A FRAMEWORK FOR THE AUTOMATION OF DISCRETE EVENT SIMULATION EXPERIMENTS

by

Bryan C. Ward

A Proposal Submitted to the Honors Council

For Honors in the Department of Computer Science

October 22, 2010

Approved: _____

Dr. L. Felipe Perrone
Thesis Advisor

_____

Dr. Stephen Guattery
Chair, Department of Computer Science

1 Introduction

Scientists and engineers seek to quantify the behavior of many different systems ranging from processes in the natural and physical world to engineered machines. In many of these applications, we seek to quantify the effect of changes to the system. With modern computers, these systems can be simulated in a virtual world and tested to understand their behavior. The use of simulation is particularly important in understanding systems which are expensive to prototype and build, dangerous to test, or otherwise difficult to observe.

One type of computer simulation is called **Discrete Event Simulation (DES)**. In such simulations, the system is described and modeled through a chronological sequence of events. The computer simulation models the behavior of these events and their effect on the system. Discrete event simulation usually involves **stochastic processes** or **random variables** to model random behavior in a system. Discrete event simulation is useful in many different applications, such as the simulation of computer networks.

When setting up a simulation, one must first create a **model** of the behavior of the system under investigation. A model consists of a mathematical description of the behavior of the system which is configured through parameters known as **factors** to the models. **Levels** are defined to be the values assigned to these factors. A **design point** is defined to be a set of valid factors and associated levels. Finally, a collection of design points compose a **simulation experiment**. A design point can be simulated to produce a large collection of quantitative statistical samples of metrics of interest which were observed during the course of simulation. This process is executed for all design points in the simulation experiment to understand the relationship between factors, levels, and performance metrics through rigorous statistical analysis.

The simulation workflow follows a standard sequence of steps. These steps are: observe the system, model the system, simulate the system, process the simulation results, and analyze the results to draw conclusions. When this workflow is applied properly, the results are **credible** in that they are reproduceable and are believed to accurately reflect the real world. Kurkowski et al. [5] demonstrated that many of the steps necessary in proper simulation methodology and statistical analysis are often skipped or conducted carelessly thereby compromising the credibility of the results. As a result, in a recent article I co-authored with Perrone et al. [8] we claimed that *"The level of complexity of rigorous simulation methodology requires more from [the simulation user] than they are capable of handling without additional support from*

software tools."

Software tools can guide the user through the complex process of proper simulation workflow by providing several functions. First, they ensure proper simulation methodology is applied to produce accurate results. Second, automation of the simulation execution can reduce the simulation runtime thereby reducing the overall time spent in computation. Third, software tools can help users analyze results, as well as make these results publicly available to researchers around the world via the world wide web. Lastly, automation tools allow users to focus their efforts on modeling the system or understanding results instead of managing simulation execution. In an educational setting this streamlined workflow can make simulation more accessible particularly in an undergraduate computer science course. In my thesis project I propose to develop a framework which integrates the functionality of many features of previous automation tools and can help guide a user to more credible results.

2 Background

The current state of the network simulation community in particular has been described by Kurkowski et al. [5] and Pawlikowski [6] as a "Crisis of Credibility." Users of different network simulators are presenting results which are not repeatable and often have mistakes in their methodology or presentation. What is even more concerning is that these flawed results have been published in peer reviewed articles. Many researchers [5, 6] have shown that credibility issues often arise from improper simulation workflow. Consequently, several software packages have been developed to automate individual steps in this workflow.

A software package called CostGlue was developed to aid the user in storing large amounts of simulation data [9]. The tool also facilitates the sharing of data between different simulation users and allows for replication of experimental results. The ability to share the results in this fashion is critical for the credibility of the results. Another similar software package called ns2measure was developed to ease the data collection and storage process within the network simulator ns-2 [2].

Other software packages, such as the Akaroa project by Pawlikowski [6], have been developed to automate the execution of simulation experiments while applying rigorous statistical methods. Akaroa aids the user in running long simulations on a collection of computers. Statistical samples of metrics of interest are collected from

independent simulations of the same design point and aggregated on a central server. By collecting samples from independant simulations of the same design point, the Akaroa project is able to reduce the time taken to collect enough samples to estimate metrics within a specified confidence interval. The server is then able to terminate all remote simulations. This methodology is known as Multiple Replications in Parallel (MRIP).

The SWAN-Tools project [7] in which I was involved was one of the first attempts to provide a more complete solution to the various issues that undermine credibility. SWAN-Tools was developed for use with the Simulator for Wireless Ad Hoc Networks (SWAN). The tool guides the user through all the steps of a proper simulation experiment, and demonstrates many features which tools for simulation automation need to contain. SWAN-Tools helps the user to create valid experiments and run independent simulations in parallel across many physical computers. Also, the tool aids the user in the data analysis by presenting results to be viewed in a web browser, to be downloaded and used with a statistics package, or to be graphically presented using proper plotting techniques via a web based interface. Lastly, the tool makes the results available via a website to which any scholarly article could be linked. The lack of flexibility in this tool is its major shortcoming. It was built exclusively for use with SWAN, and used a simulation model which was hard-coded into the tool. These constraints limit the potential uses for the tool.

These existing tools demonstrate important functionality: output processing, output storage, distributed execution, rigorous statistical methods, and a guiding user interface. My research will build upon the contributions of these projects and show how their functionality can be integrated into a single framework. I will also investigate the problem of building a framework which is flexible enough to be adapted for use with different simulators.

3 Project Description

The framework I will create will include several components. The first component of the framework will be responsible for interfacing with the user. This can be done through reading user generated input files which describe the experimental design. This first component will be able to read this file and extract the necessary information to be able to compute all of the design points in the experiment described by the input files.

Since each of the design points is independent of one another, they can be executed at the same time across several computers. My framework will use the MRIP methodology to execute the simulations in parallel across the available computers. A central server will aggregate the statistical samples collected during simulation execution. This in turn will speed up the execution of the experiment allowing users to explore larger experimental design spaces faster.

In order to use the MRIP methodology, the central server must include functionality for run control. This component will include functionality to determine when simulations are done executing, and will then notify the correct simulations to terminate. Simulations will be terminated when enough samples have been collected to estimate a metric within a user specified confidence level. In this computation, rigorous statistical processes must be applied. In particular, samples collected during the **transient** or the time while the simulator "warms up" from its initial state to a **steady-state** should be discarded. This process is known as **data deletion** and is important to ensure unbiased estimations of metrics.

These features will be made accessible through different interfaces for two kinds of users. The power user will have complete control of the framework through the manual editing of configuration files and the use of the command line interface (or terminal). Alternatively, users who are less experienced or potentially intimidated by the command line interface have the option to run an experiment through a web browser directed to a specific URL. This web interface will also provide functionality similar to that which was provided in SWAN-Tools.

In my thesis, I will discuss how my framework interoperates with two very different simulators. This will provide insight on the challenges of the integration effort required to set up the framework for use with a new simulator. The cases I will study are as follows: First I will integrate the framework with a simple queue simulator which I designed for easy interoperation. Second, I will focus on the integration of the framework with ns-3, which will pose several challenges due to the integration of code and configuration in the simulator. The comparative analysis of these two cases will document my primary scientific contribution to researchers who employ computer simulation in their research.

4 Methodology

The framework I will develop will employ several technologies and techniques to improve runtime efficiency while reducing time spent in the development. It will be implemented using the asynchronous, event-driven, network programming framework Twisted [3]. This framework performs well for these types of networked applications, and the architecture of the framework lends itself to an accelerated development cycle due to extensive built-in network communication functionality. I used Twisted in my recent work during the summer of 2010, so I am comfortable with it and understand how to employ many of its different features.

My framework will allow for dispatching simulations to be executed on remote machines to speed up the experiment. A central server called the **Experiment Execution Manager (EEM)**, will be responsible for the computation of all of the design points, as well as managing all of the connected clients which are simulating individual design points. This communication between the clients and the EEM will be done via the network, as seen in Figure 1.

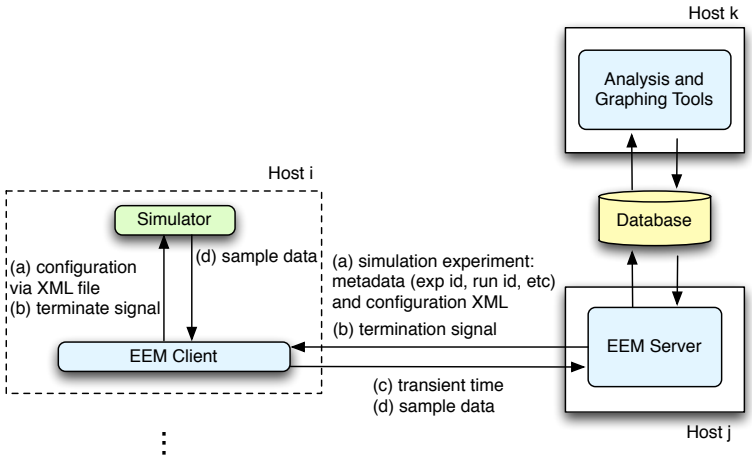

Figure 1: Architecture of the framework with respect to inter-process communication.

To support different simulators with different input and output formats the proposed framework will employ several XML-based languages (eXtensible Markup Language) which I developed with Andrew Hallagan ('11) and Dr. Perrone [4] in an independent study last year and later presented at a poster session. These languages will allow the user to configure the experiment through a few simple configuration files, as well as allow the server to communicate a simulation configuration to a processing

client. Because different simulators have different input formats and specifications, plugins or additional modules may be required to translate the XML simulation configuration to the necessary format for the given simulator.

The framework will also have to parse the output from the simulator to extract the results. A **simulation client** process will manage the execution of the simulator itself and will listen for results from the simulator as they are made accessible. The simulation client will then forward the extracted results to the EEM for persistent storage. In the case that the simulator does not support the intermediate output of statistical samples, results will be parsed and sent when the simulation terminates.

As shown in Figure 1, the EEM will store results in a relational database as they are made accessible. Using a database for storing results has several advantages. First and foremost, the use of a relational database provides persistent storage even after the process terminates or the server restarts. A database also controls access to its data, which will prevent users from compromising the integrity of the collected results. This project will use the database engine PostgreSQL [10] for its support for handing XML documents more easily than similar alternatives.

A suite of tools will aid the user in analyzing the results collected and stored in the database. Included with these tools will be a web-based plot generation page. This tool will ensure that plots generated will have confidence intervals on their estimations since the literature has shown this to be a common oversight in published results. Additionally, the framework will present the data to be downloaded in a standard format for use with other common statistical analysis packages. The framework will also provide an Application Programming Interface (API), so that developers can easily extract data for use in their own custom analysis tools.

5 Conclusion

The complexity of execution of proper simulation methodology has led to an overall lack of credibility observed in recently published simulation articles. This problem can be addressed through the automation of proper simulation workflow. My framework will integrate many of the functions seen in existing automation tools. Additionally, it will use an extensible architecture which loosely couples the automation framework and the specifics of the implementation of the simulator. This will allow for the framework to potentially be used with other simulators after a few modules

are developed to communicate with the simulator of interest.

The code developed during my thesis implementation will be released as open source software under a General Public License, and it will eventually be packaged with ns-3 as part of the work conducted under Dr. Perrone's recent NSF grant [1]. This will give the project visibility to ns-3 users around the world and allow for the continued development of the software upon completion of my thesis. Additionally, this visibility will hopefully promote the use of my framework with other simulators.

References

[1] Frameworks for ns-3. Available at
<http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0958142>
[Accessed September 1, 2010].

[2] Cicconetti, C., E. Mingozzi, and G. Stea (2006). An integrated framework for enabling effective data collection and statistical analysis with ns-2. In *Proc. of the 2006 workshop on ns-2: the IP network simulator (WNS2 06)*, pp. 11.

[3] Fettig, A. (2005). *Twisted Network Programming Essentials*. O'Reilly Media.

[4] Hallagan, A., B. Ward, and L. F. Perrone (2010). An experiment automation framework for ns-3. In *Proceedings of the 3rd International Conference on Simulation Tools and Techniques (SIMUTools 2010)*. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST), Brussels, Belgium.

[5] Kurkowski, S., T. Camp, and M. Colagrosso (2005). Manet simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev. 9*(4), 50–61.

[6] Pawlikowski, K. (1999). Akaroa2: Exploiting network computing by distributing stochastic simulation. In *Proc. of the 1999 European Simulation Multiconference*, Warsaw, Poland, pp. 175–181.

[7] Perrone, L., C. Kenna, and B. Ward (2008). Enhancing the credibility of wireless network simulations with experiment automation. In *Proc. of the 2008 IEEE Intl. Conf. on Wireless & Mobile Computing, Networking and Communications (WiMob '08)*, pp. 631–637.

[8] Perrone, L. F., C. Cicconetti, G. Stea, and B. C. Ward (2009). On the automation of computer network simulators. In *Proc. of the 2nd Intl. Conf. on Simulation Tools and Techniques (SIMUTools '09)*, pp. 1–10.

[9] Savić, D., M. Pustišek, and F. Potortì(2006). A tool for packaging and exchanging simulation results. In *Proc. of the International Conference on Performance Evaluation Methodologies and Tools (Valuetools)*, Pisa, Italy, pp. 60. ACM.

[10] Worsley, J. C. and J. D. Drake (2002). *Practical PostgreSQL*. O'Reilly Media.