Project Description: CI-ADDO-EN: Frameworks for ns-3

1 Overview of Proposed Computing Research Infrastructure

The University of Washington, Georgia Institute of Technology, and Bucknell University propose to extend the ns-3 discrete event network simulator (a free, open-source software project founded by an NSF CRI grant) with a 4-year program developing software frameworks for the simulator, under a community infrastructure award. The ns-3 project was started as a new community infrastructure project with funding from an active (2006–2010) NSF Community Research Infrastructure (CRI) program. Our proposed program will develop further the existing infrastructure and support its continued operation.

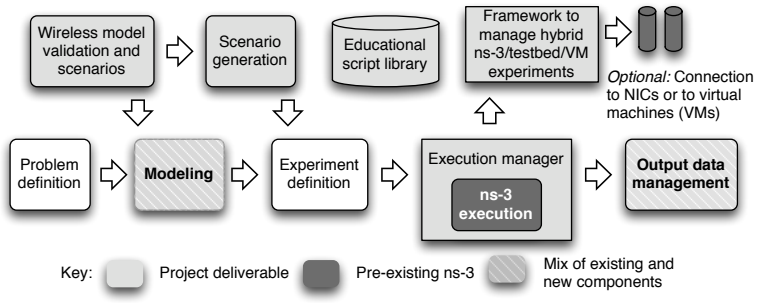


Figure 1: Simulation workflow as extended by our "Frameworks for ns-3" project.

Specifically, this program will develop new frameworks for ns-3. as shown in Figure 1, aimed at improving the ease of use and ability for users to construct *methodologically valid* simulations using ns-3, and to successfully blend the use of simulation with other research techniques such as testbeds and virtual machine environments. The new frameworks are:

1. **Automation:** The workflow of a typical simulation study requires the user's careful attention to a number of best practices in methodology so that the results are credible and reproducible by third parties. There are plenty of opportunities to inadvertently introduce errors in the process from model description to output data processing to the reporting of results. We propose to create an automation framework that will raise the level of abstraction for the user and provide a environment that is more conducive to the production of rigorous studies. Our automation framework will guide the user to produce valid network simulation models by composition and consistency checking, definition of experimental settings that exercise control over large-scale simulation studies, applying rigorous statistical methods for output data analysis and run control, and the reporting of resources that collect details required to reproduce the experiment.

2. **Scenario generation:** Published simulation results are often criticised for lack of exploration of a rich scenario space. However, scenario generation should be one of simulation's strengths, not weaknesses. Our framework will provide mechanisms to allow users to define wide ranges of simulation node topologies (including mobility and energy consumption considerations), application traffic, and channel environments, and to define and calculate some basic metrics on the generated topologies that allow users to verify that the topologies studied span a wide range across these metrics.

3. **Wireless validation and scenarios:** Wireless models are among the most popular models for simulation because of the advantages that simulation offers to researchers studying wireless systems. However, what is often lacking is validation of such models, as well as an easy ability to define and parameterize a rich set of wireless scenarios. We will extend the scenario framework introduced above to include wireless scenarios, and will also introduce validated models, documentation, and worked examples for the following wireless aspects not yet modeled or validated in ns-3:

empirically-derived channel models, multipath and multiple access interference, directional antennas, and multiple antenna (MIMO) systems.

4. **Virtualization:** We have already demonstrated several types of integration techniques for linking ns-3 with lightweight virtual machines. For instance, on a single server, it is possible to generate over one hundred lightweight machines each with its own private stack, and connect them together with ns-3. What is lacking, however, is some supporting software that eases the configuration and run-time management of such arrangements. We will develop glue software for a framework targeted towards blended ns-3/virtual machine experiments.

5. **Education:** Presently, network simulation is not widely used in computer network courses because it often shifts the focus away from the original learning objectives towards learning the simulation tools instead. We will create an extensible handbook and program repository that will show how to apply ns-3 and the frameworks we will create to the classroom teaching of a number of topics in computer networks. We expect that this resource will be a valuable addition to the tools at the disposal of instructors in electrical engineering and computer science.

Each of the above topics is probably worthy of its own dedicated infrastructure project. However, we again plan to leverage the open source community to continuously extend the frameworks we establish. We can already point to success stories in this regard with ns-3. For instance, we have focused in our current grant on building the core software infrastructure, along with a set of models that enable basic capabilities in many common areas of interest, such as a real-time simulation mode and a basic emulation capability, and a TCP/IP Internet stack for IPv4. What we have seen in many cases is that contributors from outside the NSF project (but part of the open source ns-3 project) have taken these basic capabilities, and extended them in interesting ways. For example, the research group at RWTH Aachen University that has built on our basic emulation capability to develop a novel "synchronized network emulation" capability that uses a Xen-based deployment of virtual machines whose clocks are locked to the simulation clock and are not constrained to run in real-time [WSHW08]. Another example is the research group at Louis Pasteur University that build a parallel IPv6-based Internet stack following our IPv4 architecture [MVM08b].

Therefore, in this CRI program, we plan to focus not on building out complete instantiations of any of these new frameworks, but rather on the framework itself with the goal that, over time, other contributors will extend what we have started. Using the educational framework as an example, what is most important for us is to focus on organizing the overall effort by starting a handbook, seeding it with a minimally sufficient number of examples, setting up a repository to collect educational scripts, setting up and maintaining a mailing list or wiki to coordinate the effort. As other instructors utilize what we have done, they can extend it in ways that we have not thought of or do not have the time for, and contribute back to the project. Our Frameworks project, funded by this CRI effort, would maintain the educational framework.

The second initiative is centered on the ongoing and continued maintenance of ns-3. A large-scale collaborative development project such as this requires coordination and maintenance by professional software developers whose full-time jobs are to interact with the user base, develop tutorials and documentation, review and merge code contributions, debug and test the software, and be responsible for the software release. We have experience over many years that large open source projects do not operate for free and that as projects grow, the need for dedicated software support is essential. Both of these activities (software maintenance, and development of supporting software infrastructure to broadly aid users) are activities that are not likely to be conducted by users of the simulator, and therefore are prime candidates for funding by the Community Research Infrastructure program.

The expected lifetime of the simulator with these extensions is ten to twenty years, provided that the software is maintained. The predecessor to ns-3, called ns-2, currently has been used for research for more than twelve years. It is important to emphasize that, due to the rapidly evolving computing infrastructure that is used to run the ns-3 software (namely, commodity servers and PCs running operating systems such as Linux, Microsoft Windows, and OS X), the software can rapidly fall out of date with systems (most notably compilers and libraries) and hardware (for instance, the emergence of multi-core servers) without continued maintenance. At least annual software maintenance releases that deal with such issues

are typically required to keep a software package like ns-2 or ns-3 viable for current operating system distributions, even if the software features are not extended. We expect that ns-3 will be useful beyond ns-2's lifetime because of the software engineering and code reviews that we enforce on the software core.

2 New Research and Education Opportunities

Simulation remains in heavy use for network research and education, because of its ease of use, reproducibility, availability, scalability, and ease of software development. The proposers believe that simulation will continue to provide a complementary and reinforcing role to analytical work and live network experiments. In fact, a primary goal of ns-3 software development has been to facilitate migration between simulation and network experiments of various types, and we describe below our past success and future plans in this regard.

As of this writing, we are three years into our existing effort to develop ns-3, and can point to several measures that suggest the impact that our project has already made. First, the simulator is already in widespread use, with over 4000 downloads of our released software per month since April 2009, a users mailing list with 300 subscribers averaging roughly 200 messages per month, a developers mailing list with over 900 subscribers and also roughly 200 messages per month, and use of ns-3 by several research groups. Second, we have successfully established this as an open source project, as witnessed by the number of contributions that have been made outside of the NSF-funded components. For instance, as of this writing, there are roughly 15-20 publicly announced extensions of ns-3 that are either in review for merging in the main ns-3 tree or under public development for future merging. Most of the software models for ns-3 are now written outside of the core maintenance team. We are delivering on our vision of building and maintaining a research-focused, community-maintained, open source network simulator for the networking research community.

However, the project is still young, and much work remains. To date, most of our focus has been on building and validating the core software and models for ns-3. This has established the basic ability for users to extend the simulator to perform their own research, but the learning curve is still steep. This is by design; the plan that we established in the first phase of the project was to prioritize a powerful low-level API that gives power users extreme flexibility to configure the tool, with the expectation that frameworks for ease of use can be added as overlay layers later. We have already begun to build this intermediate layer (sometimes called the "helper API"). However, we need to improve the ease of use and avoidance of misuse (i.e. misconfiguration or poorly conducted experiments) in the coming years. Adding these frameworks is one focus of this proposal.

The infrastructure extensions proposed herein will be fully integrated with the infrastructure (the ns-3 simulator) previously and currently funded under (in their final year of execution) under CNS-0551686 (University of Washington), CNS-0551378 (Georgia Institute of Technology), and CNS-0924385 (University of Washington). Below, we describe in detail our five proposed frameworks.

2.1 Automation

> **Summary of Deliverables:** The automation framework will consist of user interfaces, description languages, and tools that will help users of varying levels of expertise to produce more credible simulation experiments with ns-3. The functionality offered will enable the user to define, deploy, and control ns-3 simulation experiments that are methodologically valid and easy to reproduce by third parties. The framework will include tools for: model composition; structural validation of the model; configuration of model components; description, deployment, and control of experiments; output data processing and storage; and reporting of experimental setup. Although the framework will offer graphical user interfaces, more experienced users will be able to access automation functionality via the command-line.

In the last 10 years, a number of meta-studies of the network simulation literature have called into question the credibility of many publications. [CKC05, PJL02, Paw90] The systematic scrutiny of the literature exposed that many of the causes of the problems are *procedural*. All too often, published studies fail

to report the complete scenario necessary for a third party to replicate the experiment, use models composed of incompatible pieces or which fail to include components that have interdependencies, determine the length of simulation in manners inconsistent with the study's goals, and do not use best practices for output data collection, processing, and analysis. These problems arise from the fact that individuals in the community of users of network simulation do not always have the required combined expertise in the areas of simulation methodology and network modeling.

The automation framework we propose to construct will address the deficiencies identified in the literature by creating user interfaces for the simulator workflow that guide the user through the process in a systematic manner. Our prior work in SWAN Tools [PKW08] made important advances in this direction: it guided the user along some of the most important stages of the simulation workflow using a web-based interface. The user was able to connect to the system using a web browser and configure a network simulation experiment. This process entailed the assignment of a number of parameters in each sub-model of a predetermined scenario through interfaces that eliminated ambiguities and exposed the semantic of each configurable parameter. Once all sub-models were configured, the user could specify a list of values for each parameter. The system generated the cross product of these lists to compute the *design of experiment* space of the simulation study and each point in this space was inserted in an execution queue. An execution manager used this queue to dispatch simulation runs for each point in a cluster of distributed computers or in a multicore machine.

SWAN Tools was only a prototype, however, and one which was constructed for a simulator which cannot be publicly distributed due to licensing restrictions of its IEEE 802.11 models. The work we propose for this project will build upon the lessons we learned in the development of SWAN Tools and create a framework for ns-3. This new automation framework will incorporate the recommendations we made in [PCSW09], which were developed from the analysis of sizable body of literature. The architecture of the new framework will include three major components as illustrated in Figure 2: the *input pipeline*, the *execution manager*, and the *output pipeline* described in the following paragraphs. The figure shows the mapping of components of the automation framework to our overall project plan.

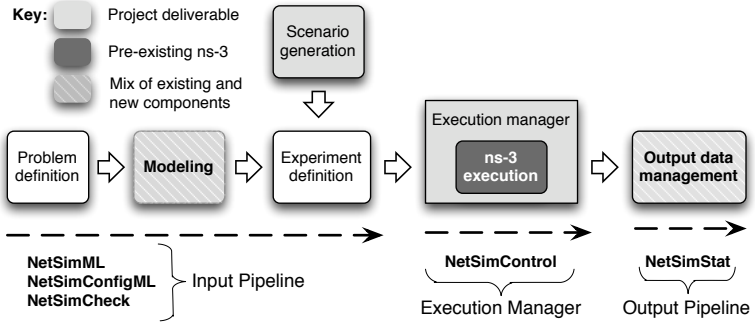


Figure 2: Mapping of automation components to the simulation workflow

1. **Model Description**. We will develop an XML-based language for describing network simulation models (NetSimML), which will be independent of the underlying network simulator. The design of this language will include the level of detail in scenario description that meet strict expectations of completeness and semantic correctness. The resulting language will support the component-based model construction paradigm, which is used in ns-3 and the network simulators based on SSF [CON02, LPN+01]. There have been several proposals for such languages, such as the one in [CEV03], for instance. This language will allow the experimenters to build network simulation models as compositions that draw elements a library of sub-models. As in SWAN Tools, users will be able to use web-based graphical user interfaces (GUIs) that generate syntactically correct model description files, such as those illustrated in Figure 3; this will enable access to an installation of the system over the world-wide web. In our first milestone, the GUIs will work with static model compositions (i.e., *scenarios*) that we will offer to the user; we will build GUIs to enable the dynamic construction of model compositions for a later milestone. The system will submit the model composition to a tool which will verify that criteria of interdependence and completeness of sub-models are satisfied (NetSimCheckStruct).

As illustrated in the model configuration interface in Figure 3, the framework will allow the user to enter parameters settings in each sub-model and also to inspect and configure any simulator's default settings not exposed in the user interfaces. Our system will ensure that any default parameter settings used in a simulation experiment become part of the recorded scenario; the framework will work with ns-3's configuration management system (ConfigStore) to obtain this information and save it for posterior use. It is important to note that to accommodate users with varying levels of expertise, our framework will allow users to bypass GUIs and to construct models descriptions with a text editor, invoking tools manually via the command-line.

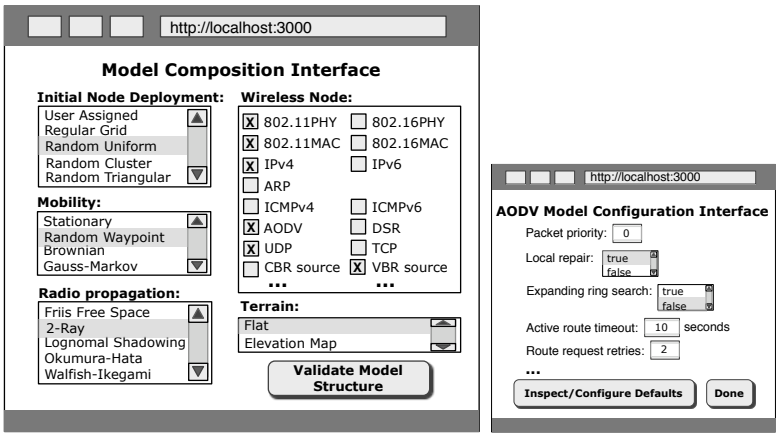


Figure 3: Sketches of web-based, graphical user interfaces for model composition and configuration.

2. **Experiment Description.** We will define a second XML-based language for the *description of network simulation experiments* (NetSimConfigML.) A NetSimConfigML file will refer to a fixed scenario described in NetSimML and will contain lists of the discrete values (or *vectors*) for the parameters that are to be varied in an experimental study. The framework will compute the cross product of these lists to define points in the *design of experiment* space of the study. Each of these points will correspond to one simulation that is executed multiple times with different seeds for the generation of random numbers. This description of the experiment will also identify the data that will be collected, identify the machines that will execute the experiment, list the desired confidence level for metrics to be estimated, etc.

   Figure 4 shows the input pipeline of the framework exposing details of our envisioned implementation with a few additional components. NetSimExpGen will use data from the description of the experiment in a NetSimConfigML file to generate the *experimental design points*. For each design point generated, NetSimExpGen will create a complete scenario description with the parameter settings from the design point, which will be fed into NetSimCheckParam for ensuring that all parameter values are within the ranges of values acceptable for each sub-model used.

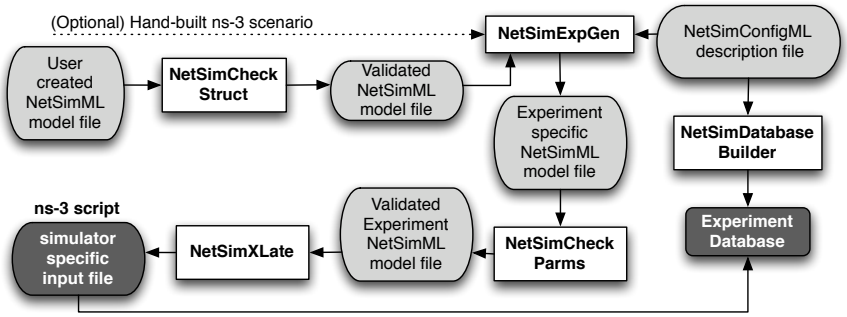


Figure 4: Detailed architecture of the input pipeline of the automation framework.

The output of NetSimCheckParam will be the validated description of the experiment design point, which will be converted by NetSimXLate into the corresponding ns-3 script. It should be noted that

an interesting feature of the architecture of this pipeline is that it will be possible to adapt it for interoperation of the framework with other underlying simulators that adhere to the philosophy of *split-level programming* described in [BEF+00], such as those based on the SSF standard. What would be required to support this interoperability is the writing of different translators to take the place of NetSimXLate.

The information generated by the user in model definition and experimental description stages will be organized in a relational database that will underlie the automation framework. In the development of SWAN Tools, we applied a lesson learned from SoS [GPPP02], which demonstrated that the database is an essential component in the implementation of functionality that allows experiments to be reproducible and credible. The database is used to aggregate all user input with output generated by simulation runs in a reliable and organized fashion. As illustrated in Figure 4, the NetSimDatabaseBuilder component will create the database schema for an experiment based on its description file in NetSimExpML.

INRIA is also building a framework for integrating ns-3 with testbeds using a higher-level description of experiments [LFH+09]; we will collaborate with them or leverage their work.

3. **Execution Management.** Some of the key aspects in the established methodology that are often neglected in published studies with network simulation involve the misuse of random number generators, the definition of simulation run length, the definition of the required number of runs to reach pre-determined levels of confidence in statistically estimated output data, and guaranteeing that data is collected only after the end of transients in steady-state simulations. Our framework will contain a module called NetSimControl that will interact with ns-3 for the purpose of launching replicated runs of a experimental design point in clusters or multicore machines. NetSimControl will interact with another framework component that handles output data processing (NetSimStat) to evaluate whether transients have abated and whether simulations have run long enough so that the desired confidence level has been reached in statistically estimated data. This functionality will be similar to what is implemented by Akaroa 2 [Paw03], a resource which we plan to leverage.

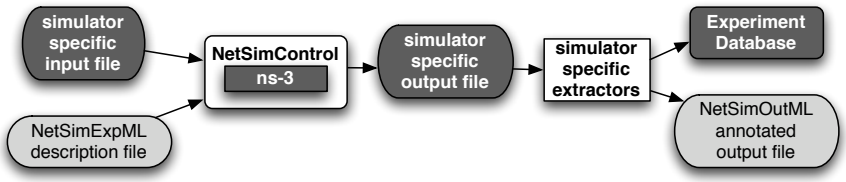


Figure 5: Detailed architecture of the Execution Manager of the automation framework.

ns-3 has an extensive tracing system to expose fine-grained output data of interest; our module will hook these trace sources to collect metrics of interest to the user. Once these metrics have been identified and isolated, they can be written according to the specification of a markup language (NetSimOutML) that we will define so that this output pipeline can interoperate with other data processing and visualization tools, as proposed by CostGlue [SPP06]. Additionally, this data will be inserted directly into the framework's experiment database. This database will accumulate records of the experimental output in direct association with the conditions that define the experiment, as was done in SWAN Tools and SoS [GPPP02].

4. **Output Data Processing and Reporting of Experimental Results.** Three modules will be implemented for the collection, processing, and persistent storage of simulation results: NetSimStat, NetSimQuery, and NetSimPresent, which comprise the output pipeline of the framework. Following the lessons we learned in the development on SWAN Tools, all these modules will work with the relational database that underlies the automation framework. NetSimStat will take the output produced by the ns-3, process it according to rigorous statistical methods, and insert it into the database associating output data with the experimental scenario. This will ensure that the results of the simulation study are not compromised by data mixups. NetSimQuery will make the database available

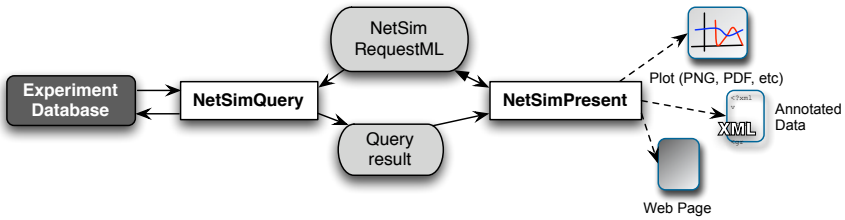


Figure 6: Detailed architecture of the output pipeline of the automation framework.

for access over the Internet with a web-based language (NetSimRequestML); this feature will overcome the constraints of reporting experimental results in published matter (conference proceedings and archival journals) and maximize the impact of scientific investigations. Finally, we will provide a web-based application for displaying the results of queries to the experiment database (NetSimPresent). We will leverage multiple resources in the implementation of NetSimPresent: our previous work in SWAN Tools for generating plots and annotated data, the functionality offered by ROOT [roo09], a framework for data analysis and presentation developed at CERN, and ANSWER [ASV09], a web-based application for mining experiment databases developed at the University of Pisa. We envision that NetSimPresent will offer output according to different user choices, among them plots in graphical formats for direct use in the preparation of documents, annotated data, or yet HTML formatted raw data for inspection via a web browser.

Educational Impact: We expect that this project will result in educational impact in two different scopes. In a narrower sense, it will create learning experiences for students involved in its research and development. The past development of SWAN and SWAN Tools, created a sustained involvement of undergraduates since 2001: these projects resulted in experiences for undergraduate that included internships, individual study courses, honors theses, and summer research. All but 3 of the 9 students who participated in these activities have discovered an interest for academia and went on to pursue graduate degrees in Computer Science.

In a broader sense, the automation tools we propose will produce as many benefits for students in computer networks courses as for the experienced researcher. As reported in[PCSW09], courses that use network simulation for performance analysis or for testing new ideas in protocol design often require that students create ad hoc solutions for automating experiments and/or for the statistical processing of output data. When the development of these ancillary programs is only tangential to the main educational goals of the activity, students end up having little time left to reflect on the results they obtain and to internalize the lessons to be learned. For these reasons, simulation is not used more often in computer networks courses. This automation framework will offer a friendlier and safer user interface that will serve to eliminate the students' misdirection of efforts and help them stay focused on the expected educational outcomes of their assignments.

2.2 Scenario Generation

Summary of Deliverables: Topology generation modules that produce random, empirically derived, and hierarchical topologies, including treatment for mobile topologies. Ability to annotate topologies with metadata about links or wireless channels. Scenarios for traffic generation. Integration with automation and XML-based configuration system for ns-3.

Another major focus of this development effort is to help simulation users to construct more realistic and meaningful simulation scenarios. For this discussion, we define a *scenario* to be the network topology to be modeled (the routers, end systems, communications links, queuing methods, etc), the protocol stack elements being modeled, the communication channel models, and the network traffic being modeled. All too often when conducting simulation–based research, the simulations used to show some interesting phenomenon in the network behavior are not nearly complex enough, with too little competing traffic, and unrealistic network topology models. A typical user of ns–3 is likely to be an expert in computer networking methodologies, but is unlikely to have extensive experience in network modeling. By enhancing

the ns–3 tool to include options to create arbitrarily large and complex topologies, with varying degrees of competing traffic and competing applications, we expect that the results obtained by researchers will be a much better representation of how the research results will perform when deployed. The work in this area will be in several separate focus areas, described below.

1. **Topology Generation** We will provide three methods for automatic network topology model generation, described below.

   (a) **Random Topology Generation**. There have been a number of networking research initiatives in the area of synthetic topology creation, as far back as 13 years ago with the *Georgia Tech Internet Topology Modeler* (*GT-ITM*) by Zegura et al. [ZCB96]. Later work by a number of researchers in analyzing and understanding the Internet topology (specifically work by Faloutsos et al.[FFF99]) led to a number of new and more capable tools, such as the *BRITE* tool [MLMB01]. These new tools randomly create topology graphs based on underlying probability distributions that are found in the actual Internet topology. Other tools, such as *skitter* [kcMM99], Lumeta [Lum07] and *Mercator* [GT00] take a more direct approach and use results from direct Internet topology measurement to construct topologies with similar characteristics using the measured distributions.

   These existing tools have proven to be quite useful to the research community, but have some drawbacks that we will address in this work.

      i. Virtually all of the tools are designed to work in a stand–alone mode, producing some output file format (ASCII text or other) that then must be imported into the simulation environment. In this work, we will integrate one or more of the existing random topology generation tools directly into the ns-3 code base. The first effort will focus on integrating the *BRITE* tool, and we will leverage our prior experience on porting *BRITE* to our prior *GTNetS* tool. By integrating the topology generation software directly into ns-3, we provide an easy method to execute a number of topology models with a single ns-3 executable instance.

      ii. None of the existing tools contain information regarding link bandwidth, link delay or link queuing methods, that is critically important for constructing simulated topologies. In addition to including existing topology modelers into our framework, we will provide extensions that randomly assign values for bandwidth, delay, and queue size with given probability distributions. Another approach would be to assign larger bandwidths to nodes with higher node degree values, since core routers with a large number of neighbors are more likely to have high–speed communication links.

   (b) **Recreating Measured Topologies**. A number of existing and prior efforts have made considerable progress in determining with high accuracy the actual Internet topology connectivity graphs at the autonomous system level. Using publicly available data from *Routeviews* [Mey] and *Rocketfuel* [SMW02] it is possible to construct a reasonably accurate model of the connectivity of the worldwide Internet infrastructure. Recent work by Dimitropoulos et al. [DKVR09] has resulted in a topology description that not only describes AS connectivity, but further provides annotations that specify peering agreements between the AS's. As part of this work, we will include C++ objects in the ns-3 code base that will construct the AS topology graph, with a specified level of detail. For example, a given scenario might only require tier one, tier two, and tier three providers with no end systems if the purpose of the simulation is to understand routing updates between ISP's under various failure conditions. Another scenario might need only particular subnetworks for tier three providers, when studying the effects of localized failure modes or localized misconfiguration.

   (c) **Artificially created topologies.** Not all simulation scenarios need the scale and accuracy as described in the previous section. To enable smaller scale scenarios, we will manually create a number of synthetic topology models can can be included in the ns-3 simulation with a single line of code. These will be parameterized to specify the relative size of the synthetic topology, the number of subnetworks, the number of core routers, and probability distributions for link bandwidth, link delay, and queuing methods. Further, we will provide simple command link

parameters for these, that will allow easy execution of a number of experiments to help ensure there is sufficient randomness and variation in the analyzed experiments.

To achieve this goal, we will provide ns-3 users with a large library of *stock topology* objects, with a common API to assign traffic generation and traffic sink applications to some or all of the topology leaf nodes. Depending on the user's specified requirements for size and complexity, a number of stock topology objects might be connected together in an hierarchical fashion to facilitate simulation scenarios with a high degree of variability in traffic mix, bottleneck link distribution, queuing methods, and link characteristics. By making it easier for users to perform a number of different and diverse experiments with a single ns-3 executable, simulator users will be more likely to perform meaningful experiments and produce more realistic conclusions from the simulation experiments.

2. **Wireless Channel Model Variability** All network simulation tools that support wireless networking, including ns-3 have detailed models of the performance of the wireless channel propagation between a transmitter and receivers. These models have varying levels of detail and accuracy. Some tools even provide capability to specify terrain characteristics in order to account for the effects of hills, trees, buildings, and other obstructions. While such additional modeling detail indeed results in superior and more accurate models, they are still lacking in realism. Several measurement studies, such as that by Reddy [RR07], Aguayo [ABB+04] and Kotz [KNG+04] show clearly that there is considerable variation in packet reception probability even under identical conditions. For example, Aguayo showed that nodes that are significantly farther away than other closer pairs inexplicably exhibit better signal strength and higher packet reception fraction. Such findings indicate that realistic wireless models must include a high degree of random variation in calculated path loss in order to produce statistically meaningful results. In the work proposed here, we will provide simple command-line argument processing that will allow users to specify the amount of noise (randomness) to include in the channel model, and make it easy for users to run a number of experiments with varying channel model characteristics. This work will be integrated with the wireless work to be introduced in the next subsection.

3. **Network Traffic Generation** Another common failing of typical simulation–based network experimentation is the lack of appropriate *competing traffic*. While it is almost always true that the researcher has a good understanding of the traffic and protocols being studied in the experiment, it is virtually never the case in actual networks that there is no other traffic in the network. The amount and mix of this competing traffic can and does play a significant role in the measured results. For example, if the experiment is designed to observe and analyze the performance of a new TCP congestion algorithm, it is imperative that the competing traffic contain both a mix of *non–conforming* traffic (UDP without congestion feedback) and TCP traffic with more traditional congestion control algorithms. If the study only contains traffic with the new algorithm, it is unlikely to to accurately represent the behavior of the new algorithm in deployed networks.

   Unfortunately, existing network simulation tools (including ns-3) do not provide easy methods for the researcher to create such competing traffic. While most tools do provide models for a number of application behaviors (such as models for web browsing and peer–to–peer applications), the researcher must individually construct application and protocol endpoints, create the connections between endpoints, and specify the mix of TCP and UDP traffic manually. For the work proposed here, we will extend the ns-3 framework to allow simple command line specification of the number (or fraction) of competing applications, the mix of competing application types, and the number of responsive and non–responsive flows. The assignment of the application endpoints, the distribution of the endpoints among the leaf network nodes, and the average data demand for each will be chosen from distributions, which will lead to more realism and make it easier for the researchers to run repeated experiments with variation in the traffic mixes.

   To achieve this goal, we will leverage existing work by Weigle et al. [WAHC+06], that creates simulation traffic models based on measured packet traces. We will extend the *TMix* approach to include both conforming and non—conforming traffic mixes.

4. **Mobility Models** All too often in simulation–based wireless research an over simplified and potentially flawed model for node mobility is used. It has been shown[YLN03] that the very widely used *Random Waypoint* mobility model degrades over time and that the average node velocity asymptotically approaches zero. Further, studies that are designed to compare two or more wireless network approaches should be performed under a number of different mobility models. The ns-3 tool presently has the ability to specify mobility models and provides a number of models to choose from. We will extend this capability to allow for command line argument specification selecting the desired mobility models, the random distributions desired for the models, and allow for differing subsets of nodes being assigned different models and different distributions. Kurtkowski et al. have also pointed out [KCN06] that the particular mobility model selected is not as important as are some derived statistics from the scenario on network partitioning and average hop count (and we would also suggest node density and rate of change of topology). Our framework will help users specify ranges and collect data on these derived metrics of mobile networking scenarios.

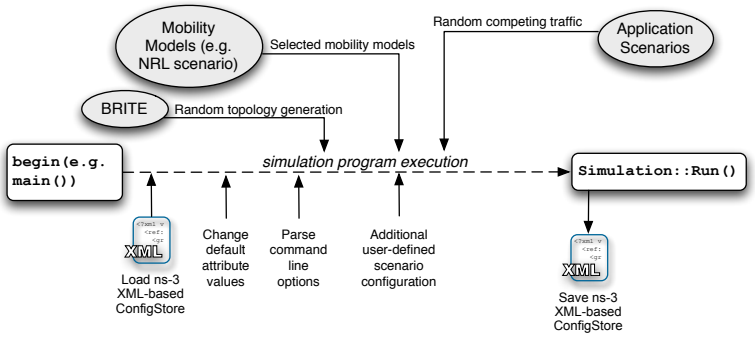


Figure 7: ns-3 workflow showing opportunities for modifying and saving scenario data.

Figure 7 illustrates how scenario generation plays a role in the overall ns-3 workflow. While we propose to embed BRITE within ns-3, because it will be easier to parameterize and configure from our other automation frameworks, we will also allow for more traditional integration of external scenario generators, whereby the generator is run prior to the simulation and ns-3 parses the result. ns-3 already has an XML-based configuration management system (the ConfigStore) that can be used at the beginning of a simulation to load an XML file, or just prior to running the simulation to save the exact configuration of the simulator prior to runtime; our scenario generation module will extend and complement the ConfigStore.

2.3 Wireless Validation and Scenarios

Summary of Deliverables: Integrate with the scenario definition and automation framework tasks to allow users to define and easily parameterize a set of wireless scenarios, including node density, loss models, mobility models, and channel models. Extend and document validation methodologies and at least one validated PHY channel model (802.11) including pathloss, short-term fading and/or lognormal shadowing, as well as multipath delay-spread and multi-user interference. Document how to use at least one wireless channel emulator (notably the CMU Wireless Network Emulator) and one commonly available hardware test-bed (Universal Software Radio Peripheral version 2 (USRP-2) planned as part of this effort) to validate ns-3 channel models. Contribute and document a validated physical layer and antenna model for modelling 802.11n (MIMO) with multiple antennas.

Despite notable progress in developing hardware test-beds (e.g. ORBIT at WINLAB, Rutgers University [OSSS05]) and emulation capabilities (e.g. CMU Wireless Network Emulator [JS08]) for network experimentation, wireless continues to provide the most compelling case for a virtuous cycle of improved network simulation tools, for the following reasons:

- Wireless test-beds are limited in capacity and usability at this point; with the future for their scalability uncertain;

- Wireless experiments are difficult to conduct and hard to reproduce;
- It may be impossible to actually experiment with future wireless devices (e.g. radios under development, no permission to use frequencies of interest).

Thus, despite the need and availability of multiple wireless simulation platforms, the trustworthiness of wireless network simulation has barely improved in recent years [CKC05]. A large contributor to this state of affairs it the lack of *validation* at key levels of abstraction in the network simulator, notably the link (PHY) and lower MAC layers. The reasons for this trace back (in significant measure) to the divergence of simulation architecture assumptions between the link and the network layers. While end-to-end PHY simulation is typically conducted at the granularity of streaming symbol sequences, network simulation uses packet-level abstractions, and further, is event-driven. As a result, the PHY abstractions inherent in the latter have been very coarse, often missing key features that distinguish the "wireless" nature of the channels. Notably, these include lack of proper abstractions for a) channel delay spread (multipath) and fading and b) interference due to contention in typical random access protocols.

Accordingly, validation of the abstractions in wireless network simulators is of paramount importance; we illustrate this with our work-to-date on IEEE 802.11b physical layer modeling and validation, often considered a canonical case [LH06]. As already stated, ns-3 abstracts the detailed symbol-by-symbol operation of the receive chain (demodulation and error-correcting coding, automatic gain control, equalization, etc.) and makes a reception decision based on the received signal-to-interference plus noise (SINR) at the decoder input relevant to the frame in question. By using a table that relates SINR to a symbol error ratio (SER), and considering the number of symbols in the frame, a decision can be made as to whether the frame is accepted or rejected.

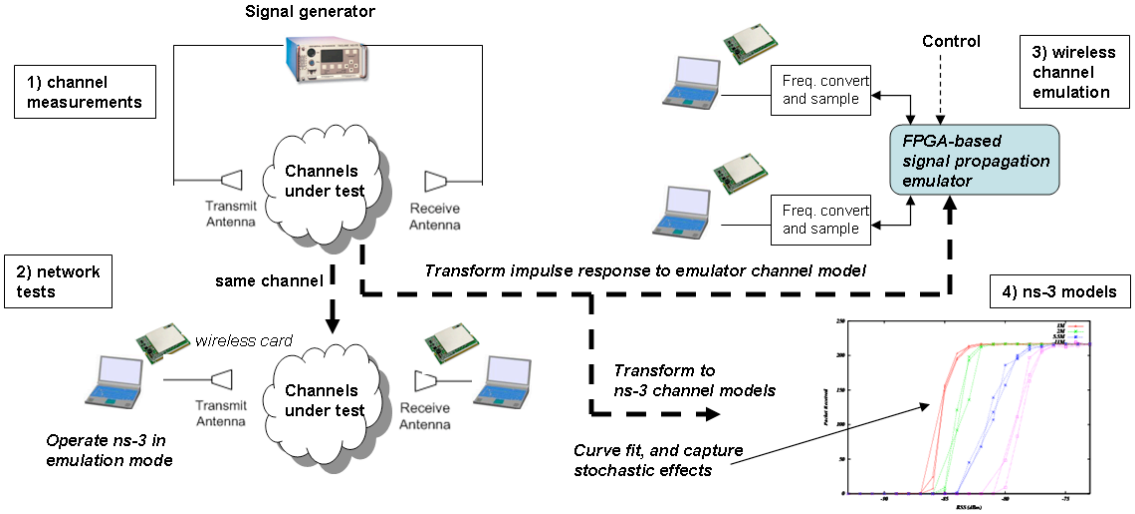


Figure 8: Example of a four-step validation process for ns-3 wireless models using CMU emulator

To validate a clear-channel scenario (no multipath or interference), results from a number of sources were collated - including analytical relationships for complementary code keying (CCK) modulation and coding schemes used by 802.11b, Matlab simulations, performance estimates in the IEEE 802.15.2 coexistence study group, and experimental data from the CMU wireless emulator published by [JS08]. These lead to several discrepancies; the results published in IEEE 802.15.2 study group diverged from experimental results, and there were no closed-form analytical results for the CCK modulation at higher data rates. Subsequently, newer performance analysis of higher rate CCK was able to match the experimental data (and confirm errors in 802.15.2 study group estimates). The right hand side of Figure 8 above illustrates the two main modes of clear-channel validation tests already performed, in which data collected on the CMU FPGA-based channel emulator was used to validate ns-3 WiFi models (the plots in the lower right hand corner, taken from ns-3/CMU data). Boeing is presently extending this framework to also include the ability to take channel measurements and build corresponding emulator channel models, allowing the overall workflow displayed in Figure 8 to be used. In summary, our experience is that such

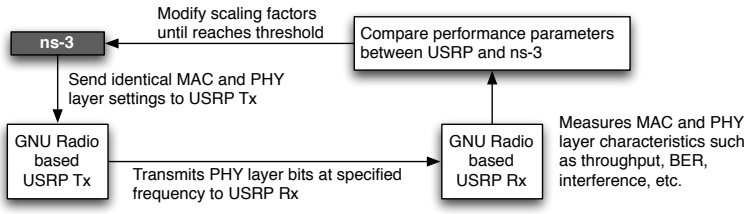


Figure 9: Proposed adaptation of our validation methodology to a USRP-based testbed

validation work is time consuming but exceedingly necessary, to improve the core PHY abstractions and build acceptance.

We propose to extend the above "mutually-reinforcing" validation methodology (Figure 8) to develop enhanced PHY/MAC abstractions for ns-3, specifically for scenarios inclusive of multipath and mutual interference. Building on our experience with the CMU wirleless emulator, we plan to follow the coupled simulation-experimentation approach to validation suggested in Figure 9. The simulator builds an initial PHY/MAC abstraction and conducts simulation to obtain network performance metrics of interest. The same MAC and PHY layer settings are coded into USRP to obtain experimental data for the same scenario and the results compared to ns-3 output. If the difference between the results exceeds a pre-determined threshold, the abstractions within ns-3 is updated. This iteration continues till the difference is below threshold.

The avenues for iterative PHY/MAC model improvements in ns-3 as suggested above will focus on better interference models (both self-interference due to multipath spread and external interference from other users due to random access). We plan to validate models to improve the scenarios of 1) a weak reference packet in the presence of a stronger interfering packet, and 2) a strong reference packet in the presence of a stronger interfering packet, as well as more complicated interference patterns as time allows. Presently, in wireless simulators including ns-3, once a packet is accepted for decoding (denoted as the 'reference' packet), any subsequent packet that arrives during this window is treated as interference and disregarded, and simulators often do not properly provide for aggregation of interference.

2.4 Virtualization

> **Summary of deliverables:** Build on the raw emulation capabilities provided in recent ns-3 releases to provide ease-of-use glue software, automation enhancements, performance benchmarking, and documentation for at least one instance of running ns-3 over a public testbed and one instance of running ns-3 in combination with virtual machines.

A commonly cited limitation of network simulation is that it provides scale at the expense of realism (e.g., [SC09]). This has led many network researchers in the past decade towards research-oriented testbeds, both in controlled environments (e.g., Emulab [Emu]) and in the Internet (PlanetLab [PR06] and PL-VINI [BFH+06]). Recently, the Trellis platform has been announced, blending the use of container-based virtualization techniques with in-kernel bridging improvements to offer packet forwarding approaching that of native kernel forwarding rates [BMM+08].

Despite these valuable improvements in the ability to interconnect large numbers of machines and virtual machines for network research experiments, we argue that simulation remains an invaulable tool for research, for the following reasons.

1. Large-scale testbeds, while vital to study large wired networks with fast links, typically do not try to model wireless networks at scale, if at all. Nevertheless, one of the most significant trends in networking is the expected continued growth in mobile wireless-based Internet devices.

2. Logistical issues in conducting testbed experiments remain challenging at large scale, due to the limited physical resources that must be shared and the difficulty to swap in and out experiments. We have first hand experience with the ORBIT testbed at Rutgers WINLAB and the CMU wireless emulator; we have ns-3 HOWTO pages devoted to running ns-3 on these testbeds. Each testbed has its own learning curve and its own set of issues. Experiments run at real-time speed and often there

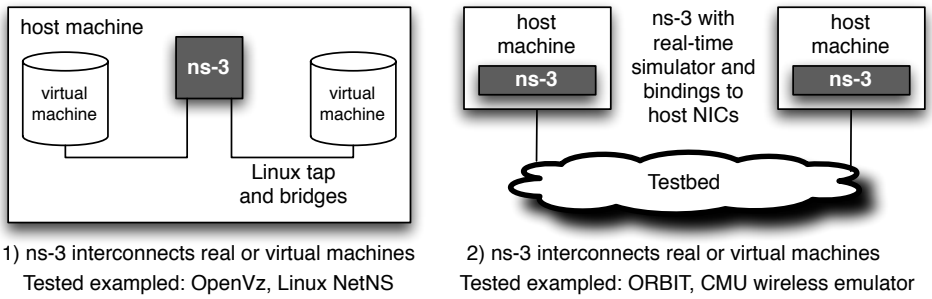


Figure 10: Two main modes of ns-3 emulation

are restrictions in the number of users supported or the length of any one experiment (it is possible to have a long-running experiment swap out just before it was about to complete). In the case of wireless testbeds, calibration of the equipment and basic validation of the configuration can be time consuming.

3. In "Isn't It Time You Had an Emulab" [LFG08], the authors argue that the ability to clone the Emulab testbed creates a low barrier to replicate Emulab for your own research. However, as the authors admit, setting up an Emulab is not a practical solution for many researchers, who do not have the support funding to maintain their own Emulab.

However, one cannot dispute the research value that these testbed facilities have brought to the networking research field. Therefore, our strategy with ns-3 has been to build the simulator with emulation in mind from the start, to allow researchers to more easily move between the simulation and experimental domains.

ns-3 has responded to this growth of testbeds and virtualizations by prioritizing the development of simulation modes and components that allow ns-3 to be used in conjunction with virtual or real machines. In the current project, we accomplished this design goal by designing our packet data structure to closely resemble network-ordered serialized data, and by aligning our simulation device APIs with those of Linux systems, as well as by adding a real-time scheduler. Figure 10 illustrates that there are two typical modes of operation for ns-3's real-time scheduler and integration with real network devices. The configuration on the left illustrates how ns-3 can form a high-fidelity networking subsystem to interconnect virtual machines; we describe one use case of this in more detail below. The right side of Figure 10 shows how ns-3 can be used as a packet generator for operation over testbeds. We have used this mode for ns-3 testing over the CMU wireless emulator and over ORBIT.

Figure 11 shows a functional diagram and GUI screenshot of the integration of ns-3 with another open source project: the Common Open Research Emulator (CORE) [ADHK08] based on the FreeBSD IMUNES project. CORE manages lightweight virtual machines (FreeBSD jails or Linux OpenVz containers) from a GUI and scripting API, and interconnects the virtual NICs from these machines with network emulation capability such as Netgraph or NetEm for Linux. Higher fidelity wireless emulations are possible by replacing NetEm with ns-3, through the use of ns-3's TapBridge device. This hybrid capability has been demonstrated in 2009 and is supported as of ns-3.5.

As of this writing, we have learned from our early experiences using ns-3 on testbeds and with CORE that the following topics need to be worked on further:

- **Configuration** of these environments is cumbersome and very hands-on, with the problems of consuming much user time and the ample opportunity to make an error. While some frameworks for managing experiments exist (e.g. the ORBIT Management Framework), we have found that these frameworks are still too much in their infancy to offer ns-3 users much help.

- **Timing** issues are problematic when working in hybrid environments. For instance, it may take some time for virtual machines to bring up their virtual interfaces, and if the simulation tries to bind too early to a non-existent interface, it may fail. To get around this, fragile ad hoc techniques (e.g. "sleep 2 seconds to let the emulator settle down") are typically used.

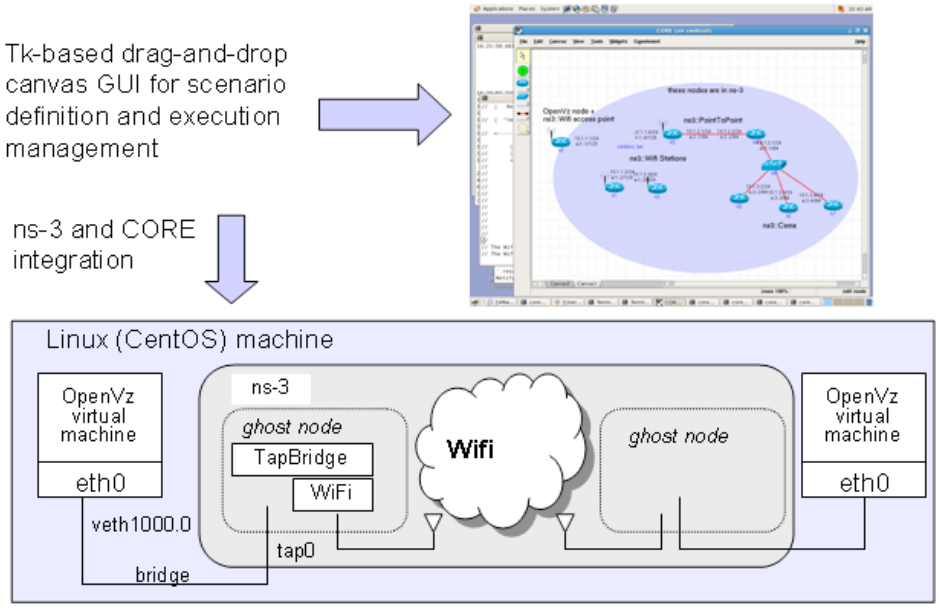


Figure 11: ns-3 and CORE virtual machine integration (enabled by the ns-3.5 release)

- **Performance** of the underlying networking system. We have observed degradation of performance when using built-in Linux bridging; this has led other projects such as Trellis [BMM+08] to find or develop other alternatives.

It is these ease-of-use, automation, and performance topics that we will work on in this additional framework topic, providing overall glue software and worked examples for integrating ns-3 with virtualization-based environments, in a manner such that others can build similar instances for their own preferred environment.

We also note that we have begun to collaborate on a related topic with INRIA, where the focus is on developing a framework for integrating ns-3 with federated testbeds [LFH+09]. We see the potential for synergy and collaboration, with the possibility that we might unify these approaches to allow for similar configuration and automation of all three (simulation, virtualization, and testbed) environments and various hybrids thereof.

2.5 Education

Summary of deliverables: Throughout the program, develop an on-line library of educational programs and laboratory experiments, and the supporting web-based framework to solicit additional contributions from the educational community.

In addition to the more traditional analysis and research use of the ns-3 simulation environment, we will encourage more widespread use of our tool in the classroom environment. Many of the networking models presently included in ns-3 (such as detailed models for TCP and routing) are major components of undergraduate and graduate classes in networking principles. The ability to create simple and easy–to–follow examples of given principles (for example the TCP slow–start algorithm) will allow classroom instructors to demonstrate how such algorithms work.

Although ns-2 previously had an educational component, our experience with using ns-2 as undergraduate courseware was that students spent a lot of time struggling with language unfamiliarity (Tcl) and the split-language model of development, and the high level of abstraction in ns-2's network and wireless modules did not map well to experience. In ns-3, we are now using Python scripting, and have a high degree of realism in our models, and we describe below two additional extensions we plan. First, we will enhance the ns-3 environment to enable a detailed graphical animation of the network being simulated. The animation will show each node, link, and packet in the scenario, and will allow stopping, backing up and fast forwarding of the time advancement. Each packet in the scenario will include more details about

this packet contents (such as TCP sequence number, acknowledgement number, flags, etc) and allow this information to be displayed with a mouse click or other stimulus. Several initial ns-3 animation projects, including one by George Riley (called *NetAnim*) will be leveraged in this effort.

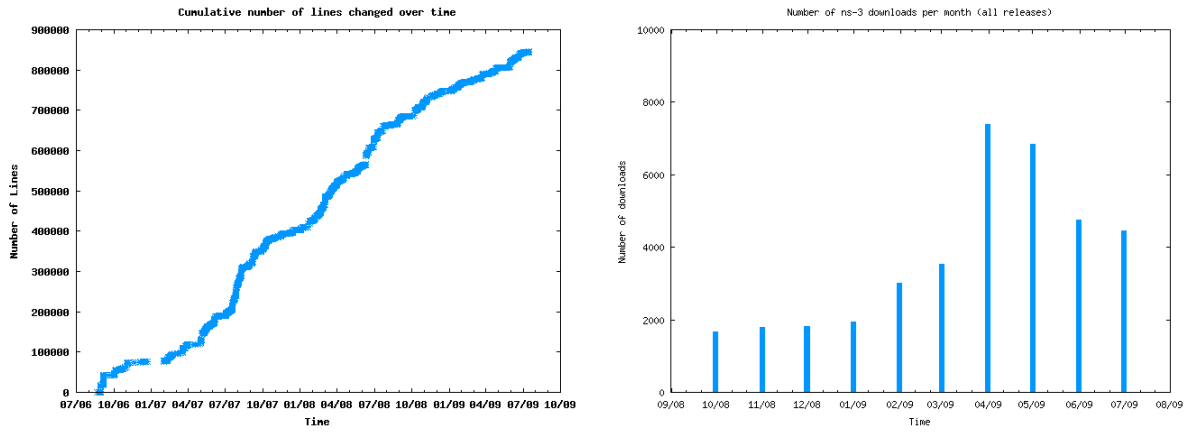
Second, critical protocol stack information found in the protocol endpoint models (such as the TCP congestion window value and the retransmit timeout estimate) will be made available in a similar fashion. Instructors will then use these animations in class, explaining how each of the various data items are set, modified, and used. The simulations will also allow modification of the various configuration parameters (such as the maximum queue length or the queuing discipline at a given bottleneck link), demonstrating the effect of these parameters on the overall performance of the network.

Clearly, the number of different protocols and networking methods that are taught in classrooms world–wide are too extensive for our team to construct such detailed aminated scenarios for each one. Therefore, as a second thrust of this effort, we will provide enabling tools for other faculty and researchers to allow creating and contributing of educational models to be used by others. To this end, we will maintain web pages and wiki discussion pages for the educational components, and easy ways for faculty and lectures to upload, download, and comment on the contributed models. Further, the animation interfaces to ns-3 will be easy to use for those faculty who are not experts in *C++* or Python.

These two thrusts of this work will lead to a large library of animations and instructional and laboratory scenarios that will assist both faculty and students in understanding complex network behaviour.

3 Evidence of Prior Contributions

As of this writing, ns-3 has been available and supported for research use for approximately one year (first stable release, ns-3.1, was made available on June 30, 2008). Since that date, the development team has made four additional releases (the most recent release is ns-3.5) and releases are made roughly quarterly. The following is offered as evidence of the contribution that ns-3 is making.



(a) Change history of ns-3 code base

(b) Downloads per month of ns-3 releases

Figure 12: Statistics on ns-3 development and usage

Figure 12 shows some statistics on the number of lines of code changed in our software repository, and the number of downloads logged by our web server per month in 2009. As mentioned above in Section 2, ns-3 has mailing lists that average roughly 200 messages per month, with over 900 subscribers to the developers list and 300 subscribers to the users list.

The following projects or investigators (outside of the proposing institutions) have announced that they are using ns-3 as part of their work. Each has contributed a letter of support in our supplementary documents section.

- **INRIA Planete**, funded by the French government and the European Commission, developing tools to allow for management and control of mixed simulation and testbed environments;

- **Naval Research Laboratory (NRL)**, Mobile Network Modeling project; this collaborative project among many universities, contractors, and U.S. Department of Defense laboratories is developing a suite of tools for mobile network emulation, and plans to include ns-3 within the framework; and
- **iTETRIS**, a Seventh Framework Programme funded by the European Commission to develop large scale wireless vehicular cooperative simulations.

The following institutions, outside of the NSF team, have already developed or are in the process of developing models for ns-3: **The Boeing Company** (802.11b physical layer model), **University of Florence, LART** (802.11n), **University of Karlsruhe** (802.11e, 802.11p), **Russian Academy of Sciences** (802.11s), **Louis Pasteur University** (IPv6), **INESC Porto** (Python bindings and binding generation library), **INRIA** (WiFi, Wimax, multicore simulator), and **Old Dominion University** (DelayBox, Tmix).

We have learned of several papers (without co-authors from the proposing institutions) that have been published or are accepted for publication in refereed workshops, conferences, and journals, that have used ns-3 for conducting simulations ([MLT08, KL08, MLL$^+$09b, MLL$^+$09a]).

We know of the following papers (without co-authors from the proposing institutions) that have been published or are accepted for publication in refereed workshops, conferences, and journals, that discuss ns-3 by describing models developed for ns-3 ([MVM08a, FT09]). One paper has compared ns-3 performance with that of other available simulators [WvLW09]. The iTETRIS project (referenced above) also compared ns-3 with OMNeT++ and ns-2 as part of its simulator selection process.

3.1 Additional Contributions

We cannot overstate the contributions that INRIA's Planete research group, through Walid Dabbous (Planete team lead) and Mathieu Lacage (research software developer), has made to ns-3. Mathieu Lacage has been supported for the duration of the ns-3 project and has traveled to the University of Washington for a half-year visit in 2008. Mathieu has written and maintained much of the new software core for ns-3, and contributed the Wifi NetDevice model. Walid and Mathieu have also arranged for and supervised several other students, interns, or staff working on ns-3. This work has been funded by the French government. We intend to continue this fruitful collaboration in the proposed project.

ns-3's existing funding from NSF did not fully cover the salaries of the two full-time developers hired on the project, but the University of Washington and Georgia Institute of Technology both agreed to supplement the balance (roughly 1/2 to 1/3) of each developer's salary.

In addition, ns-3 applied for and was selected to participate in the Google Summer of Code program in 2008 and 2009. This program funded three students in each year to work on ns-3 for the summer. Google pays a small stipend to students and mentors to work in this manner. We plan to continue to apply to this program annually.

4 Relation to Previously NSF-Funded Infrastructure

ns-3 has been previously funded through NSF CNS-0551686 (University of Washington), CNS-0551378 (Georgia Institute of Technology), and CNS-0924385 (University of Washington). The latter award was transferred to University of Washington in 2009 from a previous award made to ICSI Center for Internet Research, due to Dr. Sally Floyd's retirement. The proposed new infrastructure will be completely integrated with the existing ns-3 infrastructure. All funds remaining from earlier NSF grants will be expended prior to the beginning of this program.

5 Evaluation and Outreach

As a free, open source software project, we have the highest commitment towards disseminating our project results. All of our project software is openly available and easily acccessible, we provide tutorial and documentation support for users, and we try to respond to all user questions posted on our public mailing lists. We do this through the combined use of web sites, open mailing lists, open bug tracker, project wiki, and Internet Relay Chat (IRC) room.

We have conducted the following outreach activities as part of our current project:

- Tom Henderson held an ns-3 tutorial at the inaugural SIMUTools conference, in Marseilles France in March 2008, and followed this up with the inaugural Workshop on ns-3 at SIMUTools 2009 conference in Rome Italy in March 2009; he is now organizing the second Workshop on ns-3 at SIMUTools 2010 conference in Malaga Spain in March 2010;
- George Riley conducted an ns-3 tutorial at the Workshop on ns-2 (WNS2) held in Athens, Greece in October, 2008, and at the ACM SpringSim conference in March 2008;
- Several developers conducted a software demonstration at the ACM Sigcomm conference in Seattle WA in August, 2008;

User satisfaction is measured in a number of ways. First, we conduct the project in an open manner, giving people ample opportunity to provide feedback. Users have opportunities to express their interests in priorities for ns-3 development. Our mailing lists are very active; the developers and users lists both have hundreds of posts each month. All of our "usage" statistics (downloads, message traffic) are growing over time. We also have provided open invitations to attend our developers meetings (the last of which was held at the University of Washington in 2008).

For this project, we plan to continue all of the previously mentioned outreach activities, but we also plan to explore the development of online (web-based, video) tutorials and training, and look for more networking research venues to hold associated tutorials and workshops.

6 Project Team Qualifications

The team assembled to lead this project is especially well-qualified, with a diverse, complementary background. The management structure for the project is shown in Figure 13. Structurally, the overall responsibility for project execution will rest with the University of Washington, with the other institutions participating as collaborators under contract with NSF.

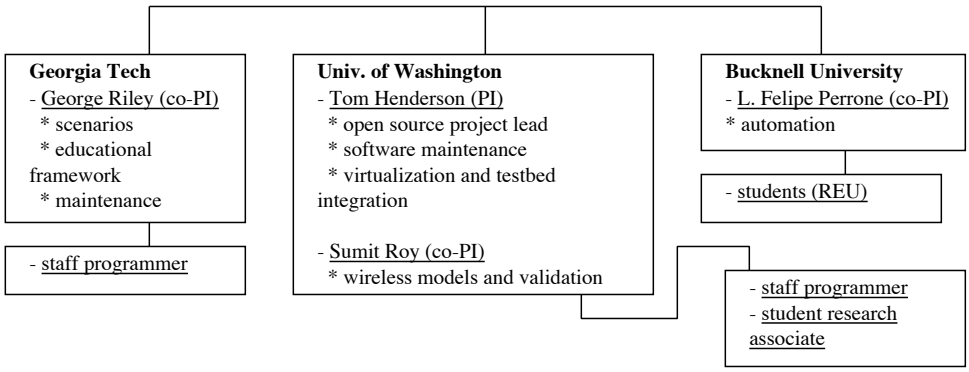


Figure 13: Management structure

The Principal Investigator on this project will be Dr. Thomas R. Henderson (Affiliate Professor, University of Washington). He has been a user and developer of ns-2 since 1997, and lead of the ns-3 project since 2006. He also continues to lead maintenance of ns-2, where he contributed several components over ten years ago. He has also developed three ns-2 educational modules/assignments for undergraduate education, which he used while teaching Introduction to Computer Communication Networks at the University of Washington. He has extensive experience on several discrete event network simulation platforms since 1992, including Simscript, QualNet, and *GTNetS*, and his research at Boeing involves Linux and BSD network and kernel programming. He also has significant research project management experience, including serving as PI or co-PI and technical lead for several Office of Naval Research (ONR) research contracts. He will be responsible for overall project management, and will focus technically on the *software maintenance* and *virtualization* tasks described above. One staff programmer will report to Dr. Henderson

Also at the University of Washington as co-PI will be Dr. Sumit Roy (Professor), focusing on *wireless models*. One graduate student will be supervised by Dr. Roy on this effort. His long record of contributions to wireless systems performance evaluation, and in particular, espousal of cross-layer design approaches, is the basis for his inclusion in the project team. Of particular note is the fact that his current research spans layers 1-3 having evolved from a strong and diverse base of layer-1 contributions in the past. His recent work in joint PHY/MAC optimization of 802.11 networks as well as link-aware routing uniquely positions him to contribute to the ns-3 research goals. Further, the presence of a 802.11 testbed at the University of Washington will allow integration of measurement-based link and MAC layer models into ns-3, an area that has been largely neglected to-date. This is of critical importance as wireless networks will operate in only partially known environments and the ability to adapt design parameters in such unstructured environments (as in software defined or smart/cognitive radios) will be fundamental to their success.

Dr. George F. Riley (co-PI, Associate Professor, Georgia Institute of Technology) will focus on *scenario generation* and *educational programs*. One staff programmer will report to Dr. Riley. Dr. Riley is a member of a network simulation team at Georgia Tech that has developed some of the largest and fastest network simulations reported to date. He has over 25 years of simulation experience, and has contributed *Nix-Vector* routing to ns-2 [RAZ01], developed the PDNS prototype (a federated, distributed version of ns-2) [RFA00], and the *Georgia Tech Network Simulator*, another C++-based packet-level simulation environment [Ril03]. He has also developed a distributed parallel execution version of ns-3 and has been co-PI of the NSF ns-3 project since 2006.

Dr. L. Felipe Perrone (co-PI, Assistant Professor, Bucknell University) will be responsible for *automation* components, with the support of undergraduate students. Since he joined Bucknell in 2003, he has been successful at creating opportunities for undergraduate students within his research activities, which resulted in several honors theses, summer research opportunities, independent studies courses, and co-authorship of papers published in international conferences. Dr. Perrone has extensive experience with simulation and has been working in the field for the last 20 years. His work includes the development of both optimistic and conservative parallel simulators, and simulation studies with cellular and WiFi networks. He is a leading developer of the Simulator for Wireless Ad Hoc Networks (SWAN) and of SWAN Tools, an automation framework for network simulation which will be leveraged in this effort.

7 Management Plan

7.1 Project Organization

The offerors propose a distributed project management team that integrates the expertise and research interests of co-Principal Investigators (PIs) from different institutions. This is the same arrangement that we have used on ns-3 previously. All of the PI responsibilities (described above in Section 6, with responsibilities shown in Figure 13)) draw on the special qualifications that each individual has in this area; combined, the PIs have over sixty years experience with developing software for discrete-event network simulators. The bulk of our requested budget goes directly towards funding the programming labor necessary to develop the infrastructure, under the guidance of the PIs.

The project will be managed as follows:

- **Open Source.** ns-3 is an open source project and it is important that open source software development practices be followed. All contributors will follow ns-3 processes for submitting and merging software into the main tree of ns-3. The PI on this proposal (Tom Henderson) is also the open source lead of ns-3.
- **Communication.** We will make use of regular IRC chats and typical collaborative tools and telecons for meetings as necessary. Where necessary, we have scheduled face-to-face development meetings.
- **Support.** Work to support the user community (primarily responding to bugs and queries on the mailing list) will be delegated and rotated among the developers and open source maintainers.

As this is a software project for use on general purpose PCs and clusters, hardware requirements budgeted for this program are minimal, mainly consisting of a few modern dedicated project servers at each site, and some wireless nodes for a testbed at the University of Washington.

7.2 Dissemination Plan

As a collaborative, open-source software project, the main payoff is not centered on the proposing institutions but instead on the broader impact that it offers the networking research and educational community at large. Specifically, we will continue the following:

- **Website and mailing lists.** We will continue to maintain the website for the project, hosting project documentation and Wiki, and mailing lists for users and developers, in addition to the new educational repository;
- **Code and regression servers.** We will continue to maintain the software repository and regression testbed for the project; and
- **Workshops and Tutorials.** We plan to continue existing workshops (Workshop on ns-3, held in conjunction with SIMUTools) and tutorials and demos (e.g. ACM Sigcomm demonstration in 2008).

7.3 Licensing

All software developed on this program will be delivered with an open-source software license compatible with the GNU General Purpose License (GPL) version 2.

Statement of Work

This Statement of Work defines a four-year infrastructure development effort for the ns-3 project, involving the University of Washington, Georgia Institute of Technology, and Bucknell University. Standard NSF reporting requirements apply.

Schedule of Deliverables

The main deliverable of the project will be new software and software maintenance contributed to the open-source ns-3 simulator project, as a result of each of the tasks defined below. The NSF project will participate in the open-source project and ensure that new ns-3 releases are released according to a regular release schedule, and the project will maintain the infrastructure (web site, mailing list, development tools, documentation, etc.) necessary to successfully develop and disseminate the software. The project will define yearly milestones for each task resulting in merged software by each annual milestone, or more frequently as needed. In the below, we have identified scheduled deliveries for most of our main tasks. (Note that the schedule of tasks for the Automation Framework has been defined to match the qualifications of the students who will be involved in the program's first two years.)

Tasks

1. **Automation Framework.**
    - **User interfaces.** By the end of year 1, create user interfaces for users with different levels of experience (student mode and expert mode). The interfaces will allow users to compose models and check their completeness and consistency, to obtain processed simulation output in different formats (annotated data or plots, for instance), and to control the execution of a simulation experiment. Refine this capability throughout the remaining years of the program.
    - **Experiment management.** By the end of year 1, define a language for the description of experiments. Develop code to generate individual points in the design of experiments space and translate them into a simulation script. Develop strategies to record information about the experiment set up in a database. Refine this capability throughout the remaining years of the program.
    - **Simulation control.** By the end of year 2, develop a framework to enable simulation experiments to take advantage of distributed and multicore computers, and to collect the results from each individual simulation, preprocess them, and store in a central database. Integrate into

this framework the ability to detect the end of transients in collected metrics and to determine simulation run length automatically. Refine this capability throughout the remaining years of the program.

- **Output processing.** By the end of year 2, annotate output of each simulation run to enable interoperability with external data processing and visualization tools. Refine this capability throughout the remaining years of the program.
- **Verifying completeness and consistency of models.** By the end of the program, develop framework to allow user to build a simulation model by piecing together components chosen from a library of sub-models. Develop mechanisms to ensure that the composed model is complete and that it respects relationships of dependence and compatibility.
- **Generation of simulator specific scripts.** By the end of the program, develop code to transform model subcomponents written in an XML-based description language to ns-3 compatible scripts.

2. **Scenario Development.**

   - **Artificially Created Topologies.** By the end of year 1, deliver a set of stock topology models for small scenarios, including the ability to parameterize them and construct them hierarchically.
   - **Random Topology Generation.** By the end of year 2, deliver a synthetic topology generator based on the work integrating BRITE into *GTNetS*.
   - **Empirical Topology Generation.** By the end of year 3, deliver software to construct AS topology graphs based on annotated network graphs.
   - **Mobility and Physical Layer Scenarios.** By the end of year 4, integrate mobility and physical layer scenario generation software.

3. **Wireless Validation and Scenarios.**

   - **Wireless scenarios** By the end of year 1, integrate with the other scenario definition and automation framework tasks to allow users to define and easily parameterize a set of wireless scenarios, including node density, loss models, mobility models, and channel models.
   - **PHY Models I** By the end of year 2, contribute a documented methodology and at least one validated PHY channel model (802.11) inclusive of a) pathloss, short-term fading and/or log-normal shadowing as well as b) multipath delay-spread and multi-user interference. Document how to use at least one wireless channel emulator (notably the CMU Wireless Network Emulator) and one commonly available hardware test-bed (potentially, USRP-2) to validate ns-3 channel models.
   - **PHY Models II: MIMO and directional antennas** By the end of year 3, contribute and document a validated physical layer and antenna model for modelling 802.11n (MIMO) with multiple antennas.

4. **Virtualization.** Build on the raw emulation capabilities provided in recent ns-3 releases to provide ease-of-use glue software, automation enhancements, performance benchmarking, and documentation for at least one instance of running ns-3 over a public testbed and one instance of running ns-3 in combination with virtual machines.

5. **Education.** Throughout the program, develop an on-line library of educational programs and laboratory experiments, and the supporting web-based framework to solicit additional contributions from the educational community.

6. **Maintenance.** Throughout the duration of the program, maintain simulation components, documentation, tutorials, and models that assist in the design, composition, execution, and analysis of network simulations:

References cited

[ABB+04] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 121–132. ACM Press, 2004.

[ADHK08] Jeff Ahrenholz, Claudiu Danilov, Thomas R. Henderson, and Jae H. Kim. Core: A real-time network emulator. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, 2008.

[ASV09] Matteo Maria Andreozzi, Giovanni Stea, and Carlo Vallati. A framework for large-scale simulations and output result analys with ns-2. In *Proceedings of the 2nd International Workshop on the Evaluation of Quality of Service through Simulation in the Future Internet (QoSim 2009)*, Rome, Italy, March 2009. ACM.

[BEF+00] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Hemy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo You. Advances in network simulation. *Computer*, 33(5):59–67, May 2000.

[BFH+06] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In VINI veritas: realistic and controlled network experimentation. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06)*, pages 3–14, New York, NY, USA, 2006. ACM.

[BMM+08] Sapan Bhatia, Murtaza Motiwala, Wolfgang Muhlbauer, Yogesh Mundada, Vytautas Valancius, Andy Bavier, Nick Feamster, Larry Peterson, and Jennifer Rexford. Trellis: a platform for building flexible, fast virtual networks on commodity hardware. In *CONEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–6, New York, NY, USA, 2008. ACM.

[CEV03] R. Canonico, D. Emma, and G. Ventre. An XML description language for web-based network simulation. In *Proceedings of the IEEE Symposium on Distributed Simulation and Real-Time Applications (DS-RT'03)*, pages 76–81, October 2003.

[CKC05] Tracy Camp, Stuart Kurkowski, and Michael Colagrosso. MANET simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, 2005.

[CON02] J. Cowie, A. Ogielski, and D. Nicol. The SSFNet network simulator. Software on-line: `http://www.ssfnet.org/homePage.html`, 2002. Renesys Corporation.

[DKVR09] X. Dimitropoulos, D. Krioukov, A. Vahdat, and G. Riley. Graph annotations in modeling complex network topologies. *ACM Transactions on Modeling and Computer Simulation (to appear)*, 2009.

[Emu] Network Emulation Testbed. `http://www.emulab.net`.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the ACM SIGCOMM*, 1999.

[FT09] J. Farooq and T. Turletti. An ieee 802.16 wimax module for the ns-3 simulator. In *2nd International Conference on Simulation Tools and Techniques (SIMUTools'09)*, March 2009.

[GPPP02] Timothy G. Griffin, Srdjan Petrovic, Anna Poplawski, and B. J. Premore. SOS: Scripts for Organizing 'Speriments, 2002. `http://www.ssfnet.org/sos/README`.

[GT00] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for internet map discovery. In *Proceedings of IEEE Infocom 2000*, 2000.

[JS08] Glenn Judd and Peter Steenkiste. Characterizing 802.11 wireless link behavior. *Wireless Networks*, June 2008.

[kcMM99] k claffy, T. E. Monk, and D. McRobb. Internet tomography. *Nature*, January 1999. `http://www.caida.org/tools/measurement/skitter/`.

[KCN06] Stuart Kurkowski, Tracy Camp, and William Navidi. Two standards for rigorous manet routing protocol evaluation. *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, 0:256–266, 2006.

[KL08] Joseph B. Kopena and Boon Thau Loo. OntoNet: Scalable knowledge based networking. In *4th International Workshop on Networking Meets Databases*, April 2008.

[KNG+04] David Kotz, Calvin Newport, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM '04)*, pages 78–82. ACM Press, 2004.

[LFG08] W. David Laverell, Zongming Fei, and James N. Griffioen. Isn't it time you had an emulab? In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*, pages 246–250, New York, NY, USA, 2008. ACM.

[LFH+09] M. Lacage, M. Ferrari, M. Hansen, T. Turletti, and W. Dabbous. Nepi: Using independent simulators, emulators, and testbeds for easy experimentation, June 2009. INRIA Technical Report RR-6967.

[LH06] Mathieu Lacage and Thomas R. Henderson. Yet another network simulator. In *Proceedings from the 2006 workshop on ns-2: the IP network simulator (WNS2 '06)*, Pisa, Italy, October 2006. ACM.

[LPN+01] Jason Liu, L. Felipe Perrone, David M. Nicol, Chip Elliott, and David Pearson. Simulation modeling of large-scale ad-hoc sensor networks. In *Proceedings of the European Simulation Interoperability Workshop 2001 (EURO SIW 2001)*, University of Westminter, London, UK, June 2001.

[Lum07] Lumeta, Inc. Research internet mapping home page. Software on-line: `http://www.lumeta.com/internetmapping`, 2007. Lumeta, Inc.

[Mey] David Meyer. Oregon routeviews database. `http://www.antc.uoregon.edu/route-views/`. University of Oregon Advanced Network Technology Center.

[MLL+09a] Shivkumar C. Muthukumar, Xiaozhou Li, Changbin Liu, Joseph B. Kopena, Mihai Oprea, Richardo Correa, Boon Thau Loo, and Prithwish Basu. RapidMesh: declarative toolkit for rapid experimentation of wireless mesh networks. In *4th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH 2009), in conjunction with ACM MobiCom*, September 2009.

[MLL+09b] Shivkumar C. Muthukumar, Xiaozhou Li, Changbin Liu, Joseph B. Kopena, Mihai Oprea, and Boon Thau Loo. Declarative toolkit for rapid network protocol simulation and experimentation. In *ACM SIGCOMM Conference on Data Communications (demo)*, August 2009.

[MLMB01] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An approach to universal topology generation. In *Proceedings of Second International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01)*, 2001.

[MLT08] Federico Maguolo, Mathieu Lacage, and Thierry Turletti. Efficient collision detection for auto rate fallback algorithm. In *Third Workshop on multiMedia Applications over Wireless Networks (MediaWiN 2008)*, July 2008.

[MVM08a] Julian Montavont, Sebastien Vincent, and Nicolas Montavont. Implementation of an ipv6 stack for ns-3. In *2nd International Workshop on NS-2 (WNS2 2008)*, October 2008.

[MVM08b] Julien Montavont, Sebastien Vincent, and Nicolas Montavont. Implementation of an IPv6 stack for ns-3. In *Proceedings of the Second International Workshop on ns-2 (WNS2)*, October 2008.

[OSSS05] M. Ott, I. Seskar, R. Siraccusa, and M. Singh. ORBIT testbed software architecture: supporting experiments as a service. In *Proceedings of the First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom 2005)*, pages 136–145, February 2005.

[Paw90] Krzysztof Pawlikowski. Steady-state simulation of queueing processes: a survey of problems and solutions. *ACM Computing Surveys*, 22(2):123–170, 1990.

[Paw03] Krzysztof Pawlikowski. Towards credible and fast quantitative stochastic simulation. In *Proceedings of the International Conference on Design, Analysis and Simulation of Distributed Systems (DASD '04)*, Orlando, FL, USA, March 2003.

[PCSW09] L. Felipe Perrone, Claudio Cicconetti, Giovanni Stea, and Bryan C. Ward. On the automation of computer network simulators. In *Proceedings of the* 2^{nd} *International Conference on Simulation Tools and Techniques (SIMUTools 2009)*, Rome, Italy, March 2009.

[PJL02] Krzysztof Pawlikowski, Hae-Duck Joshua Jeong, and Jong-Suk Ruth Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, 40, January 2002.

[PKW08] L. Felipe Perrone, Christopher J. Kenna, and Bryan C. Ward. Enhancing the credibility of wireless network simulations with experiment automation. In *Proceedings of the IEEE International Workshop on Selected Topics in Mobile and Wireless Computing (STWiMob 2008)*, pages 631–637, Avignon, France, October 2008.

[PR06] Larry Peterson and Timothy Roscoe. The design principles of PlanetLab. *SIGOPS Oper. Syst. Rev.*, 40(1):11–16, 2006.

[RAZ01] George F. Riley, Mostafa H. Ammar, and Ellen W. Zegura. Efficient routing using nix-vectors. In *2001 IEEE Workshop on High Performance Switching and Routing*, May 2001.

[RFA00] George F. Riley, Richard M. Fujimoto, and Mostafa H. Ammar. Parallel/Distributed ns. Software on-line: `http://www.cc.gatech.edu/computing/compass/pdns/index.html`, 2000. Georgia Institute of Technology.

[Ril03] George F. Riley. The Georgia Tech Network Simulator. In *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools '03)*, pages 5–12, New York, NY, USA, 2003. ACM Press.

[roo09] ROOT: A data analysis framework. Software online: `http://root.cern.ch`, 2009.

[RR07] Dheeraj Reddy and George Riley. Measurement–based physical layer modeling for wireless network simulations. In *Proceedings of Fifteenth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'07)*, Oct 2007.

[SC09] Network Science and Engineering Council. A report of the network science and engineering council. `http://www.cra.org/ccc/docs/NetSE-Research-Agenda.pdf`, July 2009.

[SMW02] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM*, pages 133–145, August 2002.

[SPP06] D. Savić, M. Pustisek, and F. Potortì. A tool for packaging and exchanging simulation results. In *Proceedings of the First International Conference on Performance Evaluation Methodologies and Tools (Valuetools '06)*, Pisa, Italy, October 2006.

[WAHC+06] M. Weigle, P. Adurthi, F. Hernandez-Campos, K. Jeffay, and F. Smith. Tmix: A tool for generating realistic TCP application workloads in ns-2. *ACM Computer Communications Review*, July 2006.

[WSHW08] Elias Weingärtner, Florian Schmidt, Tobias Heer, and Klaus Wehrle. Synchronized network emulation: matching prototypes with complex simulations. *SIGMETRICS Perform. Eval. Rev.*, 36(2):58–63, 2008.

[WvLW09] Elias Weingartner, Hendrik vom Lehn, and Klaus Wehrle. A performance comparison of recent network simulators. In *(submitted to) Proceedings of the IEEE International Conference on Communications 2009 (ICC 2009)*, Dresden, Germany, 2009. IEEE.

[YLN03] Jungkeun Yoon, Mingyan Liu, and Brian Noble. Random waypoint considered harmful. In *Proceedings of IEEE INFOCOM 2003*, October 2003.

[ZCB96] Ellen W. Zegura, Ken. Calvert, and Samrat. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE INFOCOM 96*, 1996.