An Experiment Automation Framework for ns-3

Andrew W. Hallagan, Bryan C. Ward and Luiz Felipe Perrone
Department of Computer Science
Bucknell University
Lewisburg, PA, U.S.A. 17837
{andrew.hallagan,bryan.ward,perrone}@bucknell.edu

Abstract

Recent studies have indicated that tools that automate the execution of simulation experiments can serve to enhance both the usability of network simulators and the credibility of the studies developed with them. In this poster, we present the architecture for an experiment automation framework for the ns-3 network simulator. The architecture was designed with two specific goals in mind. First, it raises the level of abstraction of the ns-3 user interface to make the simulator easier to use in large scale experimental studies. Second, it provides functionalities to guide the user along known correct methodologies for modeling and simulation, thereby increasing the confidence one can have in the correctness of the experimental results.

XML Modeling

- **Model Description Language:**
  - Consists of the overarching "model" which itself is composed of a collection of sub-models.
  - Model nesting corresponds closely to ns-3 class hierarchy.
- **Experiment Description Language:**
  - Lists each experimental factor in turn and describes some list of parameter values which the factor will take on.
  - Provides constructs for building different types of lists.
  - Describes a factorial experiment design.
- **Restriction Description Language:**
  - Prunes factorial experiment design.
  - Specifies parameters to occur in tandem and in exclusion.

These languages will all be validated by RELAX NG schemas as they are more expressive and allow more sophisticated constructs than the W3C XML Schema standard.

RELAX NG Compact Syntax

RELAX NG can describe *attribute-element* constraints that are impossible to express in W3C XML Schema. In this case, the content of the element `speed_variable` depends on the value of the `type` attribute.

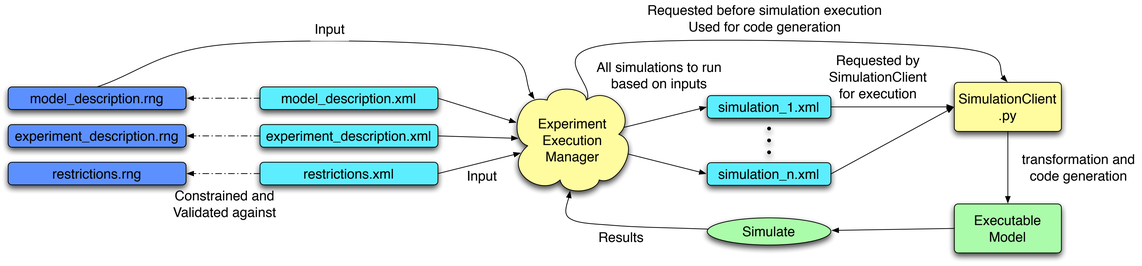
```
element speed-variable = { RandomVariable }
RandomVariable = Uniform | Exponential
# Define each distribution type
Uniform = attribute type { "uniform" },
          attribute lo { text },
          attribute hi { text }

Exponential = attribute type { "exponential" },
              attribute mean { text },
              attribute bound { text }
```

The corresponding XML is concise, intuitive, and readable (and also impossible to validate against any W3C XML Schema definition):

```
<speed-variable type="uniform" lo="0" hi="1"/>
```

When the value of the `type` attribute is set to "uniform," the contents are validated against a different definition than if the `type` attribute is set to "exponential."

Experiment Automation Framework Architecture



Architecture of a flexible experiment automation framework.

Experiment Manager Design

- **Multiple Replications in Parallel (MRIP)**
  - Execute simulations in parallel across networked machines.
  - Global experiment manager determines execution time for all simulations.
- **Built around HTTP based webservice**
  - Developed in Django.
  - Easily interface with database through common Object Relational Mapper (ORM).
  - Easy to build API using standard HTTP libraries.
  - Not bound to a specific language or platform.

Defining a Grammar

A grammar for our XML languages serves two purposes:

- Defines correct semantic structure of any simulation description. The XML standard already provides the syntactical constraints.
- Provides the structures used in *data binding*: mapping XML elements to software object definitions in a language like Python. Changing the grammar definition of a particular model will result in an automatic change in the object definition of that model.

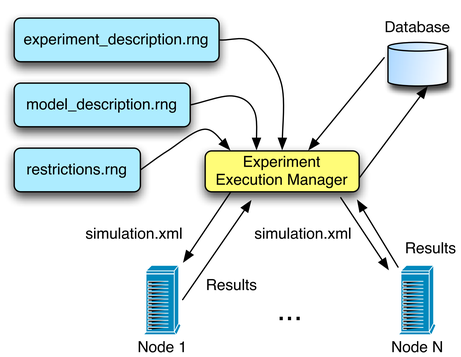
Code Generation

- Event-based parsing of XML documents uses less memory, but may require multiple passes in the case of object-oriented code generation, in order to update references correctly.
- RELAX NG Schemas provide the basis for generating various Python classes.
- Iterate through the final simulation description document, creating instances of these Python classes. Set all known constants where appropriate and add flags for object references impossible to identify.
- Re-iterate through this simulation description document again in order to update Python object references.

Simulation Client Design

- **Setup simulation environment**
  - Find or build simulator to match exact version of simulator specified by experiment manager.
  - Request Model Description RELAX NG Schema for code generation purposes.
- **Request Simulation**
  - Request simulation from experiment manager.
  - Generate ns-3 Python code from received XML simulation configuration.
- **Execute Simulation**
  - Periodically send collected samples to experiment manager.
  - Based upon experiment manager's response, continue simulation, or terminate.
  - Request new simulation upon termination.

Multiple Replications In Parallel (MRIP) Architecture



Architecture of the experiment execution manager which will simulate using the MRIP strategy.

User Interfaces

- **Web Based:**
  - Simple enough for even a novice user to work with.
  - Guides user through proper experimental design.
  - Gives user access to valid models and factor values.
  - Validates along the way to ensure no mistakes are made.
  - Browser based application makes the application cross platform capable.
- **Command Line Interface:**
  - Geared towards the more advanced user.
  - Gives user additional flexibility in configuration.
  - Driven by user developed XML files and command line scripts.
  - Values given are validated before the experiment can proceed.

Posterior Analysis

- **Tabular Results:**
  - Publish results to the web; increases credibility of results when experiments linked in publications.
  - Export results as CSV, allows users to use spreadsheets or other statistical analysis software.
- **Plotting Utility:**
  - Web based for cross platform compatibility.
  - Ensures plots accurately depict actual results.
  - encourages the use of confidence intervals.
  - Allows other users to explore results easily via a web browser.
  - Similar functionality as that found in ANSWER and SWAN-Tools.
- **RESTful API:**
  - Provides access to results to users developing custom scripts.

Related Work

[1] Andreozzi, M. M., Stea, G., and Vallati, C. A framework for large-scale simulations and output result analysis with ns-2. In *Proc. of the 2nd Intl. Conf. on Simulation Tools and Techniques (SIMUTools '09)* (2009), pp. 1–7.

[2] Murata, M., Lee, D., Mani, M., and Kawaguchi, K. Taxonomy of XML schema languages using formal language theory. *ACM Trans. Internet Tech. 5*, 4 (2005), 660–704.

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

[4] Perrone, L., Kenna, C., and Ward, B. 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)* (2008), pp. 631–637.

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

Acknowledgements

This project was started thanks to Bucknell University internal grants for the support of undergraduate research. The continued development of the framework described here is funded under NSF grant CNS-0958142.