Note on Remote Laboratory Access: A Networking Perspective

Alexander A. Kist
Faculty of Engineering and Surveying
University of Southern Queensland
Toowoomba, Queensland 4350
Australia
Email: kist@ieee.org

***Abstract*—*Remote Laboratory Access* in education has become a focus of Universities and other educators. This work addresses issues of implementing a scalable remote-laboratory access system. The main requirements which are considered include: to developed generic solutions allowing access to many different hosts via a common interface, straightforward student use, fast experiment setup and minimal configuration requirements.**

The particular focus of this paper is the networking aspect of remote laboratory access. A general discussion of alternatives is followed by specific implementation details of a prototype system. The core component of this system is the *Remote Access Gateway* which authenticates users and establishes end-to-end connections between students and experiments.

I. Introduction

Remote Laboratory Access has been on the agenda of Universities and other education providers for a while, mainly to enhance distance education facilities and improve student learning. Other drivers include: aiming to offer more flexibility to students and to reduce operational costs of laboratory classes.

A wealth of issues relate to the task of providing online access to laboratory experiments. Broadly, these issues can be divided into either educational or technical. Consequently, many papers have been published in the area of remote access. Work includes publications that investigate the educational value of these activities such as [1] and [2]. Other publications describe more specific implementations, often in robotics or related fields (e.g. [3]). This paper also concludes that learning experiences for different offer-modes are similar. Many applications use Java-based, custom interfaces. For example, [4] introduces Java remote access framework.

The research reported in this paper takes a different angle; it focuses on a systems view of the remote laboratory access problem. More specifically, it addresses networking and access control issues. Remote laboratory access provides admission to experiments, authenticate and admit users and automate laboratory tasks.

Currently, there is no turnkey solution available to provide remote laboratory access. The focus of our research approaches the problem of providing remote laboratory access from a networking angle. The aim is to provide a common framework for many different remote laboratory experiments that handles authentication, access control, scheduling and queuing.

It also introduces a prototype implementation, using network layer techniques to provide a scalable, common authentication and remote laboratory access platform for many experiments. One of the underlying design constraints is to use existing methods, where possible.

This paper is organised as follows: Section II introduces the underlying usage scenario and system requirements. Section III discusses related networking research outlining alternative access methods. The proposed system is discussed in Section IV and operational issues are discussed in Section VI.

II. Scenario & Requirements

The underlying usage scenario of this framework is depicted in Figure 1. A number of experiments are available for remote-access. These are connected to a corporate network and depicted on the left hand side (Experiment 1-4). The network is firewalled from the public Internet. Potential users are shown on the left-hand side. They are directly connected to the Internet (Client 2). Alternatively, the users can also be located behind firewalls (Client 1) or access the Internet via *Network Address Translation* (NAT) gateways or proxy servers (Client 3). The two groups communicate via the Internet. No specific infrastructure requirements, such as bandwidth or latency, are assumed.

The aim of this work is to provide connectivity between users and experiments; furthermore, authentication, access control and management are provided.

Details of experiment automation are not in the scope of this paper. However, it is assumed that many different applications might be used, including remote Desktop software, such as Virtual Network Computing (VNC) or Windows Remote Desktop, web cams, java user interfaces, voice and video, specialist software tools, custom applications etc. These applications might have different requirements for Quality of Service (QoS).

Experiments are managed by one host. These hosts are dedicated and have specific IP addresses, but do not require

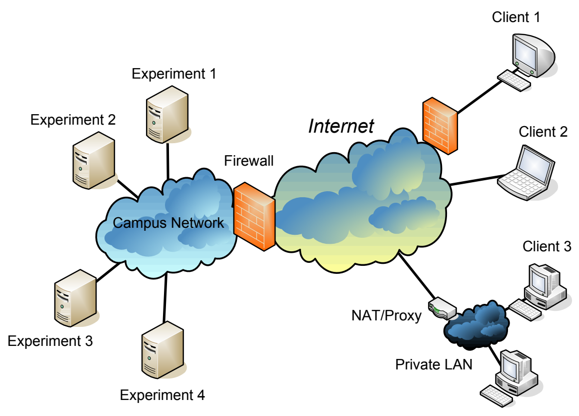

Fig. 1. Remote Access Laboratory Scenario

dedicated hardware. VMware [5] or other virtualisation software can be used to host several virtual machines on one physical computer. One guest operating systems manages one experiment.

System requirements can be summarised as follows:

- Minimal requirements for client hosts
- Simple configuration
- Experiments can be added easily and with no detailed knowledge of the system
- Scalable, approximately 1000 users, 100 experiments
- Extendable, ie. number of experiments, multiple experiments of the same type
- Additional functions, e.g. video can be added easily
- Secure, experiments can not be accessed by unauthorised parties, including local and remote users
- The system handles authentication and authorization
- Scheduling and booking of experiments is provided by the framework
- Several simultaneous connections [1]
- No direct access to the Internet via experiment server
- No access from student to student computer
- All experiments appear on the same IP address
- Networking aspects are transparent to the user

III. Background

The core aspects of this framework directly relate to access problems, encountered in other networking areas. This section summarises research that relates to remote access problems. Two areas are discussed in detail: server farms and user IP identification.

A. Server Farms

The problem of accessing multiple servers of the same type, commonly occurs in the context of virtual servers or server farms. This is directly related to the issue of providing network access for experiments. A number of solutions have been suggested and are widely deployed. The current state of the art, for methods in setting up a distributed web-server systems under one administrative control, is discussed in [6]. The article focuses on architecture, routing and dispatching algorithms. Dynamic load balancing is specifically discussed in [7]. In the remote laboratory context, the following admission methods can be adopted for remote laboratory access.

1) Unique Experiment Domain Names: One domain name is allocated per experiment and the Domain Name System (DNS) is used to map the corresponding name to an IP address. This mapping is dynamic and takes the system online status into account. DNS load balancing techniques are used, if multiple experiments of the same type are available. This approach provides very limited control and is not transparent for users. Different domain names for different experiments are required. Experiment allocation and access are not handled via the same interface.

2) Network Address Translation (NAT): NAT has enabled Internet growth beyond the address space limits of IPv4. However, at the same time, NAT has been controversial in the Internet community [8].

Network address translation and TCP/UDP port translation are often used to map private addresses to one or more public IP addresses. Another common use is Destination NAT (DNAT) which makes local services on private networks available to public Internet. This mechanism is also employed to implement virtual server load balancing.

A server cluster is accessed via one public IP address. A dispatcher, hosting the public IP address, receives requests and assigns them to cluster servers by rewriting the destination address from the cluster address to the specific server address. The dispatcher tracks connection-state to ensure that all packets that belong to the same transaction are routed to the same server. Arbitrary server assignment policies can be used including *round-robin, least load or random*.

The Linux netfilter framework [9], used in the Linux 2.4.x/2..6.x kernel series, can be used to implement NAT. It is a very powerful tool allowing arbitrary IP packet filtering rules and packet mangling. It uses Kernel hooks with call-back functions and a selection of tables that contain forwarding rules. A user space program, *iptables*, is used to modify these tables. Various, user-friendly interfaces are available, implementing firewalls and other, similar software tools.

3) ONE-IP: The ONE-IP approach uses the Linux *ifconfig alias* option to configure one interface with a second IP address [10]. Several hosts, forming part of a cluster, have a valid IP address, as well as an alias IP, common to all cluster hosts. A dispatcher accessible from the Internet, has the only public interface with the common IP address.

The dispatcher routes incoming requests to the cluster servers, that use an arbitrary mapping to the cluster hosts primary IP address. All packets belonging to the same transaction

[1]To support authentication, remote desktop applications, file transfers, video others.

are sent to the same cluster server. The cluster servers respond to the request, using the alias IP address. This method does not rewrite IP addresses.

This section introduced a number of methods, commonly used to distribute load to cluster members in server farms. To apply these methods in a remote laboratory case, the server assignment is not determined by a policy, but by an arbitration server that grants access to particular users.

Because of the ease of configuration and reliability, option three, Network Address Translation is being used by the prototype. However, other options can easily be implemented.

B. *Uniquely Identifying User IPs*

The identification of users is an issue, encountered in many contexts. Examples include, corporate network access, secure web servers and *Virtual Private Networking* (VPN).

For the purpose of user authentication, it is necessary to uniquely identify the host addresses of student computers. IP addresses are unique; however, due to NATs and proxies, it can not be ensured that a IP address belongs to one user. To overcome this issue and to encrypt data, VPN solutions are often used.

There are many flavours and implementations, however, broadly two [2] alternatives can be identified: IP security (IPsec) [11] and Secure Sockets Layer (SSL) [12] VPN solutions. IPsec is an encrypted transport protocol. In the past there have been issues with IPsec traffic and NATs, however, these are being addressed by IPsec NAT Traversal (NAT-T) [13], [14]. IPsec NAT-T is widely available, i.e. supported by Windows XP, since Service Pack 2.

SSL allows an encrypted connection between web-browsers and remote host. *OpenVPN* [15] uses SSL to provide a full VPN application. SSL-type techniques are more flexible in overcoming firewall issues, however, they are not industry standard-based and not seen as best practice techniques. They offer greater flexibility, but have limited commercial support. In principle, both techniques can be used for remote laboratory access.

IV. System Design

The system is depicted in Figure 2. As in Figure 1, experiment hosts as well as student hosts are depicted. The core component of the system is shown in the centre, the *Remote Access Gateway* (RAG). The connections symbolise logical node attachments. Users connect to the experiment farm via RAG. The gateway handles all incoming requests, authenticates users and forwards the traffic to the appropriate experiment hosts.

It is assumed that users reaching the RAG have IP addresses that uniquely identify their host. As outlined in Section III-B, this can be achieved with various VPN implementations.

[2]In the past the *Point-to-Point Tunneling Protocol* (PPTP) has often been used to build VPNs.

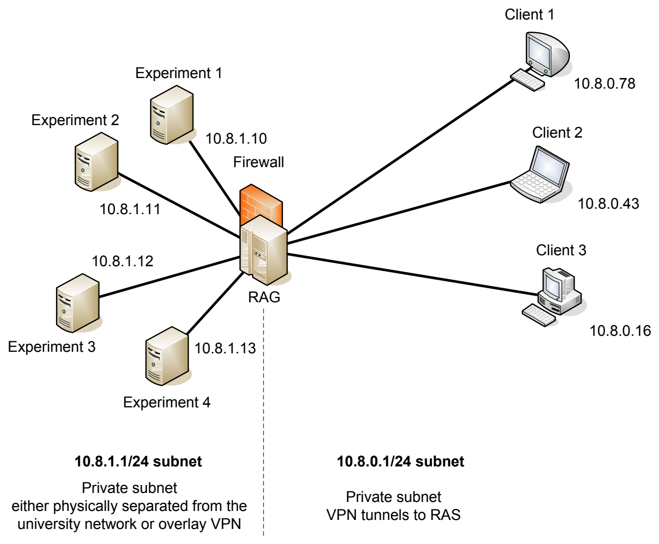

Fig. 2. Remote Access Laboratory Scenario

At the same time, it is also assumed that the experiments have unique, known IP addresses. Experiments are isolated from the campus network, as well as the public Internet. Therefore, they are located on a separate LAN, firewalled or are connected via the experiment VPN. Experiments can be co-located with the RAG or they are located in different sites. Experiments are either physically or logically connected to RAG.

The Remote Access Gateway handles all user requests and traffic to and from the experiments. Figure 3 depicts the main function blocks of the gateway. This includes database, forwarding engine, user interface and core functions. The system is designed in a modular way. Different roles are handled by separate entities and can be implemented on co-located systems. Therefore, the gateway does not necessarily consist of a single server.

A. *User Interface*

The user interface is used to interact with students as well as a configuration and administration interface. HTML pages are generated using *php* scripts providing a web interface. Authenticated users see a list of active experiments available to them.

This laboratory hosts can instantly be accessed, if they are not in use. Otherwise, they can be booked for future access. If an experiment is selected, the forwarding engine is configured to enable access.

Once a user is authenticated for an experiment, a status page displays the remaining time for this authentication. It also re-authenticates the experiment on an ongoing basis. If the page is closed, the authentication expires and access rules are removed. The experiment is no longer reachable.

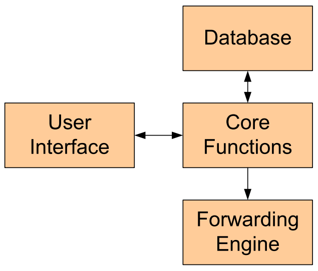

Fig. 3. RAG Function Blocks

The user interface can be integrated with existing Course Management Systems (CMS), such as moodle [16]. This also allows the reuse of CMS functions, such as authentication and student management.

B. Forwarding Engine

The forwarding engine uses network address translation or port forwarding to enable access to various experiments.

It is important to ensure the IP address of the current user remains for the whole experiment. Due to the widespread use of proxy servers and network address translation, it cannot be guaranteed that an IP address does not change between requests (load balancing proxy) or that the IP belongs to a single user (many users behind a NAT use the same public IP address). Only within one administrative domain it can be ensured that one IP matches one user, for example, within a University subnet. To overcome this issue, VPN technology is used. As outlined in Section III-B, IPsec or SSL VPNs are available. The particular technology does not have an impact on this proposal, as long as it ensures an unique IP address.

For the actual implementation of the forwarding rules, this proposal uses the *iptables* interface. An alternative implementation could use *One-IP* techniques to avoid packet mangling discussed in Section III-A3. Due to the modular setup, this can be implemented in the future, if necessary.

All experiments are located on the same IP address, implying that the domain name is also the same. This avoids any configuration issues on the student site. All client software tools are configured with the same IP address for all experiments. For discussion, it is assumed that experiments are located at *10.10.10.10* [3]. As outlined above, every experiment as well as every authenticated user has an unique IP address.

Table IV-B depicts an example of forwarding rules, for three users with the IP addresses of *10.8.0.78*, *10.8.0.43* and *10.8.0.16*, respectively. They are authenticated for three experiments, located at *10.8.1.10*, *10.8.1.11* and *10.8.1.12*, respectively. Note, experiments are located on a different subnet from the users. If a packet is received that matches

source	*destination*	*source to*	*destination to*
10.8.0.78	10.10.10.10	10.8.0.78	10.8.1.10
10.8.0.43	10.10.10.10	10.8.0.43	10.8.1.11
10.8.0.16	10.10.10.10	10.8.0.16	10.8.1.12

TABLE I
EXAMPLE FORWARDING RULES

the source and destination address, the destination address is rewritten from *10.10.10.10* to the new destination address and the packet is forwarded to the experiment host. Packets, in the reverse direction, are translated back accordingly.

If a packet arrives with a destination of *10.10.10.10* and a source that is not included in the table, the packet is not routable. If users are authenticated, entries are added to this table, if users lose authentication entries are removed from the table. The experiment can no longer be accessed[4].

C. Database

The database is used to store user information, system state and all relevant experiment data, including user and experiment IPs. The current implementation uses a standard *mysql* database.

D. Core Functions

The core functions interact with the different modules. They are implemented using *php* for general function and *bash* scripts for operation system related function and to interact with *iptables*.

V. OPERATION

This section outlines the RAG operation in more detail. Figure 4 depicts the steps that are necessary to access experiments:

1) Before student access is possible, experiments have to be registered with RAG. This establishes connectivity between RAG and the host.
2) A user logs on to the remote laboratory webpage.
3) The interface employs core functions to access the database.
4) User access rights are verified and experiments are offered via the web interface.
5) On selection of an experiment, the database is updated and the forwarding rules are configured.
6) The user is able to access the experiment.
7) End-to-end connections from the experiment-client to the experiment-server can be established.

More details on specific functions are provided in the subsections below.

[3] If a VPN is used, a route to this IP can be advertised. It has to point to the local gateway. An additional route is required from the VPN gateway to the RAG.

[4] In practise, a additional firewall entry is required to terminate all active connections to the experiment on lose of authentication.

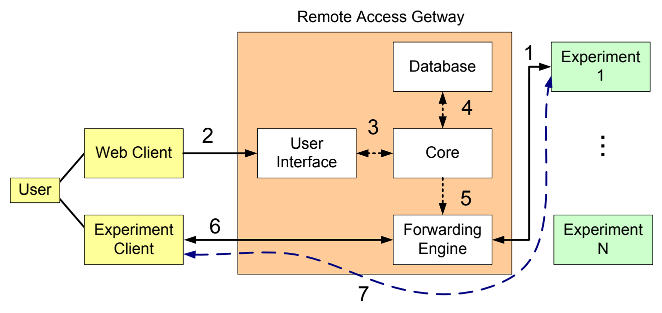

Fig. 4. RAG Operation

A. Authentication

Currently, simple *http* authentication is used. In the future, this can be replaced by authentication API functions, provided by student management systems or other databases. As outlined above, this function can also be offered by a course management system. During authentication, the student IP address is detected and stored. This information is used to configure access rules by the forwarding engine.

B. Experiment Registration

Experiment servers host software that drives the laboratory trials. They are added once to the database and connected to the RAG on a separate VPN subnet. During operation, it is automatically detected if systems are online, by evaluating connected VPN information.

VI. Discussion

This implementation can serve as a framework for providing flexible, unified access to experiments in an online learning environment.

A. System Requirements

In Section II a number of system requirements were outlined. All of these have been addressed by the RAG system. The only issue, not discussed in great detail, is scheduling and experiment booking. These functions can be implemented using *php/mysql* and can be directly integrated in the user interface.

B. Scalability

The RAS can be operated as a server cluster if a single system reaches it's limit. The framework is scalable to a large number of systems. Using address translation for the experiment access allows several, identical experiments to be operated transparently from the user perspective.

C. Quality of Service

The *iptables*-based implementation has the additional advantage that QoS measures can be included and bandwidth guarantees can be enforced.

For one experiment, for example, 30% of the bandwidth, consumed by the experiment, is reserved for video transmissions and 40% are reserved for other real-time applications, etc. Furthermore, bandwidth consumption of experiments overall can also be limited, i.e. all experiments use fair bandwidth share.

VII. Conclusion

This paper discussed network-related aspects of remote laboratory access. Specifically, alternative server management methods were outlined. In the second part a specific implementation was introduced. The prototype demonstrates a system that addresses the networking related requirements for a flexible and scalable remote laboratory access. The modular design allows for the exchange of function blocks by alternative implementations. The main component, the Remote Access Gateway, is implemented on a Gentoo Linux system. The current design uses standard *php mysql* to generate the user interface web pages. Experiment access employs the *netfilter* framework. The remote access gateway provides user-transparent, flexible access to offside students.

References

[1] B. Moulton, V. Lasky, and S. Murray, "The development of a remote laboratory: educational issues," *World Transactions on Engineering and Technology Education*, no. 1, pp. 19–22, 2004.
[2] J. Bourne, D. Harris, and F. Mayadas, "Online engineering education: Learning anywhere, anytime," *Journal of Engineering Education*, no. 1, pp. 131–146, January 2005.
[3] C. Tzafestas, N. Palaiologou, and M. Alifragis, "Virtual and remote robotic laboratory: Comparative experimental evaluation," *IEEE Transactions on Education*, pp. 360–369, August 2006.
[4] C. Röhrig and A. Jochheim, "Java-based framework for remote access to laboratory experiments," *In IFAC/IEEE. Symp. Advances in Control Education Gold Coast, Australia*, 2000.
[5] (2007) Vmware. VMware Inc. [Online]. Available: http://www.vmware.com/
[6] V. Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu, "The state of the art in locally distributed web-server systems," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 263–311, 2002.
[7] V. Cardellini, M. Colajanni, and P. Yu, "Dynamic load balancing on web-server systems," *IEEE Internet Computing*, no. 3, pp. 28–39, May/Jun 1999.
[8] P.Francis, "Is the Internet going NUTSS?" *IEEE Internet Computing*, pp. 94 – 96, Nov.-Dec. 2003.
[9] (2007) Netfilter project. [Online]. Available: http://netfilter.org
[10] O. Damani, P. Chung, Y. Huang, C. Kintala, and Y.-M. Wang, "One-ip: Techniques for hosting a service on a cluster of machines," *J. Computer Networks and ISDN Systems, Elsevier Science, Amsterdam, Netherlands*, p. 10191027, September 1997.
[11] S. Kent and K. Seo, *Security Architecture for the Internet Protocol*, IETF, December 2005, RFC 4301.
[12] W. Chou, "Inside ssl: the secure sockets layer protocol," *IEEE IT Professional*, no. 4, pp. 47– 52, Jul/Aug 2002.
[13] T. Kivinen, B. Swander, A. Huttunen, and V. Volpe, *Negotiation of NAT-Traversal in the IKE*, IETF, January 2005, RFC 3947.
[14] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M. Stenberg, *UDP Encapsulation of IPsec ESP Packets*, IETF, January 2005, RFC 3948.
[15] (2007) Openvpn. OpenVPN Solutions LLC. [Online]. Available: http://www.openvpn.net/
[16] (2007) Moodle course management system. [Online]. Available: http://www.moodle.org/