# Towards Scalable and Realistic Node Models for Network Simulators

Stein Kristiansen, Thomas Plagemann, Vera Goebel
Department of Informatics, University of Oslo, Gaustadalléen 23 D, N-0373, Oslo, Norway
{steikr, plageman, goebel}@ifi.uio.no

## ABSTRACT

Network simulators typically do not include node models. Our studies show that in networks such as mobile networks, the impact of nodes on performance can be significant. Existing techniques to simulate nodes' are not scalable for network simulations, and require a too large modelling effort to be feasible for network research. In this paper, we propose to capture flexible per-protocol performance profiles from real, running systems using instrumentation and traffic benchmarking techniques. By using the obtained profiles as input into an extended scheduler simulator, the behaviour of the node can be accurately reproduced. Since the processing overhead is represented statistically, we preserve scalability and a low modelling overhead.

## Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

## General Terms

Experimentation, Measurement, Performance

## 1. MOTIVATION AND PROBLEM

Network protocols are most commonly evaluated with network simulators. These simulators rely on accurate models of protocols and communication media, which are assumed to primarily impact the performance. The impact of nodes on the performance is generally ignored. We claim that it is important, especially in mobile networks, that node models in network simulators properly reflect the impact of protocol execution in the nodes onto the protocol performance. Our general observation is that the weaker the node is and the more workload it has to handle, the larger is its impact on the performance. Furthermore, the more nodes involved in protocol execution, the less accurate the results will get, i.e, in an end-to-end protocol over several hops in a Mobile Ad-Hoc Network (MANET). The severity of this problem grows with the time sensitivity of the involved protocols and applications. In network technologies such as sensor networks, MANETs, or opportunistic networks, this problem is particularly prominent as nodes might have to work as end-user devices and routers. We recently studied the forwarding capacity of the Nokia N900 smart phone in real-world experiments [2]. Only with the simple task of IP forwarding, the device was not capable of forwarding more than 6 Mbps over a 54 Mbps network. By analysing traces from a run with 2 Mbps, we found that most packets spent 10-40 ms within the device, depending on the bus frequency. The equivalent experiment in ns-3 yields an average end-to-end delay of 6.85 ms.

**Requirements:** To overcome the discrepancy between results from real world experiments and simulation experiments it is necessary to enable the network simulator to calculate the impact of protocol execution in the nodes. This requires an accurate model of the protocol execution in a particular device, which we call *node model*. In addition to accuracy, we require the node model to be sufficiently lightweight to allow network simulations with many nodes. Furthermore, it is important that the effort to create node models is not too high, otherwise it would be unrealistic to assume that models for the multitude of possible combinations of protocols and devices will be developed. Therefore, it should be possible to (1) re-use the large base of existing models present in popular, general-purpose network simulators such as ns-2/3, and (2) combine models of individual protocols to models of entire protocol stacks.

## 2. APPROACH

Our overall approach is based on four elements. First, we take on a modular approach where protocol stacks can be composed from individual protocol execution profiles. Second, for these profiles to be realistic, they are obtained from analysis of traces from real, running systems. To preserve scalability, the profiles contain execution duration distributions rather than implementation details. Third, to capture the dynamics of multi- threaded systems, they are used as input to a scheduler simulator. Fourth, to facilitate the re-use of existing protocol models, we provide the framework to integrate the scheduler simulator with a network simulator. The goal is to make the timing behaviour of protocol models in the network simulator reflect that of a real-world implementation. We focus primarily on small, handheld devices operating in MANETs, but the usefulness of our model extends to other scenarios where nodes can have a significant performance impact. Although we use ns-3 and Linux in our implementation, our approach can be implemented with other OSes and simulators as well.

The overall contribution is a solution to improve accuracy of network simulations where nodes have a considerable impact on performance. This contribution is twofold. First, we provide a tracing and analysis technique to capture the timing behaviour of protocols running on existing multi-threaded devices. Then, we provide the framework to map this behaviour onto existing protocol models in network simulators. As opposed to existing techniques, our model realistically captures the dynamics of multi-threaded systems, while preserving scalability and a low modelling overhead.


*SIGCOMM'11,* August 15–19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

## 3. KEY CHALLENGES

Due to the increasing heterogeneity of available hardware, OSes and protocol implementations, finding parameter values for models is an inherently difficult task. Establishing representative default values is in many cases impossible. Furthermore, a protocol's timing behaviour is determined by its implementation complexity rather than its specification. Since there are several ways to implement protocols, it is clear that it is not feasible to predict execution time from a protocol model alone. If we have the implementation available, along with an architectural description of a device, we can estimate its execution time with existing code analysis techniques. These techniques typically only estimate the number of CPU cycles consumed, and are therefore by themselves not sufficient to characterize overall timing behaviour of a node. Modern devices are usually equipped with several physical execution units in the form of CPUs, DMAs and processors on the NIC. How protocol processing is delegated among these, and the degree of parallelism achieved, has a large impact on performance. Different protocols can run concurrently within different time-shared Logical Execution Units (LEUs) on the same CPU. Their time of execution, and the length of their time-slice, depends on scheduling rules, timers and the amount of workload. This concurrency results in queuing delays between execution units and causes traffic burstiness and tail-dropping of packets. Since packets traverse several protocols, the processing for each packet can be spread across different LEUs. This results in an intermixing of processing stages of different packets, which we call *pre-emptive packet handling*. Our experience with the Nokia N900 demonstrates that the high number of LEUs involved in packet packet forwarding results in a significant amount of imposed traffic burstiness. The reason for the burstiness stems from the common approach to distribute the handling of individual packets among several LEUs. It is important that a node model is able to reproduce this effect.

## 4. STATE-OF-THE-ART

Although it is possible with existing techniques to model nodes, none of these fulfill all the above mentioned requirements. System simulators are generally too computationally expensive to be used for large scale, long-lasting network simulations. They impose an unreasonably large modelling effort onto a network researcher, because it is required to gain a deep insight into the OS and hardware, and because existing network simulators can not be re-used. To address the latter, a co-simulation framework has been proposed to combine OMNeT++ with SystemC [3]. Scalable models target either sensors or routers. Sensor simulators either do not provide any means to relate processing overhead to real implementations [4], or they generate their model directly from real implementations. The latter is unsuitable for general purpose network simulators. Works on modelling routers [1] do not consider pre-emptive packet handling, and focus on forwarding only.

## 5. DETAILED APPROACH

We extract per-protocol models from real-world traces. The traces are obtained from protocol-centric traffic benchmarking on a device running an instrumented Linux kernel. We instrument entry- and exit-points of protocol services to obtain service execution time distributions. To unveil the effect of concurrent workload, we perform the benchmark with varying CPU load and parallel memory usage. We also instrument the request queues between (logical) execution units to trace per-packet and -service workflows, and to account for queuing during resource contention. To quantify resource utilization and to unveil the distribution of services among execution units, we instrument the scheduler queues, context switches and subsystems for handling interrupts, timers and deferred work. To be as non-intrusive as possible, we use a combination of dynamic and static tracing techniques. Low-overhead, compiled-in static tracing is used for the scheduler, context switches and the different OS-specific subsystems. These events occur frequently and the location of the trace points do not change with the composition of the network stack. Dynamic tracing is used for services and request queues between protocols. This is to preserve the flexibility of tracing different protocols without the need to re-compile the kernel. To reduce the overhead of dynamic traces, we traverse a user-specified list of services and request queues, adjusting the configuration of dynamic probes to target individual services at a time. To quantify the overhead of processing on the NIC, we deduct intra-OS delay from the intra-node delay obtained by an external monitoring computer. By statistically analysing the resulting traces, we obtain Service Execution Profiles (SEPs) containing (1) per-packet and -service resource utilization distributions, (2) service-to-execution unit mappings and (3) blocking and non-blocking requests made between execution units. We are currently implementing the static tracing component using the *ftrace* tracing utility, and are analysing the traces to ensure that the approach yields acceptable accuracy. To aid our understanding, we have created tools to visualize these traces.

The only currently available alternative to accurately model pre-emptive packet handling is system simulation. We are convinced that our approach will yield a significantly lower modelling effort. Obtaining SEPs mainly involves providing the list of services and request queues of a given protocol, which protocol designers will have readily available. Once the SEP is obtained, it can be made publicly available for reuse by other researchers. Adjusting models in existing network simulators is significantly less work than re-modelling them in a system simulator. SEPs are composed of statistical representations of execution durations, and do not contain any details of the operations performed on packets. Determining simulation durations is simply a matter of distribution sampling rather than actually running or analysing code. Therefore, we are convinced our approach will be scalable.

## 6. FUTURE WORK

We will complete the dynamic part of the tracing framework, and plan to extend LinuxSched to meet our needs for a scheduler simulator. The next step is to implement the co-simulation framework to integrate the modified LinuxSched with ns-3. The design should allow primitives to be inserted in-line with existing network simulator code to synchronize the virtual time in both simulators.

## 7. REFERENCES


[1] R. Chertov, S. Fahmy, and N. B. Shroff. A device-independent router model. INFOCOM, Phoenix, USA, 2008.

[2] S. Kristiansen, M. Lindeberg, D. Rodriguez-Fernandez, and T. Plagemann. On the forwarding capability of mobile handhelds for video streaming over manets. MobiHeld '10, New Delhi, India, 2010.

[3] B. Muller-Rathgeber and H. Rauchfuss. A cosimulation framework for a distributed system of systems. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, Calgary, USA, 2008.

[4] P. Pagano, M. Chitnis, G. Lipari, C. Nastasi, and Y. Liang. Simulating real-time aspects of wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2010:2:1–2:12, April 2010.