Experimental Methodology For Wireless Networks

Diego Dujovne, Thierry Turletti, Walid Dabbous

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Experimental Methodology For Wireless Networks

Diego Dujovne — Thierry Turletti — Walid Dabbous

N° 6667

October 2008

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6667--FR+ENG

INRIA
SOPHIA ANTIPOLIS

Experimental Methodology For Wireless Networks

Diego Dujovne*, Thierry Turletti†, Walid Dabbous‡

Thème COM — Systèmes communicants
Projet PLANETE

Rapport de recherche n° 6667 — October 2008 — 12 pages

Abstract: The adoption of new protocols and network mechanisms requires an experimentation phase which is complex to deploy, and even more on wireless networks. Generally, ad-hoc tools are developed for the specific needs on a experimental scenario to capture, analize and store statistical data. Because there is no common experimental methodology within the research community, and research publications do not provide often the experimental conditions, it is almost impossible for another researcher to reproduce the same experimental scenario. Within this research report, we define a network evaluation methodology to simplify the deployment of wireless experiments and their analysis. The proposed method uses a modular approach for the different experimentation phases. We also illustrate this methodology by an example.

Key-words: platform, experimental methodology, wireless networks, measurement

* diego.dujovne@sophia.inria.fr
† turletti@sophia.inria.fr
‡ walid.dabbous@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Une Méthodologie pour les Expérimentations Réseaux Sans Fil

Résumé : L'adoption des nouveaux protocoles et mécanismes réseaux nécessite une phase d'expérimentation préalable qui est complexe à mettre en œuvre, en particulier pour les réseaux sans fil. Le plus souvent, des outils ad hoc sont développés pour les besoins spécifiques au scénario expérimental afin de capturer les statistiques, les analyser et les stocker. Comme il n'existe pas de méthodologie partagée par l'ensemble des expérimentateurs, et que le plus souvent, les articles de recherche ne contiennent pas assez de détails sur le matériel, les logiciels et les conditions expérimentales, il est quasiment impossible pour un autre chercheur de reproduire le même scénario expérimental. Dans ce rapport de recherche, nous définissons une méthodologie d'évaluation réseau pour simplifier la mise en œuvre des expérimentations sans fil et leur analyse. La méthode proposée utilise une approche modulaire pour les différentes phases de l'expérimentation. Nous illustrons cette méthodologie par un exemple.

Mots-clés : Méthodologie d'évaluation, mesures, plateformes expérimentales, réseaux sans fil.

1 Introduction

The validation of new network protocols is a complex task requiring mathematical analysis, simulation and experimentation with networking platforms. Experimentation platforms can be classified in two types: *Emulation testbeds*, which offer controlled conditions for the experiments [1], [2] and *Real Overlays* aiming to do experimentation on top of production networks, which offer access through virtualization to real working conditions [3], [4].

In real overlays, the traffic generated by an experiment shares real paths and restrictions with other users' traffic. Since networking conditions vary with time, experimental results obtained will also vary. To take into account these varying conditions, it is important to use a common methodology to first establish the experimental setup, executing the experiment, capturing, processing and storing the results. Provided that a strict methodology is followed, it would be easier for researchers to compare between experiments which were executed under the same setup from one instance of an experiment to the following. For example, if the load increases, it is interesting to be able to check whether the protocol or application under test will react (or not) to this change. In order to analyze this possible reaction, the traffic variation load must be registered.

The required methodology should concern the main objective of networking experimentations: the validation of an algorithm, protocol or application under realistic conditions. Unlike emulation testbeds, whose aim is controlling network conditions so that one is able to reproduce the same experiment, real overlays provide no control on network conditions. Basically real overlays will allow to repeat experiments with the same setup in a different time or place and to analyze in detail the effect of varying network conditions on the performance on the protocol under study.

From this analysis, several important items arise. The layout (or the setup) of the experiment must be well defined; this is critical if part of the experiment is executed within a wireless network since not only traffic load or delay can change with time, but also transmission conditions due to environmental factors. All the participating devices (including stations, access points, routers, switches, links) have specific configuration which must be saved in order to reproduce these conditions. During the execution of the experiment, the network variations must be monitored and registered too, in order to correlate them with the captured data. Another important item is the data capture process: on the one hand, this process describes all algorithms and statistical functions to obtain the results; on the other hand, raw data processing extracts parameters that can be used to compare the different instances of the same experiment between them. Finally, all these items must be classified and stored in an efficient way to enable easy access to the data. In a later stage, this stored data can be used to extract new results, and to build new experiments based on the same setup.

In this paper, we propose an experimentation methodology to validate network experiments on real overlay platforms. This methodology aims to enable researchers to follow a concrete structure, defining a common set of experimental steps, which involves the definition, execution, processing, analysis and storage of the statistics obtained by experimentation. Section 2 describes the different steps of the experimental methodology. Section

3 details the design issues and structure of an experimentation toolkit that supports the proposed experimentation methodology. This toolkit is being implemented at INRIA and its description is the main contribution of this paper. Section 4 presents briefly the related work, and section 5 concludes the paper.

2 Experimental Methodology

In this section, we propose a methodology for running experiments on top of real overlays such as PlanetLab or OneLab. Such overlays provide the infrastructure to deploy operating system images, support monitoring and clock synchronization, transfer packet traces and allow remote access to the stations.

In this paper, we define a *run* as the atomic experimentation unit, and an *experiment* as a group of *runs*. *Runs* are executed in different times and under the influence of potentially varying network conditions.

Our experimental methodology is composed of a sequence of the six following steps, as illustrated on fig 1.

Layout Definition: This step includes the description of the environment, the hardware and software components used, and the configuration of each of the participating devices, such as nodes, routers, access points and links. In order to achieve reproducibility, the user must be able to rebuild the same experimentation layout in another place or at another moment by loading and executing the same group of steps.

Parameter Configuration: With all the devices already configured and tested, the experimentation dynamics must be defined in detail: what data to capture and when, when to generate traffic and what kind of traffic pattern, how many times the experiment must be repeated, and at what timescale. The timescale is an important item since the networking conditions can change according to time, for example.

Multiple Runs and Capture: This step consists of the experiment itself. The *runs* are executed as many times as defined in the former step, following the predefined sequence. All the devices must have their time bases synchronized, to execute the tasks within schedule and to timestamp the captured packets as they traverse the network. During this stage, raw data from the network is acquired and stored to be further processed. Realtime monitoring can be used here to check the evolution of key parameters like traffic load and packet loss during the experiment, so as to discover anomalies or divergences before the processing step. To our knowledge, there are no such tools that support multiple *runs* available for Planetlab or Onelab overlays.

Traces Processing: This step performs offline processing of the captured data. It includes synchronizing the packet timestamps from the packet traces, correlating and detecting missing packets, filtering and extracting parameters from the stored statistics.

Analysis: This step aims to present the experimental results. Once the data has been processed, the results must take a human readable form as a graphical representation of the data. Timelines, flowcharts, bar graphs and curves are typical examples of these representations.

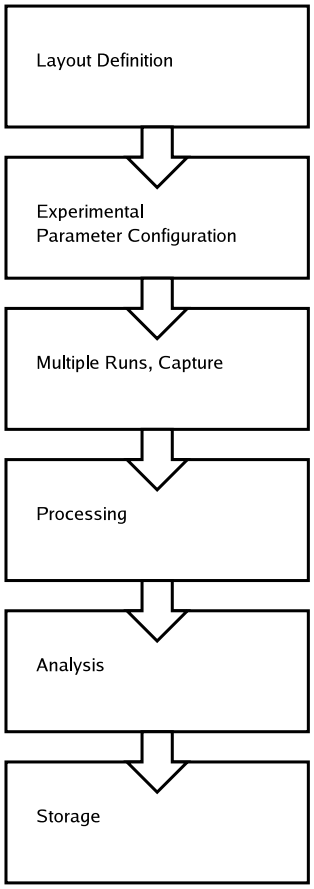

Figure 1: Experimental Methodology

Packing and Storage: Here the data is classified, organized and stored in an easily recoverable way. The package includes the raw data, network layout, system configuration setup and processed results. This will enable researchers to configure the same layout and setup to execute a new experiment, and also to have the whole data for the experiment definition.

3 Experimentation Toolkit

In this section we outline the architecture of the toolkit that implements the experimental methodology discussed in the previous section. Basically, the main objective is to keep a simple interface for researchers, so they can concentrate exclusively on the experiments.

3.1 Design Requirements

The experimentation toolkit should satisfy the following requirements.

Time Synchronization: It is necessary to synchronize stations to schedule events on the network. Packets and logs must have timestamps to correlate events for the merging step. Time synchronization can be achieved through NTP or using beacon received on each wireless node. Each of the events (e.g., the reception of a packet or the generation of a report from an interface) occurs at a certain time that will be used later to build the database.

Standardized Data Structure: Raw data can not be used to load the database unless the parameters can be extracted and adapted to a common format. This common format has two main objectives: first, to represent data in a structured manner, so as to access each of the fields; and second, to adjust to the search and processing capabilities of the database engine. For example, some data formats are standardized, like tcpdump files, BGP tables, flow export data from Cisco routers, and SNMP reports. On the other hand, network interfaces can produce non-standard data formats. Each of these sources requires a parser and interpreter to load the information on the database.

Realtime Monitoring: This feature is required so that a *run* may be aborted when necessary (e.g. after hardware failures or unexpected instabilities). To enable realtime monitoring, only a few key parameters along with their bounds must be supervised in order not to add a too much measurement overhead on the overlay network.

Modularity: The toolkit should be extendable in order that the user can add new modules without damaging nor interfering the activity of the other modules. The whole system may benefit from integration of well known and debugged tools. For example, the integration of ethereal can simplify the packet capture and interpretation steps. It is important not to reinvent the wheel and to simplify the maintenance and upgrade tasks of this tool. Wrappers should be developed for external tools whose interface do not fit on the capture requirement formats, as for example, *athstats* from the Madwifi [12] driver.

Open-Source Availability: It is important that the toolkit be developed under a collaborative philosophy in order to benefit from possible contributions and improvements (e.g. new networking environments and layouts) from the user community. Free access to the source code makes the data processing transparent and allows the user to modify the behavior of the tool. We believe that the networking community can highly benefit from a tool with the above features. Its widespread use should allow the exchange of compatible data between researchers to cross-validate the experimental results.

3.2 Hardware virtualization

In this subsection we define the elements composing the hardware platform where experiments are run. These elements are:

- *Nodes*: a node is microprocessor based device with one or more interfaces to connect to other nodes. E.g. a station, a router, or an access point.

- *Links*: a link is communication channel that connects two or more nodes. For example, a wireless channel, an optical link or a twisted pair cable.
- *Probes*: a probe is a device that listens to the traffic on a specific point in the network, and stores this data for further transmission.

Each of the nodes is connected to Planetlab or Onelab, and is running to manage the hardware resources and to create virtual nodes. Each of the virtual nodes executes an operating system, a network stack, and the application under test.

Probes can run on the same node than applications or they can be created as standalone nodes with capture and storage software running, e.g. a probe can consist of a notebook, an ethernet interface and a wireless card. In the case probes and applications share the same nodes, processing done by the probes should be kept at the minimum in order not to overload the system. As a consequence, only monitoring data will be transferred on realtime and on a different medium in order not to interfere with experiments. At the end of the experiment, all the raw and processed data including the layout and configuration information should be stored and packed.

3.3 Structure of the Database

We propose to organize all data using tables on a relational database. Five different types of tables will be used:

1. **Protocol table**: Contains the headers of the packets matched with that protocol, e.g.: ETHERNET and PRISM tables.
2. **Node table**: Contains the node identification and node-related statistics obtained from the discovery or from user-made layout information for each node on the network, e.g.: INTRANET_NODES and WIRELESS_NODES tables.
3. **Interface table**: Contains the parameters classified by interface from the node table, e.g.: WIRELESS_INTERFACES and ETHERNET_INTERFACES tables.
4. **Flows table**: This type of table lists the flows extracted from the protocol tables and their characteristics, e.g.: TCP_FLOWS and RTP_FLOWS tables.
5. **Source table**: this type of table identifies all data sources and their characteristics, e.g.: LOCAL_SOURCES and REMOTE_SOURCES tables.

Each *run* corresponds to one separate database. Protocol tables are configured with the protocol fields as columns. Each entry on a protocol table stores the fields corresponding to a packet header matching the protocol.

The proposed methodology allows to tradeoff preprocessing with classification and analysis tasks. Furthermore, SQL provides a human readable, unambiguous and powerful method to control searching and filtering criteria on the database. For example, from the packets

table, an AE module can extract the node list, and a different AE module can create the flow table.

An important advantage of this approach relies on the use of a single language to perform the queries on the database, which simplifies the analysis methodology. This solves one the most difficult problems with measurement studies: the ability to fully understand how the data was interpreted, and the ability to compare it with new experimental results.

3.4 Description of Modules

We propose a modular architecture where each module has one specific task to fulfill. These modules are coordinated by a task dispatcher that executes each module within a programmable schedule.

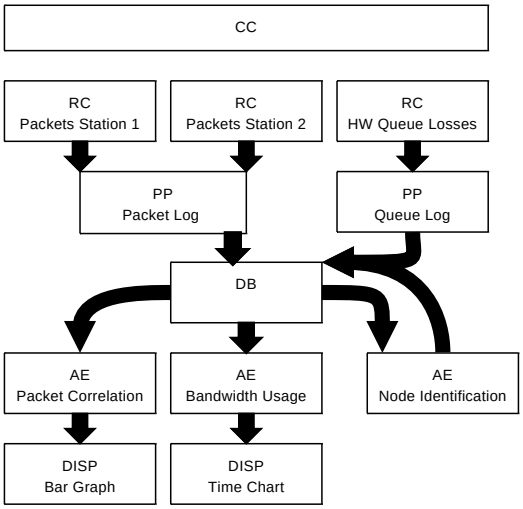

Figure 2: Relationships between modules

Fig 2 illustrates the relationships between the following modules:

Remote Capture modules (RC): One RC module runs on each of the probes and logs the events during the predetermined experimentation period. For example, if the RC module is in charge of packet capturing, a raw packet data file is created.

Preprocessing modules (PP): These modules run on the central processing station and read event logs from RC modules for data interpretation and storage on the database. For example, if the event log is a packet log, the PP module interprets the file and generates a description of the packets, using Packed Description Meta Language (PDML) [13]. Then, a classification engine parses this PDML file to load the database. There is one type of PP for each type of capture source. The PP connects as a client to the database server to store the preprocessed data.

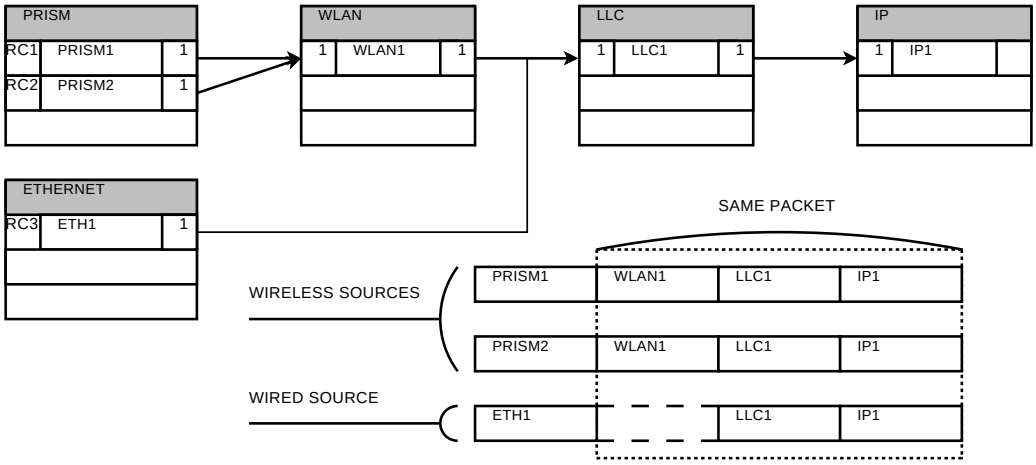

Figure 3: An example of database structure

Database module (DB): This module receives and stores the data from the PP modules using a standard structured query language (SQL) [14]. This module is a database server that interacts with the PP and the AE modules. The stored data can be used afterwards in different ways, and in particular to: 1/ verify the validity of experimental results; 2/ repeat the same setup in a new experiment and obtain new packet logs from the stations; and 3/ analyze the processed data using different criteria.

Analysis Engine modules (AE): These modules connect to the database and perform queries using SQL. The query results are analyzed and post-processed. The output is sent to the display (DISP) modules or goes back to the database to create node, flow, interface and/or source tables.

Central Coordination module (CC): This module is the dispatcher that generates the sequence of tasks (called experimentation sequence) to be executed on each of the stations. The experimentation sequence is created by the user to control the applications on the experimental nodes.

Display module (DISP): This module is able to create graphics, tables or any other representations from the outputs of the AE modules.

3.5 Database Structure for an Example

In this section, we describe an example of packet log processing. We assume the presence of three packet capture RC modules that generate three different lists of captured packets. These lists are processed by the same PP module, which dissects each packet on each of the lists and identifies the fields to record them on the corresponding tables.

Fig 3 illustrates the original packet structure and the final destination of the headers on each of the protocol tables. We can note that the corresponding protocol tables are: PRISM,

containing the reception parameters for the packet; WLAN, containing the addresses and protocol specific data; Logical Link Control (LLC); and IP.

Now let us assume that another probe has captured the same packet. The merging function should verify if the packet is already present on the database, through the comparison of capture timestamps and verification using a hash on the packet payload. If the payload hash has the same value and if the difference with the former timestamp is within the clock drift window, one can deduce that it is the same packet. Then, a new entry is added on the PRISM protocol table, and the index on this entry points to the former WLAN protocol entry.

Furthermore, let us assume that there is a new probe that captured the same packet: a server connected to a LAN in a remote location. A probe logs this event on its own packet capture file. At the end of the experiment, the log file from the remote RC module is processed by the packet log PP module. Since the packet already exists on the database, this event will be saved on the ETHERNET protocol table with a different timestamp. The index will identify the same IP payload. To obtain this behavior, the packet log PP module calculates a hash on the invariant part of the packet, i.e. in this case, the IP packet and its payload. We use this mechanism because the timestamp method is not valid here. Indeed, the packet has traversed the sniffed node and the delay has no relationship with the propagation delay on a Wireless LAN.

Finally, the AE module can execute a filtering rule that selects all the WLAN and Ethernet headers referencing the same packet. The results from this query will list the number of the times the packet was seen on the network.

4 Related Work

Specific workflow management tools exist for grid applications, as it was shown by Jia Yu et al. [10]. Several experimentation management tools exist in the context of the Emulab project [11], which are designed for controlled experimentation environments and they cover experiment definition using NS2 compatible scripting, control and data storage. On the "real overlays" side, there are few proposals, like Plush [5], and Weevil [6] both oriented to deployment, remote execution and monitoring of distributed applications on overlays. However, they do not include the processing and analysis stages, which are indispensable to display the experimental results [7]. On our side, we have developed WisMon, a wireless probe that supports monitoring, data processing and analysis on wireless experimentation platforms [8]. The experience acquired with WisMon was very helpful in the definition of the experimental methodology proposed in this paper.

To the best of our knowledge, there is not yet a proposal of a full experimental methodology combining all the different steps for capture, processing, analysis and storage.

5 Conclusion

We have presented a new methodology for experimentation in real overlay networks. We believe that such an experimental methodology is crucial to characterize the traffic, execute experiments under common methods and promote the exchange experimentation data between research teams. We have put emphasis on the fact that the processing and analysis stages will be of great help to researchers by simplifying their task so that they concentrate on the experiment itself. We have also defined the architectural description of a tool that fulfills the requirements of the methodology. The modular approach simplifies the expansion and improvement of the tool, and the database engine provides structured storage, flexible search and filtering functions. The combination of preprocessing modules, database storage and postprocessing modules creates a balance between the raw data on one side and the experimental results on the other side. We are implementing such toolkit and our aim is to build a validation environment for wireless networking experiments.

References

[1] `http://www.emulab.net/`[Last visited: October 2008]

[2] Raychaudhuri D., Seskar I., Ott M., Ganu S., Ramachandran K., Kremo H., Siracusa R., Liu H. and Singh M., "Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols", WCNC'05, March 2005

[3] `http://www.planet-lab.org/`[Last visited: October 2008]

[4] `http://onelab.eu`[Last visited: October 2008]

[5] Albrecht, J., Tuttle, C., Snoeren, A. C., and Vahdat, A. 2006. "PlanetLab application management using plush". SIGOPS Oper. Syst. Rev. 40, 1, Jan. 2006, pp. 33-40.

[6] Wang, Y., Rutherford, M. J., Carzaniga, A., and Wolf, A. L. 2005. "Automating experimentation on distributed testbeds". In Proceedings of the 20th IEEE/ACM international Conference on Automated Software Engineering (Long Beach, CA, USA, November 07 - 11, 2005). ASE '05. ACM Press, New York, NY, 164-173.

[7] Andersen D., Feamster N., "Challenges and Opportunities in Internet Data Mining". Parallel Data Laboratory, Carnegie Mellon University, Research Report CMU-PDL-06-102, Jan. 2006.

[8] `http://planete.inria.fr/software/WisMon/` [Last visited: October 2008]

[9] `http://www.paris-traceroute.net/` [Last visited: October 2008]

[10] Yu J. and Buyya R., "A Taxonomy of Workflow Management Systems for Grid Computing", Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005.

[11] Eide E., Stoller L. and Lepreau J.: "An Experimentation Workbench for Replayable Networking Research", Fourth USENIX Symposium on Networked Systems Design and Implementation (NSDI '07), Cambridge, MA, Apr. 2007.

[12] `http://www.madwifi.org`[Last visited: October 2008]

[13] `http://www.nbee.org/`[Last visited: October 2008]

[14] Codd, E.F. "The Relational Model for Database Management Version 2", Addison Wesley, 1990.

Contents

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399