Enhancing the Credibility of Wireless Network Simulations with Experiment Automation

Dr. L. Felipe Perrone perrone@bucknell.edu
Christopher J. Kenna cjk@cs.wm.edu
Bryan C. Ward bryan.ward@bucknell.edu
Dept. of Computer Science
Bucknell University, Lewisburg, PA, U.S.A.
Why we run simulations

We need to understand the technology before we can rely on it for mission-critical applications.

Performance can be quantified/estimated with computer simulation.
Credibility issues

Experiments published are not always reproducible.

• What was the version of the simulator used?
• What sub-models were used?
• Where to find the complete experimental set up?
Credibility issues

Output data is **unavailable** or **unreliable**.

- Papers publish a thin “slice” of experimental results.
- Methodology to compute the statistics of output data doesn’t conform to best practices.
- Plots without units on axes, legends on data series, and/or confidence intervals.
Credibility issues

We know **where** and **how** we’re failing.

- ...

We can use **automation solutions** to avoid problems.
Model Composition

Radio Propagation Channel

- Model: 2-ray ground reflection
- carrier_frequency: 2.4 GHz
- temperature: 290 K
- noise_figure: 10.0 dB
- ambient_noise_factor: 0
- system_loss: 1.0

Terrain

- Model: Flat
- xdim: 5,000 m
- ydim: 3,000 m
- zdim: 5.0 m
- boundary: wraparound

Mobility

- Model: Random waypoint
- min_speed: 5.0 m/s
- max_speed: 10.0 m/s
- pause_time: 65 s

Node Deployment

- Model: Random

Wireless Node 1

- Model: Host
- packet size: 512
- bit rate model: CBR
- bit rate: 3000 bps
- protocol_graph: wireless
SWAN Tools

A web-based application that guides the construction of the simulation experimental study and its data analysis. SWAN Tools is self-documenting: it allows experiments to be reproducible.
Web-based Interface to SWAN

• Control of the simulation process:
  • Experiment design
  • Configuration
  • Execution
  • Data analysis

• Remote access via a web browser.

• Browser talks to server which automates the simulation process.
Experiment Configuration

- User can work with multiple experiments.
- Each experiment is bound to a specific code base of the simulator (user uploaded).
- For each experiment, define number of runs for each simulation, random number generator seeds, length of transient time (for data deletion).
- User can’t change an ongoing or completed experiment.
Model Specification

• GUIs guide the user to enter all the parameters for all the sub-models employed.
• No default values for model parameters.
• Validates parameter data.
• Shows the units for each parameter and helps user to understand context.
• Experiment design builds list of levels for each parameter.
Simulations

• Generates all design points for the experiment.
• Creates a configuration file for each simulation run.
• Dispatches simulation runs for an experiment across multiple processors (multiple replications in parallel - MRIP).
Results

- User can see the raw data from each simulation run.
- Results are stored together with the configuration that was used to generate them.
- Uses established methodology to generate statistics (average, standard deviation, confidence intervals, etc.)
Plotter

• User can request the system to built plots from stored results.
• The user can define what the plot will include from the web browser. The system processes the relevant data, generates a plot that meets standards, and sends the results back to the browser for display or download.
Implementation

or

Why Ruby on Rails Rocks
It’s All about the Database
Database Interactions

- Migrations
  - Easy to add/remove columns or tables
  - Agile for developers

- Abstraction
  - Data accessible as class variables
  - Easily map relations between classes
Ruby

- Easy and powerful
- Metaprogramming
  - Dynamically create new methods
  - Flexibility
AJAX

- AJAX helpers provided by Ruby on Rails
- Easier interface
  - Reduced page loads and refreshes
- Component validation
  - Constrains user to enter only valid parameters
Embedded Ruby (ERb)

- **Views**
  - Dynamic content with embedded ruby in views
  - Access to RoR AJAX helper functions

- **DML Configuration**
  - Dynamically generate configuration files
  - Allows for a single global template
DML Configuration file

mobility [
  model "mobility.random" # import models for random movement
  deployment "random" # distribute mobiles uniformly in space
  period 1 # time interval between position updates in sec.
]

motion [
  type "waypoint" # use random waypoint mobility.
  mobid 1 # unique identifier for this pattern of movement
  pause_time 4 # pause time
  min_speed 5 # minimum node speed
  max_speed 30 # maximum node speed
] # end of random waypoint motion

...
Is this configuration interface safer?

Edit Mobility Configuration

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mobid</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>Mobility Type</strong></td>
<td>waypoint</td>
</tr>
<tr>
<td><strong>Pause Time (s)</strong></td>
<td>60</td>
</tr>
<tr>
<td><strong>Increment</strong></td>
<td>30</td>
</tr>
<tr>
<td><strong>Number of Levels</strong></td>
<td>4</td>
</tr>
<tr>
<td><strong>Min Speed (m/s)</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>Increment</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>Number of Levels</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>Max Speed (m/s)</strong></td>
<td>10</td>
</tr>
<tr>
<td><strong>Increment</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>Number of Levels</strong></td>
<td>2</td>
</tr>
</tbody>
</table>
DML Configuration file

mobility [
    model <%= self.model %> # import models for random movement
    deployment <%= self.deployment %> # distribute mobiles uniformly in space
    period <%= self.period %> # time interval between position updates in sec.
]

motion [
    type <%= self.motion_type %> # use random waypoint mobility.
    mobid <%= self.mob_id %> # unique identifier for this pattern of movement

    pause_time <%= self.pause_time %> # pause time
    min_speed <%= self.min_speed %> # minimum node speed
    max_speed <%= self.max_speed %> # maximum node speed
]

# end of random waypoint motion

...
Distributed Ruby (DRb)

Multiple Replications in Parallel (MRIP)

- Run on multiple systems
- Collect more samples by running more simulations
- All results reported to server process
- Statistics automatically generated – no user error
- Results stored on filesystem and in database
DRb and MRIP

Simulations → DRb Experiment Server → Database

Configurations → DRb Experiment Server → Configurations

Results → DRb Experiment Server → Results

Node 1

... → DRb Experiment Server →...

Node N
Future Work

- AJAX plotting utility.
- AJAX runtime queue management.
- Automatically determine length of transient and length of simulation run given a desired level of confidence.