An Automation Framework for ns-3

Dr. L. Felipe Perrone,
Bryan C. Ward, and
Andrew H. Hallagan

Department of Computer Science
Bucknell University

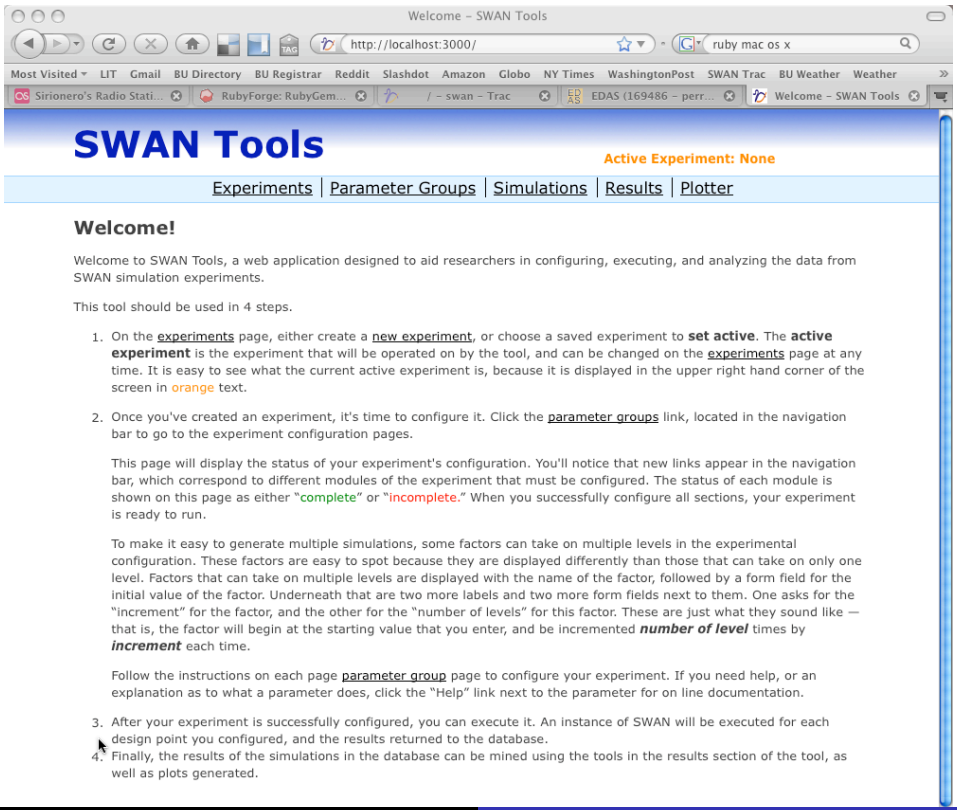
March 14, 2010

Motivation

- Network simulation is no easy business. One must:
  - Build a model that is consistent. (Lots of assumptions, not all valid.)
  - Describe the simulation model for a given simulator. (Unpleasant learning curves.)
  - Design experiments and carry them out. (Often a lot of compute time required to run them.)
  - Process a lot of output data using good methodologies. (Do people always know how to?)
- We could raise the level of abstraction on the user interface to network simulators. It would help both researchers (experts?) and students (certainly not experts).[3]

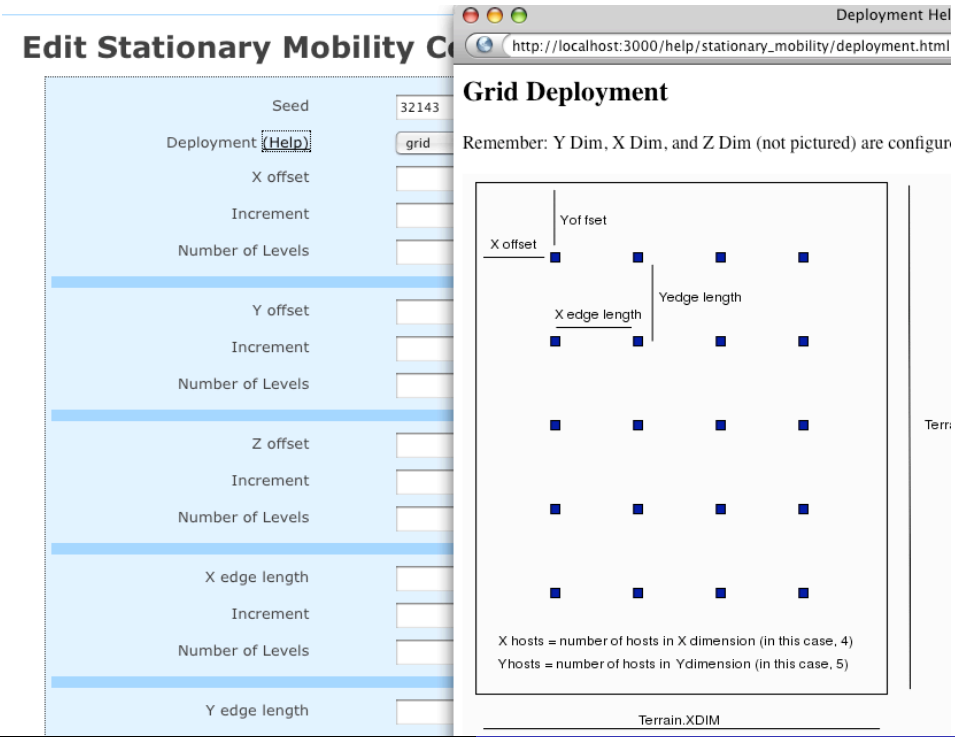
SWAN Tools

A web based application to enhance the usability of an SSF-based simulator for wireless ad hoc networks. SWAN Tools constrains the user to *do the right thing.*[4]



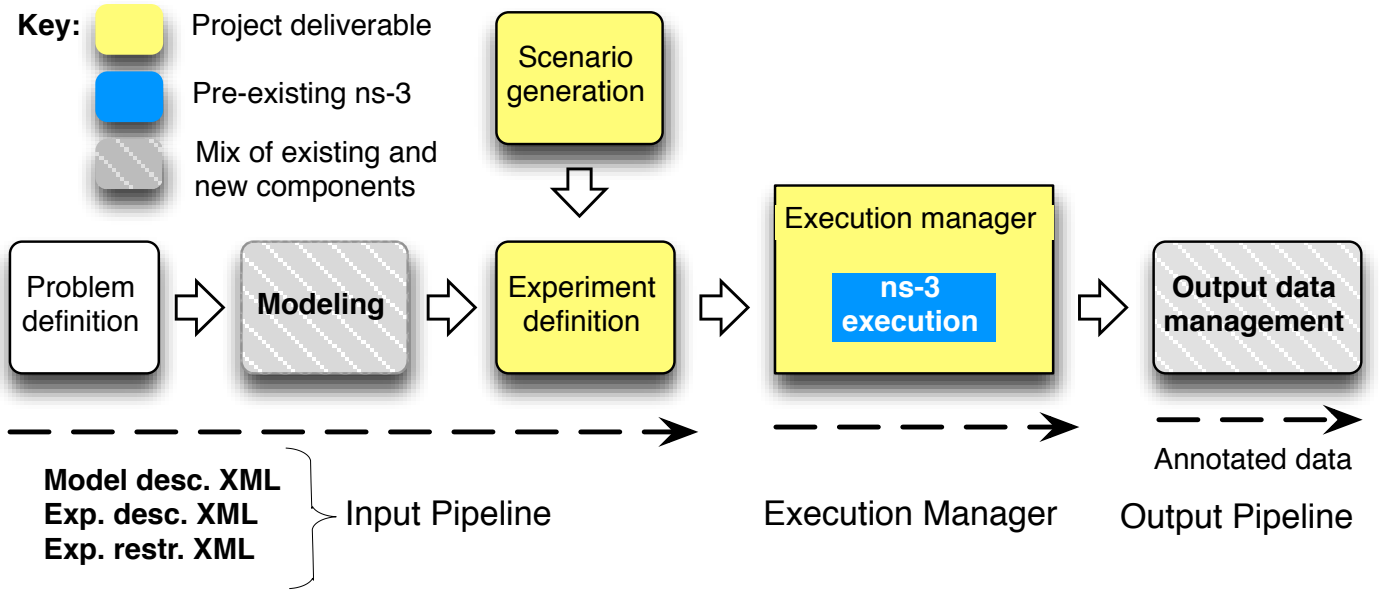
Model Configuration Interface

The *in browser* interface guides the user through model building without requiring any knowledge of the simulator's configuration language (DML). It also clarifies the semantics of model factors and helps with the *design of experiment*.



Lessons Learned from SWAN Tools

- User should be able to build custom models and not be constrained to use a standard default.
- Need to have different interfaces for power user and inexperienced user.
- Need to allow for finer controls in design of experiments.
- Using a database backend to store models, experiment descriptions, and simulation output data is essential.
- Using a web application framework speeds up our development.
- MRIP is a cheap way to speed up experiments. To exploit it better, we need to incorporate automatic run length and transient detection for data deletion.[2]


Key:
Project deliverable
Pre-existing ns-3
Mix of existing and new components
Scenario generation
Problem definition
Modeling
Experiment definition
Execution manager
ns-3 execution
Output data management
Annotated data
Model desc. XML
Exp. desc. XML
Exp. restr. XML
Input Pipeline
Execution Manager
Output Pipeline

XML Languages

- **Model Description Language:**
  - An overarching *model* built as a composition 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.

Validation of XML Input

- The W3C XML Schema standard can't handle situations we find with our languages (e.g., interleaving and element-attribute constraining).
- We're experimenting with RELAX NG (RNG) schemas [1], which are more expressive and allow more sophisticated constructs.

From XML to ns-3 Code

- We have prototype code that can translate from our XML model description language to ns-3 code (in C++ and Python). It works fine with simpler models.
- Using the levels defined for model factors written in XML has been complicated (data-binding issues). Code generation hasn't always worked.
- We're investigating the idea of having *augmented RNG schemas* that carry embedded code hints to aid in data binding and code generation. The schema goes through a compiler which generates a custom SAX parser to finish the job.

Model Description Document Example

```
<!-- element and attribute contents are left
     blank to be filled in later according
     to the experiment description document  -->
<mobility_model>
  <random_walk_2d>
    <pos_vector x="" y="" z=""/>
    <time units=""></time>
    <speed_variable></speed_variable>
  </random_walk_2d>
</mobility_model>
```

Experiment Description Document Example

```
<!-- id's can be arbitrary identifier strings -->
<parameter id="x-comp"
path="mobility_model.random_walk_2d.pos_vector.x">
  <!-- a list of arbitrary values -->
  <value>3</value>
  <value>7</value>
  <value>11.5</value>
</parameter>

<parameter id="pos_y"
path="mobility_model.random_walk_2d.pos_vector.y">
  <!-- a simple sequence -->
  <sequence start="2" delta="0.5" levels="3"/>
</parameter>
```

Restrictions Description Document Example

```
<restrictions>
  <one-to-one type="inclusive">
    <!-- sets identified by the id attributes
         in the experiment description document -->
    <set>x-comp</set>
    <set>pos_y</set>
  </one-to-one>
</restrictions>
```

Console vs. Web based Interface

- Console based interface gives the power user a more flexible interface.
- Advanced users can even script things to make running experiments easier.
- Web based interface will provide many of the same functions, but will help a novice user through the process.
- Both interfaces will be built upon the same backend and experiment manager.
- Both interfaces will ensure the user is not providing invalid models or factors.

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.

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.

REST

- Style of building a web service.
- Uses the common HTTP protocol.
- Client - Server model:
  - Client not responsible for data storage.
  - Clients make requests to server for a representation of a resource or object.
- Used extensively in web development and libraries are available.
- Easy to add authentication - could be used to ensure the validity of results.

Multiple Replications in Parallel

- Execute the same simulation on multiple machines.
- After transient has expired, collect samples from each simulation client.
- Terminate all simulations after desired confidence for given metric(s) is reached.

MRIP Diagram

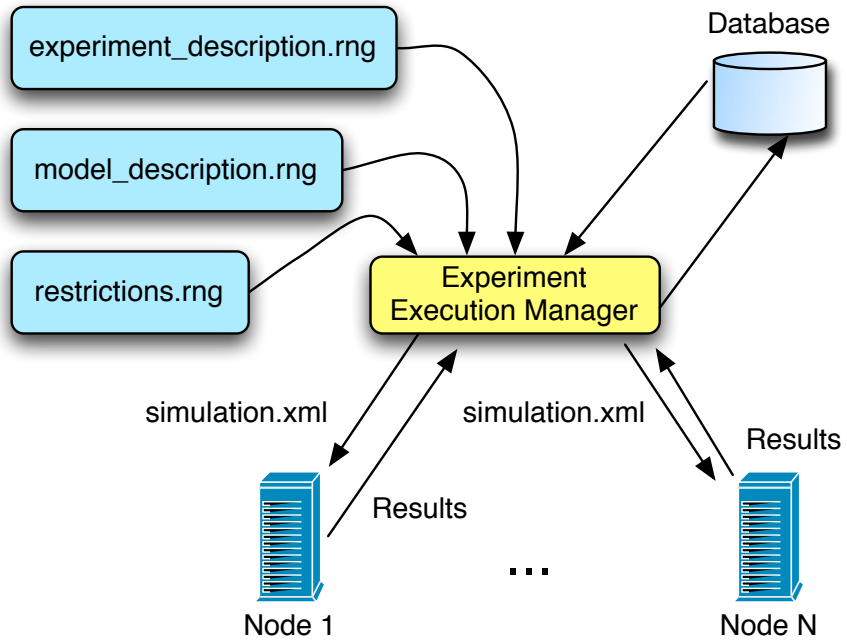


Figure: Architecture of the experiment execution manager which will simulate using the MRIP strategy.

Traditional Relational Databases (RDBS)

- Data in rows and columns.
- Well understood.
- Common libraries for interfacing and abstracting the RDBS.
- Limiting when data does not easily conform to a predefined structure.

NoSQL Databases

- Schema-Free/document-oriented
  - Data does not have to fit into specific columns.
- Newer and not as well developed or supported.
- Support in many frameworks is being actively developed.

Why NoSQL for Simulation Automation?

As discovered in SWAN Tools, an automation framework needs to be flexibile.

- Different simulation models have different factors.
- These are difficult to store in traditional RDBS.
- Different factors have different types (e.g. int,float,enum).
- Difficult to store these different types in RDBS.
- Even more difficult to query for things when you do.

Prototype Stage

- Language for the description of experiments.
- Generation of individual points in the design of experiments space and translation into simulation script.
- Command-line user interface.

Investigation/development stage

- Server side component of web application.
- MRIP controller: collect the results from individual simulations, preprocess them, and store in central database.
- Strategies to record information about the experiment in database.
- End of transient detection in estimated metrics.

Coming Up Later

- Checking of completeness and consistency for custom built component-based model.
- In browser interface for model creation & configuration, control of running experiments, and visualization of output data.
- Enable interoperability with external data processing and visualization tools. Rene this capability throughout the remaining years of the program.

Makoto Murata, Dongwon Lee, Murali Mani, and Kohsuke Kawaguchi.

Taxonomy of XML schema languages using formal language theory.
ACM Trans. Internet Tech., 5(4):660–704, 2005.

Krzysztof Pawlikowski.

Akaroa2: Exploiting network computing by distributing stochastic simulation.
In *Proc. of the 1999 European Simulation Multiconference*, pages 175–181, Warsaw, Poland, 1999.

L. Felipe Perrone, Claudio Cicconetti, Giovanni Stea, and Bryan C. Ward.

On the automation of computer network simulators.
In *Proc. of the 2nd Intl. Conf. on Simulation Tools and Techniques (SIMUTools '09)*, 2009.

L. Felipe Perrone, Christopher J. Kenna, and Bryan C. Ward.

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)*, pages 631–637, 2008.