constrains us on the number of points where we can observe and forces us to design only simple test cases that are visible. In comparison, RNL gives the users the full visibility on every wire in the test. In addition, since all traffic capture is done in software, we are not constrained by the number of observation points, so that we can verify fully whether the test is working as expected. Beyond traffic capture, RNL can also generate traffic without specialized equipment. Unlike in a physical environment, RNL can generate traffic on any wire and it can generate traffic in only one direction, i.e., even though two ports are connected in the test lab, only one port sees the generated traffic.

To illustrate the value of automated tests, let us consider the example shown in Fig. 6, which is a simplified example adopted from [9]. There are four routers. Initially, R3 is connected to R1, R1 is connected to R2, and R2 is connected

to R4. Suppose there is a security requirement that subnet A cannot talk to subnet B. This policy is easy to enforce by setting up a packet filter at interface R1.2 and R2.2. However, when a new link is added between R3 and R4 in the future, packets from subnet A are routed through R3 and R4 to reach subnet B, thus violating the security policy.

Figure 6: Setting up an automated test that can verify connectivity requirements

The security policy is easy to verify in RNL using an automated test. The test first sets up the topology as shown and loads the current configuration file. It then invokes the web service API to generate a packet destined to subnet B on port R1.1. Lastly, the test calls the web service API to capture packets at port R2.1 to see if the packet has made through. Instead of using the web service API for packet generation and capture, the user could also hook up an IXIA traffic generator to port R1.1 and R2.1 to achieve the same goal.

Using an automated test has the benefit of capturing policy information automatically. Instead of asking a user to manually verify each enforced policy whenever a topology or configuration change happens, the test could automatically check such policies and flag the user only when a policy is violated. Similar to a nightly unit test commonly used in software development, RNL enables these automated tests to be run regularly whenever a topology or configuration change happens. In our example, the policy violation could be caught during the nightly run after the link addition, instead of waiting to be discovered after a security breach.

RNL is originally designed for lowering the cost of building a test lab by efficiently sharing the expensive testing equipment, but we are pleasantly surprised that there are many other use cases, which we will describe in the rest of this section.

3.3 Avoid shipping

There are certain diagnostic and management equipment that must be deployed in a client's enterprise network for a short period. For example, Nercordia NetMRI product can help troubleshoot network problem and analyze network performance. When a client's network experiences problems, we have to ship the NetMRI equipment over, deploy for a few weeks to diagnose the problem, then ship it back. The shipping is not only costly, but more importantly, it causes several days' delay before one can start to diagnose the problem. Since network outage is disruptive, a network

problem persisting for more than a few days is often not acceptable. In addition, because of the hassle in shipping, the users are reluctant to relinquish the equipment until they are absolutely sure that the problems are resolved, resulting in inefficient sharing of the test equipment.

RNL can avoid the shipping hassle and improve the utilization of the test equipment. First, the user needs to expose the internal network, i.e., connect a PC with RIS to one Ethernet port within the Enterprise network, and join it to RNL. Then with the web user interface, the user can create a new design with two pieces of “test equipment”: the NetMRI equipment and the exposed Ethernet port. Once the two are connected and deployed, the NetMRI equipment is virtually deployed in the Enterprise network.

3.4 Training

Existing training environments are mostly based on emulation or simulation. They are limited both in terms of the types of equipment available and the realism offered. To overcome the limitations of simulation and emulation, dedicated training facilities with real routers are built, but because it is difficult to change the wiring, they only offer a small number of topologies. With RNL, we are no longer bounded by a few, but instead, we can experiment with a variety of topologies to gain a full understanding of the effects of router configuration.

3.5 Application testing

RNL can test applications under real-life scenarios. Applications designed in a local network may experience widely different behavior when deployed in a real-life scenario where the users may be far away. RNL can inject delay and jitter to simulate any wide area links. By deploying applications on top of a test network in RNL, we can test how an application behaves under a real-life scenario. The capabilities to inject arbitrary delay and jitter are under active development.

3.6 Remote collaboration

RNL not only allows a network equipment to be located anywhere, it also allows the users to be located anywhere as long as they have an Internet connected browser. This not only results in more efficient use of the expensive routers, but it also allows efficient sharing of human experts. When a configuration fails in testing, one can simply send a URL to experts at remote sites to help debug the problems. Since no time is wasted in travel, a few experts are enough to help on a large number of projects.

4. ONGOING WORK AND REMAINING CHALLENGES

Much more work remains to make RNL a useful and scalable system. RNL currently only contains a limited set of equipment. Because of the limited inventory, it is not yet valuable to a network administrator, hence it has not been deployed for production use. In this section, we describe the challenges remaining and our plan to address them through real deployment as we acquire additional equipment and build up the inventory.

Using a real router means that a physical router could be used by only one user at a time. Although we expect to increase the number of router resources when the number of users of RNL increases, it is still highly desirable to share

the available equipment as efficient as possible. Some commercial routers [5] [10] support router virtualization already (referred to as a logical router). For these routers, we plan to enhance RIS to multiplex/de-multiplex traffic so that a user could reserve a slice of the router, in addition to being able to reserve the whole physical router, for example, to play around with the logical router features.

Although Ethernet is the dominant one, there are a large number of other layer 2 protocols. We believe that, as long as we can find a PC adapter for a layer 2 protocol, we can capture the complete packet, send it through the Internet tunnel, and replay the layer 2 packet at the other end. Although possible, the RIS likely needs customization for each layer 2 protocol that we will support in the future.

Although not designed for performance testing, we are looking into addressing the limitation using a couple of approaches. We should note that full performance testing is not always required since one can scale down the system and still be able to predict the system performance[16].

Layer 1 switch: For equipment located at the same physical location, we can add a layer 1 switch, such as MRV’s Media Cross Connect product [13], to provide full link bandwidth. It will be connected as shown in Fig. 7.

Figure 7: Wiring diagram with an additional layer 1 switch

The layer 1 switch is programmable and it is connected to the routers directly. During performance testing (selectable by user), the layer 1 switch can be programmed to directly bridge the two ports. Alternatively, the layer 1 switch could connect the router port to RIS, which is in turn connected to the Internet. Programming the layer 1 switches will be through the same web services API so that the users can benefit from test automation even for performance testing.

Compression: Performance testing packets often look similar to one another. They are often generated from the same template, where each packet may have a slight different marking, for example, having a different sequence num-

ber. By exploiting the similarities across packets, we could achieve a high compression ratio. We are also looking into GPU and the Intel SSE instruction set capabilities to speed up the compression effort.

Even with effective compression, to support full speed, the interface PC must be able to drive the port at the full line rate. To scale RIS, we can simply scale the number of PCs used, and limit the number of ports supported on each PC. In the extreme, we can have one PC per router port. Today’s multi-core high-end PCs could drive a 10G link comfortably.

In addition, the route server must also be scalable, which is not as trivial. To simplify implementation, we funnel all traffic through the central route server in the initial release, so the route server can easily become the bottleneck. To scale the route server, we are looking into a distributed architecture for the next release. Since the routing matrices between different users do not overlap, we can have one route server per user. Ideally, since a connection is fixed, we should pass the forwarding responsibility to the RIS, who should pass packets to the RIS sitting in the other end of the connection directly. Unfortunately, one of the design requirements is to support routers behind corporate firewalls, and if two routers are both behind firewalls, it is difficult to make a direct connection.

There are also a couple of issues we plan to investigate through real deployments. First, packet delay and jitter through the Internet tunnel could pose a problem. We do not believe delay and jitter will affect configuration testing, but they may impact performance testing. Second, Internet traffic is not free. If we have to provision a large amount of Internet bandwidth for performance testing, the cost could reduce the savings from equipment sharing. Again, we do not believe configuration testing would pose a problem since the volume of traffic exchanged is small.

5. RELATED WORK

The configuration complexity has been recognized by many. Greenberg et al. [9] argued for a new architecture to separate the decision logic from the protocol to make configuration easier. Alimi et al. [1] proposed new capabilities in routers to support virtual routers for configuration validation. All these proposals require router changes that are unlikely to happen over night. Although simulation, emulation and test labs are the predominant solutions in practice today, one could also use static configuration file analysis techniques [20]. However, the analysis is limited (only to reachability analysis) and it cannot capture an individual router’s behaviors.

There are several other experimental evaluation facilities that have been built. Emulab [18], VINI [3] and Open Network Laboratory (ONL) [11] are designed for evaluating networking protocols and algorithms – a very different purpose than ours. To enable new routing protocol/algorithm experimentation, they allow the routing node to be changed, either in software running on a general purpose PC or in programmable logic. Unfortunately, their routing nodes cannot emulate the behavior of real routers. In contrast, we use real routers to accurately reproduce the effects of configuration changes. Emulab uses VLAN tagging to emulate a link and ONL uses a layer 1 switch for programmable connections, so they can more accurately model a network link. In contrast, both VINI and RNL uses IP tunnels to simulate links. Similar to RNL, WAIL [2] uses real routers, but they have

to be centralized in the same location, thus limiting its scale and flexibility.

The key idea behind Remote Network Labs is wire virtualization, which is only one part of the network virtualization. Router virtualization has been under active development for some time. If it is a software router, one can simply put it in a hypervisor and virtualize both the control plane and the data plane [8]. Even for hardware router platforms, there are commercial offerings already [5] [10].

Wire virtualization could be achieved by VPN, but one must configure the peering routers to be in the VPN mode. Whereas in RNL, the router could be set to any configuration the users want. Since the users’ settings could conflict with the VPN setting, we cannot use VPN as an implementation mechanism.

RNL gives users the ability to configure an overlay network. But differing from other overlay networks, such as PlanetLab[15], RNL is not confined to a specific topology. In addition, the users have direct full control of the test equipment hardware, with the ability to change both the configuration and the firmware.

6. CONCLUSION

We present Remote Network Labs (RNL), a network cloud facility from where end users could request network equipment to construct a virtual test lab. It is designed to efficiently utilize test equipment and lower the cost of building a test lab. Beyond simple cost savings, it also has many features that were not possible before; it can reduce the time to build a lab, fully automate tests from setup to tear-down, help training, avoid shipping, recruit remote experts etc. RNL is based on a flexible architecture. Even if a new router is not available in RNL, the users could add their own and still leverage other existing equipment in the inventory.

RNL has its limitations. We are addressing the performance testing limitation through a combination of layer 1 switches and packet compression. Another limitation is that each router equipment can only be used by one person at a time. This problem can be easily addressed through router virtualization. Our use case is a strong motivation for having full virtualization support in network equipment.

7. ACKNOWLEDGMENTS

The authors would like to thank Minchi Hu Chang, Manjula Shankar, Francisco Yip, Francisco Flores, Sunitha Hariraman, and Israel Jordan for their help on implementing part of RNL functionality. The authors would also like to thank the anonymous reviewers and Dr. Albert Greenberg for their generous and helpful comments.

8. REFERENCES

- [1] R. Alimi, Y. Wang, and Y. R. Yang, “Shadow configuration as a network management primitive,” in *Proc. SIGCOMM*, 2008.
- [2] P. Barford and L. Landweber, “Bench-style network research in an internet instance laboratory,” in *Proc. SPIE ITCOM*, 2002.
- [3] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, “In vini veritas: Realistic and controlled network experimentation,” in *Proc. SIGCOMM*, 2006.
- [4] “Catalyst 6500 series switch and cisco 7600 series router firewall services module configuration guide,”

- http://www.cisco.com/en/US/docs/security/fwsm/fwsm32/configuration/guide/fail_f.html.
- [5] "Cisco logical routers," http://www.cisco.com/en/US/docs/ios_xr_sw/iosxr_r3.2/interfaces/command/reference/hr32lr.html.
 - [6] "Configuration management delivers business resiliency," The Yankee Group, Nov. 2002.
 - [7] "Dynamips," http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator.
 - [8] N. Egi, A. Greenhalgh, M. Handley, M. Hoerd, L. Mathy, and T. Schooley, "Evaluating xen for router virtualization," in *Proc. Computer Communications and Networks (ICCCN)*, 2007, pp. 1256–61.
 - [9] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4d approach to network control and management," in *Proc. of SIGCOMM*, 2005.
 - [10] "Juniper logical routers," <http://www.juniper.net/techpubs/software/junos/junos85/feature-guide-85/id-11139212.html>.
 - [11] F. Kuhns, J. DeHart, A. Kantawala, R. Keller, J. Lockwood, P. Pappu, D. Richard, D. Taylor, J. Parwatikar, E. Spitznagel, J. Turner, and K. Wong, "Design and evaluation of a high-performance dynamically extensible router," in *Proceedings of the DARPA Active Networks Conference and Exposition*, 2002, pp. 5–2002.
 - [12] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding bgp misconfiguration," in *Proc. SIGCOMM*, 2002.
 - [13] "Mrv media cross connect," <http://www.mrv.com/product/MRV-MCC-Chass/>.
 - [14] D. Oppenheimer, A. Ganapathi, and D. Patterson, "Why internet services fail and what can be done about these," in *Proc. USENIX USITS*, Oct. 2003.
 - [15] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," in *Proc. HotNets'02*, October 2002.
 - [16] K. Psounis, R. Pan, B. Prabhakar, and D. Wischik, "The scaling hypothesis: Simplifying the prediction of network performance using scaled-down simulations," *ACM Computer Communication Review*, 2003.
 - [17] "Router simulator command reference," <http://routersimulator.certexams.com/help/commands.html>.
 - [18] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*. Boston, MA: USENIX Association, Dec. 2002, pp. 255–270.
 - [19] A. Wool, "A quantitative study of firewall configuration errors," *Computer*, 2004.
 - [20] G. G. Xie, J. Zhan, D. A. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford, "On static reachability analysis of ip networks," in *Proc. of IEEE Infocom*, 2005.