View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Open Archive Toulouse Archive Ouverte

IREEL: Remote Experimentation with Real Protocols and Applications over Emulated Network

Laurent Dairaine
LAAS-CNRS/ENSICA
1 place Emile Blouin
31056 Toulouse Cedex 5,
France
dairaine@ensica.fr

Guillaume Jourjon, Emmanuel Lochin, Sebastien Ardon
National ICT Australia
Locked Bag 9013
Alexandria NSW 1435,
Australia
firstname.lastname@nicta.com.au

Abstract: This paper presents a novel e-learning platform called IREEL. IREEL is a virtual laboratory allowing students to drive experiments with real Internet applications and end-to-end protocols in the context of networking courses. This platform consists in a remote network emulator offering a set of pre-defined applications and protocol mechanisms. Experimenters configure and control the emulation and the end-systems behavior in order to perform tests, measurements and observations on protocols or applications operating under controlled specific networking conditions. A set of end-to-end mechanisms, mainly focusing on transport and application-level protocols, are currently available. IREEL is scalable and easy to use thanks to an ergonomic web interface.

Keywords: Virtual Lab, E-Learning, Networking courses.

1. Introduction

Every networking teacher has faced the difficult problem to illustrate their courses with real networking experiments. During such courses, experimentation is a common way to understand and quantify both functional and non-functional properties of networking protocols. Indeed, in order to facilitate the learning of the theoretical basis, students should see and study the real behavior of the network and transport protocols and the resulting impact on the applications. This is generally accomplished through experimentation.
Building testbed, running experiments, observing packets exchanges and measuring systems performance from real protocol implementations are typical tasks for the network experimenters. These tasks usually incur a high setup and maintenance cost in time and efforts for the experimental testbed (machines, operating systems, applications), which is a significant burden for the teacher. In addition, using a testbed often does not allow the participation of the entire classroom simultaneously. Finally and importantly, the typical network setup has some serious limitation in the range of network conditions available to students to experiment with.

As an alternate solution, simulation and particularly event-driven simulation such as ns-2 [5] or OMNeT++ [2] or OPNET [1] are a powerful tools to evaluate protocols and algorithms. These simulation packages are typically easily installed and maintained on the school network of workstation and readily available to all students simultaneously. However, the simulation approach comes with a steep learning curve for simulation languages and model abstractions, which are largely outside the scope of most networking courses. Finally, simulation typically does not allow observing real protocol implementations (only models), and results accuracy are therefore dependent on the model hypothesis.

For several years, high-performance and affordable computing and networking infrastructure have enabled the development of network emulators [4, 9, 12, 13]. The main goal of network emulation is to reproduce the behavior of a real network such as specific Internet topologies [12], Wireless networks [7, 13] or satellite links [3, 6] without the complexity of building the actual testbed. Emulation usually consist of a device (which can be a standard PC running emulation-capable operating system) which can introduce "artificial impairments" at the packet forwarding level, for example reproducing specific packet loss distribution or available bandwidth variation. These impairment can be static (i.e. constant for the duration of the experiment), or dynamic, following specific scenarios.

Emulation is therefore particularly interesting in the context of computer networking education as it allows real protocols implementations to be tested or studied, under a wide range of controlled network conditions. However, as in the case of real network testbed, deploying emulation approach can be as difficult as deploying a real testbed.

In the present effort, we propose the Internet Remote Emulation Experiment Laboratory (IREEL), which allows realizing predefined experiments, while using real protocols and emulated networks. Through its web-based user interface, the IREEL is used remotely and asynchronously by students who can queue up experiments to be executed in the controlled and emulated IREEL network.
This paper is organized as follow. Section 2 gives the related work. In section 3, the requirements about network experiment platform for education are provided. The IREEL architecture is described in section 4 and section 5 presents the experiment process in IREEL. Section 6 gives some details on various types of experiment results available. Finally, section 7 provides a conclusion.

2. Related work

There are only a few remotely accessible networking platforms practical to use in an educational context. Most of the available platforms are large-scale infrastructure and mainly dedicated to networking research more than education. Emulab [10] is an example of such platform. It is a large grid computing-oriented network emulator, based on a set of network links and computer hardware which can be interconnected following many topologies. Emulab provides an integrated access to three disparate experimental environments: simulated, emulated, and wide-area network testbeds. It unifies all three environments under a common user interface, and integrates the three into a common framework. Although the system is very powerful, the level of knowledge required to use the platform remains high. In addition, even if it is opened to education projects for higher level course, it is mostly dedicated to research projects.

Another example of large research platform is Planetlab [8], which is a distributed resource overlay network. In this system, each node is located across the world (mainly in research departments of Universities) and is linked using real Internet links. PlanetLab does not therefore support the emulation of various network condition, it's goal is more to provide the experimenter with computing infrastructure deployed the real Internet topology.

Finally, the Open Network Lab (ONL) [11] allows experimenter to build testbeds composed of several routers, using a graphical user interface. Its plugin architecture further allows experimenters to include their own packet processing routines, e.g. introduce delay). It allows the test of end-to-end applications and the evaluation of in-network traffic and present results to users in graphical form. The main drawback of the ONL approach is that it is using dedicated hardware, built around a gigabit ATM switch core network and dedicated FPGA. As such, the deployment of the platform remains a difficult and costly exercise.

3. IREEL Platform overview

The main goal of IREEL is to facilitate the experimental study of end-to-end transport protocols and applications, using an emulation system that allows packet impairments to be introduced. This allows experimenters to test protocols and applications under a wide range of network conditions, without the burden of an experimental setup to operate and maintain.

The platform design is extensible, enabling the addition of end-system applications, protocols and emulation systems. For instance, new applications such as voice over IP or new emulator such as a virtual topologies oriented network emulator like [12] can be supported.

One of the key design goal of IREEL was to provide a simple user interface requiring no or minimal training and supervision. To this end, experimenters access the platform via a web-based interface. A content management system and a modular experimentation framework allow experiment designers to design not only the experiment but also the corresponding user interface, which allow the experimenter to set the experiment parameters. For example an experiment consisting of a video transmission application over a lossy network could allow the user to set the video codec used and the network packet loss probability.

Each experimenter can set several experiments which are then queued in the system and processed sequentially and automatically by the platform. IREEL uses an asynchronous experiment completion notification service, which is currently implemented through email. When an experiment is completed, an email is sent to the experimenter. When logging back in, the experimenter can retrieve the experiment results. Each experiment is processed on a dedicated network where impairments (packet loss events and delay) are introduced by a network emulator. End-stations are exchanging packets using various applications and protocol, depending on experiment parameterization. At the end of the experiment, the results are provided to the experimenter in various types of outputs such as graphs, packet traces and high level application outputs.

The user can configure one or several experiments at a same time. This consists in setting up the emulation system with a proper network impairments models, choosing and configuring the application and/or transport protocol

mechanisms. In addition, the experiment is carried out over the remote platform when resources are available. Once the experiment achieved, the user has access to a report with the results provided in various complementary forms.
The experiment results consist of a set of complementary data, such as packet traces (at sender and receiver side) and performance measurements graphs (e.g., throughput, loss). To enhance the advantage of using real protocols and distributed applications, it is also possible to receive a record of applicative results. For instance, in the context of multimedia, the resulting (altered) audio or video is provided. Using these results and according to a lab description, the experimenter is able to compare and understand the performance impact on the end-host application of various network states, network technologies and protocol mechanisms.

4. IREEL Architecture

The IREEL system is composed of four different entities: Front-End, Experiment Server; End-systems, and Network Emulator., as illustrated on Figure 1.

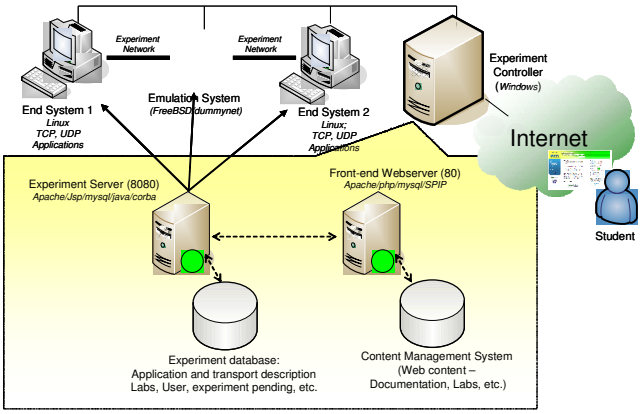
The **Front-End** provides the web user interface from which the experimenter is able to login, manage his account and set up a new experiment. Each experiment set-up page allows the user to choose and configure the applications, transport protocols and network impairments to be applied on each experiment flow. The Front-End further allows the user to configure what type of experiment results and how they will be presented. Depending on the type of the experiment programmed, a selection of standard results is automatically selected. The Front End is implemented using a standard web server, and a content management system.

IREEL uses the Shibboleth web Single Sign-On (SSO) technology from the Internet2 effort to manage user authentication across or within organizational boundaries. Shibboleth allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner. As IREEL was developed in the context of the EuQoS[1] EU project, the current implementation of IREEL is part of the SWITCH AAI Swiss university web ring. It allows students of SWITCH AAI participating institutions to use their institution authentication credential and transparently use the IREEL platform. This allows IREEL to be seamlessly integrated in other e-learning platform from these institutions, such as WebCT.

The **Experiment Server** keeps the main state of the platform. It manages the experiment queue as several users can log-in and simultaneously setup experiments; it receives the experiment configuration from the Front-End via a database. It processes it and dispatches the network emulator and applications parameters respectively to the emulation systems and end-system controllers.

The E**nd Systems** are hosts which execute the experiments communicating entities and collect traces and measurement. End Systems execute the real software and protocol stacks which are being experimented with.

The **Emulation System** is implemented with a single host equipped with multiple interfaces. It relays packets from one interface to another and applies packet impairments. The current implementation of this functionality is with FreeBSD/Dummynet [9], with a high-level abstraction allowing other emulation systems to be used easily. Three types of network emulation scenarios are available to the experimenter: static impairments (e.g. stable values over time), timed scenarii or other specific emulator scenarii, for instance, a particular satellite system. The emulator then sets up the different impairment channels before the beginning of the experiment (static emulation) or control the emulation parameter in real-time for a timed scenario.



1. Implementation of IREEL.

Experiment design process

The experiment designer (i.e., the professor in charge of the teaching course) designs and builds hands-on sessions based on one or more experiment. This process is composed of several tasks, which including writing the experiment controller in the JAVA programming language, and the results presentation webpage snippets, and the lecture material explaining the experiment. Templates are provided to assist the experiment designers in these tasks.

[1] http://www.euqos.org/

The burden of this experiment design process is furthermore over the number of time the experiment can be reused effortlessly by many classes across different institutions.

5. IREEL Experiment Process

The experimenter only interacts directly with the Front-End which provides a unique interface for the whole experiment system. The use of this system is mainly structured into four phases: user login, experiment configuration, experiment execution, and results collection.

The user login process is dictated by the Shibboleth login management. The system then proposes a set of predefined experiments scenarii. The experimenter then selects one of these scenarii, and the Front-End provides the experimenter with various parameters from the different software levels implemented (i.e. application, transport and network), according to the selected software associated to the current experiment.

The experimenter then configures the various experiment components and parameters values, which are stored in the Experiment Repository. Note that the controller can be busy with another experiment at the time the new experiment is started. In this case, the experiments are queued and will be processed when the resources become available.

When sufficient resources are available, the controllers retrieve the network emulator and end-systems configurations for the first experiment in the queue and send them to the end systems and emulation controllers. The experiment is then carried out in real-time. A timeout is used to limit the maximum experiment duration, in such a way to avoid freezing the platform if a problem occurs with a specific experiment.

Upon the experiment completion, the experiment results are stored into the repository and an email is sent to the experimenter. The experimenter can then access the results by logging back in.

6. Current experiments and results available on the platform

We have currently implemented a number of experiments, which are being used in several contexts such as networking courses at the University of New South Wales, and other institutions. In this section we describe a few of these experiments

Three main types of results are available from experiments conducted using IREEL, namely application level outputs, packet traces and traces analysis graphs. Application-level output is application-dependent and allows experimenters for example to evaluate the subjective, user-perceived QoS delivered by the application. Packet traces are the output of a packet sniffer and allow the experimenter to study the exchange of packets which took place during the experiment. Finally a set of standardized graphs built from the packet traces which allow various performances metric to be evaluated (e.g. time sequence diagram for TCP traffic) can be generated by the platform. An Example of such graph is illustrated on Fig. 3.

Here are some examples of experiments and the associated outputs:

- *Ping* is the classic networking application typically used to test networks and in particular the reachability of a given IP address. The results of this experiment provided by the IREEL platform are a text file containing the ping program output (typically round-trip time and number of lost probe packets), and packet traces. Packet traces, for all experiments, are provided both in textual mode (where packets fields are interpreted by the *tcpdump* program, but also in binary form to allow experimenters to save traces and re-use them in other software for further analysis (e.g *tcptrace* or *ethereal*).
- *THTTP* (Test HTTP) and is an experiment used to study and compare the behavior of different versions of the HTTP protocol. This experiment page configuration allows the experimenter to select between different HTTP protocol versions and to specify a number of pictures/object included in a web page requested. The experiment output presented to the user details the size and downloading time of each object.
- *Iperf* is a programmable traffic generator. With this experiment, the experimenter can configure some transport-level parameters such as socket buffer size, transport protocol used (TCP or UDP). The experiment results consist of the Iperf application output (bandwidth experienced, packet loss statistics, etc), and the network-level output packet traces as explained bellow. This experiment is for example typically used to illustrate a class explaining the TCP protocol behavior.
- *Video Streaming* is a video streaming application where a video stream is sent from one end system to other. The experimenter can select the video codec and the transport protocol used. The result of this experiment, as presented to the user, is two video

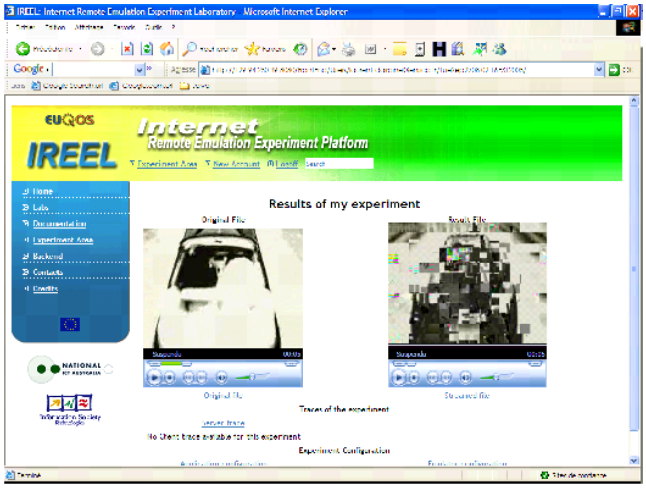
files: the original and the received video file transmitted through the network. This is illustrated on figure 2. This experiment allows students to visualize for example the impact of different packet loss probability over different video codec and transport protocol. A similar experiment could be used by video codec designers to evaluate their design under various network conditions.

7. Conclusion

In this paper, we have proposed IREEL[2], a remote networking laboratory that allows users to experiments with real networked applications and protocols. This platform consists of a remote controlled network emulator system and end-stations providing applications and protocol mechanisms opened to test with. The platform is strongly configurable allowing other protocols and application to be integrated. Control on the emulation system, and on the end systems will be given to experimenter in order to perform experiments through a web interface. This platform is currently tested and improved in the context of the dissemination activity of the EuQoS European Integrated Project.

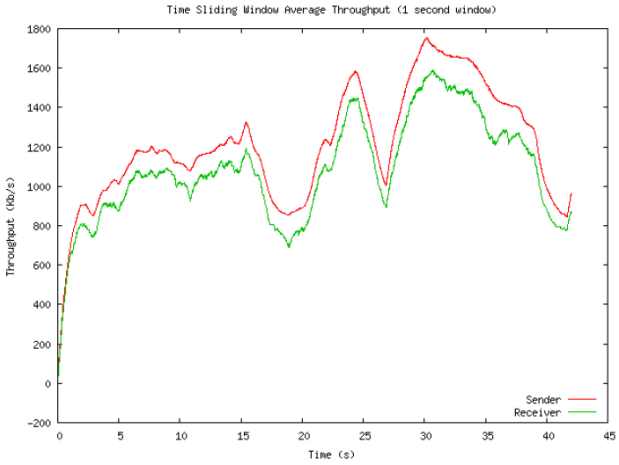
8. Acknowledgments

This research work has been conducted in the framework of the EuQoS European project (http://www.euqos.org) and has been financially supported by National ICT Australia (NICTA).



2. Example of application results for video over lossy network.

[2]A first version of IREEL can be accessed at http://ireel.npc.nicta.com.au



3. Example of throughput results measured at the sender and receiver side for a video over a lossy network.

9. References

[1] Opnet technologies - http://www.opnet.com, 2001.

[2] Omnet ++ - http://www.omnetpp.org, 2003.

[3] F. Arnal,et Al.. Reliable multicast transport protocols performances in emulated satellite environment taking into account an adaptive physical layer for the geocast system. In *International Workshop of COST Actions 272 and 280*, 2003.

[4] M. Carson and D. Santay. Nist net: A linux-based network emulation tool. *ACM Computer Communication Review*, 2003.

[5] K. Fall. Network emulation in the vint/ns simulator. In IEEE, editor, *Fourth Symposium on Computers and Communications*, 1999.

[6] M. Gineste, H. Thalmensy, L. Dairaine, P. Senac, and M. Diaz. Active emulation of a DVB-RCS satellite link in an end-to-end QoS-oriented heterogeneous network. In *23rd AIAA International Communications Satellite Systems Conference (ICSSC)*, 2005.

[7] T. Perennou, E. Conchon, L. Dairaine, and M. Diaz. Two-stage wireless network emulation. In *Proceedings of the Workshop on Challenges of Mobility (WCM 2004)*, Aug. 2004.

[8] L. Peterson, T. Anderson, D. Culler, and R. T. A blueprint for introducing disruptive technology into the internet. In *First Workshop on Hot Topics in Networking (HotNets-I)*, 2002.

[9] L. Rizzo. Dummynet: a simple approach to the evaluation of protocols. *ACM Computer Communication Review*, 1997.

[10] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. H., C. Barb, and A. Joglekar. An integrated experimental environment for

distributed systems and networks. In *In Proc. of the 5th Symposium on Operating Systems Design and Implementation*, Dec. 2002.

[11] K. Wong. Open network lab demonstration. In *In Proc. of Infocom 2005*, 2005.

[12] M. Zec and M. Mikuc. Operating system support for integrated network emulation in imunes. In *First Workshop on Operating System and Architectural Support for the on demand IT InfraStructure*, Boston, USA, 2004.

[13] P. Zheng and L. M. Nil. Empower: A network emulator for wireline and wireless networks. In *IEEE Infocom*, San Francisco, 2003