# Developing the Next-Generation Open-Source Network Simulator (ns-3)

CRI Proposal #0551686
Project Year: 2006

Thomas R. Henderson and Sumit Roy (University of Washington)
Sally Floyd (ICSI Center for Internet Research)
George F. Riley (Georgia Institute of Technology)
thenders, roy@ee.washington.edu, floyd@icir.org, riley@ece.gatech.edu

## 1. Abstract

We are organizing a software development program to comprehensively re-design, enhance and maintain the popular Network Simulator (*ns*), to address research and educational challenges for the next generation of data networks. In our four-year program, we will i) refactor the simulator's architecture, ii) develop new networking protocol models, iii) provide new opportunities for software encapsulation, and iv) integrate the tool with virtual network testbeds. We will also introduce proven open-source practices that should enable *ns* development and software maintenance to become self-sustaining in the future, based on a large community of developers and power users. This project will advance the state-of-the-art in simulator design of the identified areas, along with supporting the mundane yet critical activities of code maintenance, documentation, integration, validation, and educational script generation. Since the resulting simulation code will be freely available to any individual or organization, *ns-3* will facilitate a significant increase in use for new simulation-oriented research as well as integration into courseware. Further, the project will emphasize a software development model that encourages and incorporates contributions from the user community.

## 2. Introduction/Background

*ns-2* is the second major iteration of a discrete-event network simulation platform programmed in *C++* and Object Tcl (OTcl). *ns-2* was first released in 1996, and derives from earlier work on the REAL simulator by Keshav and *ns-1* simulator by McCanne, Floyd, and Fall. *ns-2* is a major architectural change from *ns-1*; the simulator became entirely based on the blend of OTcl and *C++*.

The core of *ns-2* is written in *C++*, but the *C++* simulation objects are also linked to shadow objects in OTcl. Simulation scripts are written in the OTcl language (an extension of the Tcl scripting language). This structure permits simulations to be written and modified in an interpreted environment without having to resort to recompiling the simulator each time a structural change is made. In the timeframe that *ns-2* was introduced (mid-1990s), this provided both a significant convenience in avoiding many time-consuming recompilations, and also allowing potentially easier scripting syntax for describing simulations. *ns-2* has a companion animation object known as the Network Animator (*nam*), used for visualization of the simulation output and for (limited) graphical configuration of simulation scenarios.

Presently, *ns-2* consists of over 300,000 lines of source code, with probably a comparable amount of contributed code that is not integrated directly into the main distribution but is maintained elsewhere. Figure 1 illustrates the past funding history of *ns*.

> The **DARPA VINT** (Virtual InterNetwork Testbed) was a collaboration between USC/ISI, Xerox PARC, Lawrence Berkeley National Laboratory, and UC Berkeley. The project ran from 1997-2000, and was focused on developing the core simulation code for *ns-2* and *nam*. During this period, *ns-2* saw its most intense core development. During the course of building out *ns-2*, a number of research aims were also met, including the study of composable simulation frameworks, abstraction techniques and tools, visualization techniques, real-time network emulation, and network topology and traffic generators.

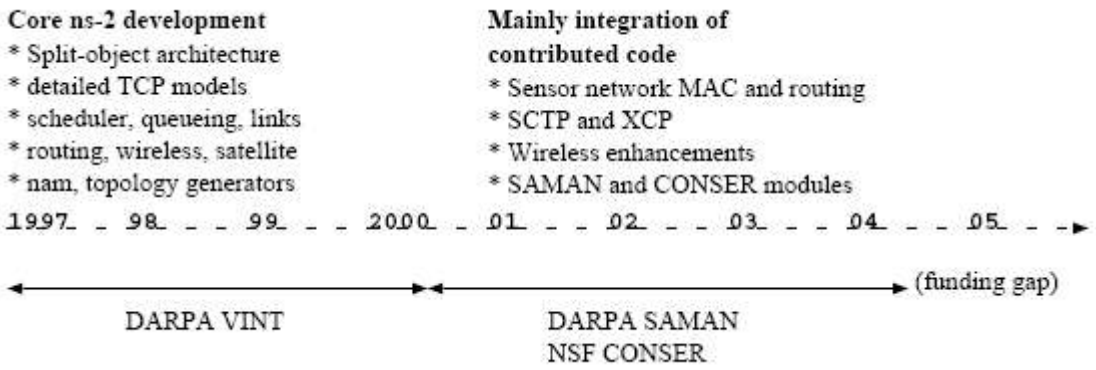


*Figure 1. Previous ns funding and results.*

From 2000-2004, *ns-2* development was no longer funded directly by a dedicated project, but instead used two programs **DARPA SAMAN** (Simulation Augmented by Measurement and Analysis for Networks) and **NSF CONSER** (Collaborative Simulation for Education and Research) to fund code maintenance activities. During this period, core development of *ns-2* and *nam* slowed, although a number of contributed modules were ported in annually.

Since 2004, no funding for *ns-2* has been in place. Development and extension of the simulator has slowed further still; although there is still a quite active user community, there is less support to integrate contributed code into the project, and little available manpower for addressing core architectural issues. In 2005, one of the PIs (Henderson) began to maintain the *ns-users* and *ns-developers* mailing lists, coordinated an effort to streamline the source code licensing (to conform all licenses to a free software license model), and initiated the move of the *ns-2* project to the Sourceforge project hosting web site.

## 3. Goals, Objectives, Targeted Activities

Despite *n*s-2's popularity, there is a critical need to do core refactoring, software maintenance, and extension of the simulator. Experience has shown that issues like software maintenance (e.g., tracking compiler and operating system evolution), documentation, educational development, and code integration are lesser priorities for individual contributors focusing their limited time primarily on producing research output for publications. In particular, we have identified the following key areas that require development of a new *ns-3* simulator:

**Software core:** The current *ns-2* core often limits the scalability (both speed and memory footprint) to tens or perhaps a hundred or so nodes. The software is not 64-bit clean, is not modular enough, does not make use of more recently established object-oriented design principles, and does not have native support for multi-processor or distributed, federated simulations. The emulation capability (ability to interoperate with live networks) is outdated and of limited utility.

**Internal composition:** The software architecture places restrictions on how simulation objects may be combined in new ways. Many pieces of the software do not correctly interoperate (for example, wireless simulations can not use the OTcl-based routing protocols). This is repeatedly a reported source of frustration for *ns* users.

**External software integration:** *ns-2* is currently missing out on opportunities to leverage the wide number of advanced, externally developed programs such as open-source routing daemons (quagga, XORP, OpenBGP), traffic generators (iperf, tcplib), network virtualization testbeds (IMUNES, PlanetLab), network analysis software (Ethereal, tcptrace), etc. These tools can be integrated by the development of better *ns-3* APIs and approaches to syntactically port application- and kernel-level code into the simulator's object-oriented, event-driven architecture.

**New models:** The protocol and traffic model library needs updating to cover more recent growth of IEEE 802-based wireless network and channel variants, IPv6 protocols, and newer popular applications and services (e.g., peer-to-peer, messaging, voice over IP, BitTorrent).

**Documentation:** The *ns-2* documentation has not been significantly updated for many years, is no longer consistent with the software core, and does not include more recently added modules.

**Visualization:** The *nam* animator is not well integrated with much of the code (wireless, Ethernet, satellite) and needs updating or replacement.

**Educational use:** More work is needed to build a comprehensive set of educational scripts, such as tracking the contents of popular undergraduate networking texts, providing protocol models that map better to actual implementations, and providing "newbie" support for simulator operation, visualization, and simple model development.

## 4. Infrastructure and Facilities

Our software will be developed as application software using widely and freely available software tools and general purpose desktop and server computers. The infrastructure needs of our project are lightweight; namely, a reliable and maintained software source code repository server, tinderboxes or virtual machines for development and regression testing, mailing lists, infrastructure (mirrors) for software dissemination, and a web site. The institutions involved in *ns-2* and *ns-3* will support these infrastructure needs.

## 5. Project Description

The *ns-3* development project is divided into multiple tasks. Below, we provide brief description and justification for three major tasks: i) core refactoring, ii) integration, and iii) maintenance.

### 5.1 Core refactoring

This task will design and refactor the core of the *ns-2* simulator. The output of this task will be a cleaner, more scaleable, and more extensible core. George Riley (architect of the Georgia Tech Network Simulator (*GTNetS*) and parallel, distributed extensions to *ns-2*) will lead this task. In this section, we outline a set of design objectives for the *ns-3* simulator. We see these as basic requirements for the design, implementation, and overall success of this tool.

**Reuse:** First, the *ns-3* design and implementation should leverage large parts of the code base of existing tools, such as *ns-2*, *GTNetS*, and others. These existing tools have hundreds of thousands of lines of code, and hundreds of existing network models. While we fully expect some modifications to these code bases will be necessary to fit within our overall design, we will strive to avoid complete re–design or re–implementation of existing modules as much as possible. To achieve this goal, we plan to design an automated tool to analyze and convert of much of the existing *ns-2* OTcl code to enable inclusion in our design. We expect to be able to reuse much of the existing *ns-2* and *GTNetS* code.

**Scalability:** One of the major concerns about *ns-2* cited by its users is scalability. *ns-2* is a sequential execution simulator with a single event processing loop running on a single processor. Although such a simulator can scale to hundreds of nodes when underlying communications models are heavily abstracted, the memory and processing resources of a single machine become a bottleneck when more sophisticated channel models (e.g., wireless) or higher-rate links (e.g., 10 Gbps) are included. Researchers have taken various approaches to improve *ns-2* scalability, including the caching of redundant computations and function calls (the "Staged NS (SNS)" project at Cornell), use of on-demand route computation, and partitioning the simulation into wireless clusters (the *ns-2 gridkeeper* and similar structures).

There are a number of factors contributing to the scalability limitation, but the fundamental bottleneck is the execution on a single processor. We believe that the *ns-3* simulator should be designed from the outset to support parallel and distributed simulation. We have already applied these concepts to *ns-2* to create a parallel, distributed version of *ns-2* (PDNS). In previously reported results, PDNS achieved a speedup of a factor of 80 and a simulation speed of 6 million packet hops per second, on 128 processors).

**Other Design Considerations:** Our *ns-3* design will also consider the following: better application of modern C++ design such as the use of C++ templates, polymorphism, and type-casting facilities; replacement of the OTcl scripting capability with a new scripting interface that avoids the split-object implementation paradigm that has proven to be problematic for *ns-2*; better match of simulation model design to the design of real networks and real network elements; memory-efficiency considerations; tracing and statistics logging facilities; emulation capabilities; network topology creation facilities; and visualization.

### 5.2 Integration

A key step forward of our proposed *ns-3* project will be the level of integration that we will obtain with the vast amount of free, open-source software and research projects available on the web. We have three specific goals in mind:

1) Extension of the simulation capability via integration with open-source software (e.g., Ethereal packet analysis, Click/XORP routing).

2) Abstraction layers and interfaces for porting implementation code into the *ns* environment; and

3) Interfaces to allow users to easily migrate between simulation and network emulation environments.

**Open source integration:** An opportunity that most every simulation environment has missed is the opportunity to leverage the extensive amount of free, open-source networking code within operating systems and applications. Typically, simulators reimplement protocols from scratch, leading to a costly software effort and divergence from actual implementation code. There are limited exceptions (the TCP code in QualNet, for example, is ported from BSD, and the NCTUns simulator uses a kernel-reentrant programming paradigm to use actual Linux stack code), but predominantly, protocols are reimplemented for the simulation environment. It is also difficult in general to write simulation software that can run both in simulation and implementation environments. The Naval Research Laboratory's *protolib* toolkit is an example of a publicly-available library that allows newly developed models to be written to a software abstraction library supported both in a simulation and implementation environment. This approach works well if a protocol implementation is written from scratch; however, it generally does not work so well when existing software, often written in a lower-level, non-object-oriented language such as *C*, is used.

In the *ns-3* project, we intend to focus on simulator design that facilitates the reuse of existing software and applications. Such an approach helps to meet our educational goals as well, since the simulation models mimic how the software is run in real implementations. Specifically, we see the following key opportunities for software integration and reuse: i) ported *application code* using sockets API, ii) *routing protocols* such as XORP, quagga, and OpenBGP, iii) *network stack code* such as the Network Simulation Cradle, and iv) *tools to parse output data* such as tools that work on pcap format traces such as tcpdump and Ethereal.

Finally, we recognize the significant research infrastructure advances of the past few years, with such projects as PlanetLab, Emulab, and WHYNET. These testbeds provide opportunities to explore protocol interactions in less controlled environments (such as cluster-based, remotely executed testbeds including Utah's Emulab) and to deploy long-running experimental services and overlays on the existing Internet (PlanetLab). Presently, Emulab uses *ns* scripting syntax to describe its experiments, and offers a version of the *ns* emulation environment to experimenters. Our project will coordinate with testbed projects to ensure that *ns* can successfully execute in those environments and is well documented enough for ease of use.

### 5.3 Maintenance

Some activities planned for this task include:

develop functional and technical specifications for the new software design,

establish a source code control system, coding styles, and software development model, including tinderboxes for daily builds on different platforms,

*ns-2* has a validation facility for ensuring that simulation output in one module is not changed by a code change in an unrelated module. Create new validation test suites for the new simulator and models. Port validation scripts that do not work in backward compatibility mode,

update the documentation and create tutorials,

establish a web presence for collaborative software development, wikis, mailing lists, and dissemination of the software, and

attend and participate in annual program review meetings, at sites to be determined.

Once *ns-3* is established, we intend to get a broader community involved in ongoing maintenance activities, by encouraging contributors and power users to take up portions of the maintenance activities, as in other open-source projects.

## 6. Indicators of Success and Major Accomplishments

Broadly, we intend for our simulator to become a preferred simulation tool for research and education on data networking. The current version of *ns* (*ns-2*) enjoys widespread use in the research community; the simulation code has been contributed by over one hundred individuals and organizations, there are several thousand logged downloads per month, and use of the simulator is consistently referenced in 10 to 25% of the papers in the top conferences in the field. We intend to maintain or improve these statistics with our next-generation simulator.

## 7. Publications/Patents

We have collectively authored one publication under submission:

T. Henderson, S. Roy, S. Floyd, and G. Riley, "ns-3 Project Goals," (*submitted to) Workshop on ns-2*, Pisa Italy, October 10, 2006.

## 8. Outreach Programs and Supported Students

Presently, since our project has not yet officially started, we do not have outreach programs in place. Our project intends to develop and discuss the simulation software in open forums on the web. We also intend to collaborate with other projects. We have one collaboration already in place with the Planete research group at INRIA Sophia-Antipolis, who is funding software development for the next-generation simulator.

We have budgeted for partial funding of one student at the University of Washington, pending the funding of our program.

## 9. Future and Strategic Directions

Our project intends to hold a software design review and kickoff meeting in the mid-August timeframe. Future directions will be discussed on the ns-developers mailing list.[1]

[1] http://mailman.isi.edu/mailman/listinfo/ns-developers