Simulating Router- and Domain Characteristics

Florian Baumgartner, Matthias Scheidegger and Torsten Braun

Institute of Computer Science and Applied Mathematics, University of Bern, 3012 Bern, Switzerland
E-mail: baumgart|mscheid|braun@iam.unibe.ch

Abstract

This paper describes the architecture and implementation of a hybrid ns2 based network simulator. The system uses a generic xml description for the simulation jobs and uses the Java Messaging System to communicate with components like a graphical front-end or a toolkit for a visualization of the results. The hybrid simulator is an extension of standard ns2 *, which supports the representation of autonomous systems by delay and loss models and provides a more complex node behavior by adding support for forwarding or processing delays within* ns2 *nodes. An xml scheme has been developed to describe autonomous systems.*

1 Introduction

In traditional packet-based simulators the ”world” is modelled in terms of nodes and links with individual capacities and delay characteristics. When simulating whole Internet domains this approach quickly becomes problematic, due to the sheer amount of events to be processed. A multitude of approaches to this scalability problem have been proposed, each with slightly different application ranges [7, 4, 5] [3] [1].

The hybrid simulation module presented in this paper combines packet-based simulation of ns2 with analytical models by using a hot-plug mechanism, which makes single ns2 nodes behave like whole networks (e.g. autonomous systems). This abstraction allows the user to simulate large topologies in a fraction of the time a full scale packet-based simulation would take. Suggested application areas for the hybrid simulator include end-to-end QoS evaluation of single flows – simulated using traditional packet-based models – over a complex backbone network, or the effect of changes in a backbone network (e.g. addition/removal of links, capacity changes, big changes of network load due to new SLAs, etc.) on flows traversing the domain.

Since the models represent a situation in the real world, measurements from the network are needed to configure them. For traffic load models, these measurements consist of the loads on the inbound links of a domain and their distribution to the outbound links. Furthermore, knowledge of inter-domain topology, inter-domain link capacities and SLSs is required for realistic network modeling.

Since real world data of that kind is not easy to obtain, the approach is evaluated by simulating networks and training the models based on the simulated data. The advantage of this approach is that the results of a modeled network can easily be compared with the node by node simulation. Even if this approach does not require any real world data, it requires a more real world alike behavior of the simulator. Therefore, besides the hot-plug mechanism for domain models, ns2 was further extended to provide support for packet forwarding and processing delays on a per node level. By this we try to model a real network more realistically than this can be done with standard ns2. With the developed extension ns2 nodes cause packet forwarding delays similar to those caused by the packet processing of real IP routers. Using the same mechanism even the behavior of whole network domains can be scalably simulated, using so called multi-domain models [2].

The paper is structured as follows. Section 2 describes the general concept of multi-domain models and gives a short impression of their xml representation. In Section 3 the extensions of the ns2 network simulator are presented. A set of short evaluations show the impact of forwarding delays and how measurement data can be used to obtain a model for a node or a set of nodes. Section 4 focuses on the embedding of the hybrid ns2 simulator in the InterMON framework and describes the automatic processing of simulation jobs.

2 Multi-Domain Models

2.1 Concept

The extension to ns2 mentioned above may be used to simulate the behavior of a huge network inside a single simulator node. To this end we have developed an analytical model capable of simulating scenarios with several ISP networks in a scalable way.

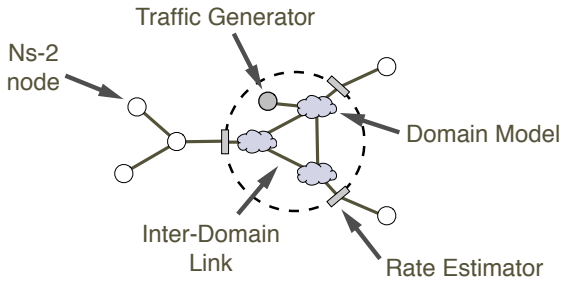


Figure 1. Example of a multi-domain model

This model is based on the assumption that, at a given time, the Internet can be divided into areas where congestion is negligible, interconnected by bottleneck links. We treat congestion free areas as black boxes, which we call *domain models*. Modelling congestion free areas has the advantage that we can neglect packet losses and excessive queuing in large parts of the network and restrict the model to quasi-static delay behavior. Apart from its scalability advantage this approach may be useful to model network areas of which we do not know the exact topology. Domain models use empirical cumulative distribution functions (ECDFs) to simulate the delays of packets crossing the domain. The distribution may vary depending on the ingress and egress nodes on which the packet enters and leaves the domain, respectively.

The bottleneck links between two domains of a simulation scenario are represented by *inter-domain link models*. Here, packet loss and the effects of queuing on delay are simulated. The basic parameters of an inter-domain link model are similar to those of an ns-2 link. In fact, ns-2 links can be used for this purpose. However, it may be more efficient to combine several domain and inter-domain link models to form a so called *multi-domain model*. Unlike ns-2 multi-domain models are not event-driven, but use a system similar to fluid flow simulation. Packet arrival events are converted into a scalar rate estimate for every path going across the model. Additionally, highly scalable *traffic generators* model traffic aggregates (e.g. 1000 VoIP flows) to populate the multi-domain model with traffic. They take the form of a function that yields the load generated by the traffic aggregate given a (monotonously rising) point in time. Fig. 1 gives an example of how these models may be combined to make a multi-domain model.

To make the concept of multi-domain models work we need a way to model the behavior of inter-domain links in a not event-driven fashion. A well-known approach to this problem are analytical queuing models, which take the offered load and calculate the packet loss ratio and the delay distribution from it. A simple example is the M/D/1/K queue: it models a standard router queue with queue capacity $K-1$, deterministic service time and a poisson arrival process. Preliminary evaluation has shown that the choice of arrival process is the critical element when building an inter-domain link model.

```
<imonmodel family="hybrid">
    <modinfo>
        ...Generic Intermon model header...
    </modinfo>
    <model category="multi">
        <border>
            ...Configuration of interface to ns-2...
        </border>
        <submodels>
            ...Paths to the submodels' XMLs...
        </submodels>
        <topology>
            ...Connections between submodels...
        </topology>
        <routing>
            ...Routing paths through the model...
        </routing>
    </model>
</imonmodel>
```

Figure 2. Structure of master configuration file

In the Intermon context it is particularly useful to partition the network into autonomous systems ($\Rightarrow$ domain models) and their inter-connecting links ($\Rightarrow$ inter-domain link models). This partitioning seems reasonable since autonomous systems are usually controlled. For example, internal routes can be changed to distribute traffic load, and ingress routers might police flows to prevent congestion inside the AS. Moreover, the bandwidths inside an AS network are usually bigger than the bandwidths of inter-domain links.

2.2 NS-2 Integration

On the ns-2 level a multi-domain model can be attached to a node by loading the respective module and pointing it to its master configuration file. This file is in XML format and contains the configuration for the module's interface to ns-2, its internal topology and routing. Furthermore, it contains paths to the XML descriptions of all required domain, inter-domain link, and traffic generator models. Fig. 2 shows the basic structure of this file.

3 Simulator Extension

For the integration into ns2 the node has to be slightly modified, as can be seen in Figure 3 (taken from [6]). The typical ns2 node consists of an address classifier, a port classifier and a set of agents. The address classifier routes incoming packets either directly to outgoing links or to a port classifier forwarding the packets to an appropriate agent.

External Predictor Modules

To delay and discard packets, an additional component, the Delay and Loss Predictor (DePred, LoPred) was implemented and put before the address classifier. Therefore incoming packets are first processed by the predictor module, which decides how long the packet has to be delayed, or whether the packet has to be dropped.

The module itself is not part of the ns2 node. The node only provides an interface to the external module and takes care of delaying or dropping the packet. Apart from some functions allowing the dynamic loading and unloading, the initialization and configuration of the module, a predictor module only has to provide a single function which is called for each arriving packet. Depending on the return value, the module interface either discards or delays the packet. With each call of the predictor function, the predictor gets information about the packet itself, the previously passed and the next ns2 node, and the simulation time. Any type of metering has to be done by the predictor itself.

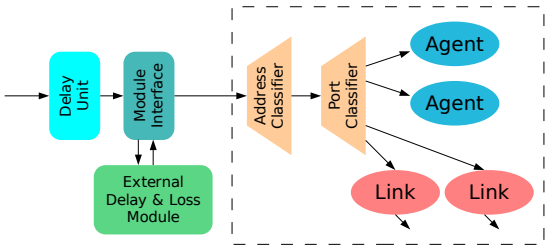


Figure 3. Extension of the ns2 network simulator with an interface for external delay and loss predictor modules and an internal module for standard distributions

Internal Predictor Modules

While the external modules are perfectly suited to model the behavior of complex domains, the module interface can be used to attach different module implementations also. However, since a lot of simulation do not require complex delay patterns, an additional, simple delay module was directly build into the node extension itself. The internal module provides a set of simple distributions which can be used to generate delay patterns for packets passing that node. In general the same functionality could also be provided with external modules, but would increase the overhead. The different types of distributions and how they can be used from within ns2 is discussed in the next section.

3.1 Standard Distributions

Packet forwarding delays can be simply switched on and configured from within any ns2 script. Currently three different distributions are implemented. The syntax for activating packet forwarding delays and for a configuration of the distributions is:

$dm dist gamma <shape> <offset> <scale>
$dm dist gauss <offset> <scale>
$dm dist empiric <filename>

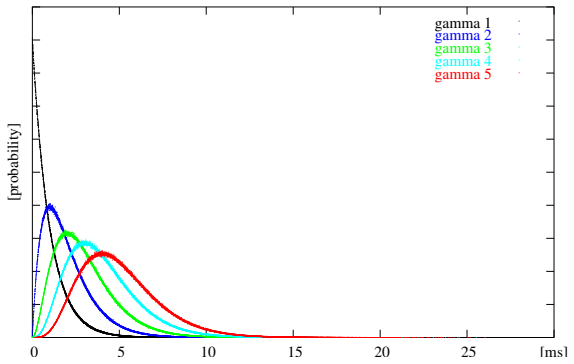
The $dm is the TCL reference to the delay module. The *gamma and* gauss commands activate an accordingly shaped packet forwarding delay distribution. The <offset> and <scale> parameters allow to guarantee minimum delays or to scale the distributions. The gamma distribution has an additional parameter <shape>.

While the gauss distribution was mainly implemented as the classical example of a probability distribution, the gamma distribution was chosen because of some measurement results. The delays between several hosts in the SWITCH network were measured and analyzed. The measurements were performed were only very little congestion had to be expected and showed gamma like distributions. Therefore the gamma distribution has been implemented as a close approximation to that behavior.

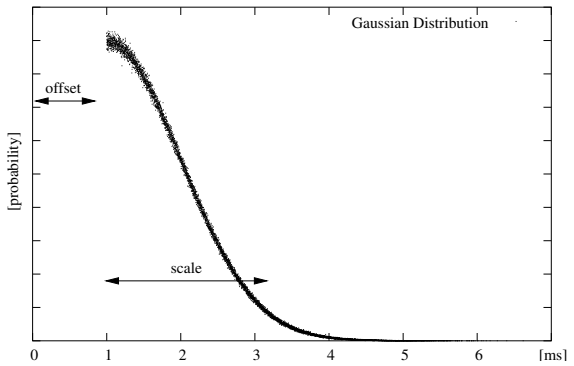
In contrast to the gamma and gauss distribution the empiric distribution expects a filename as an argument. The file has to contain sample delays. After the simulation has started the node will delay packets according to the distribution given in the file.

Figure 4 shows typical gamma and gauss distributions which can be used as delay pattern for the internal delay module. Figure 4(a) shows the gamma distribution with different shapes, while Figure 4(b) shows the gaussian sample distribution. The lower graph also shows the impact of the <offset> and <scale> parameter on the delay distribution. The effect of the parameters on the gamma distribution are similar.

Using this built-in distributions to generate packet forwarding delays allows to model simple node behavior. Especially it is suited to produce more complex delay patterns in ns2 simulations.

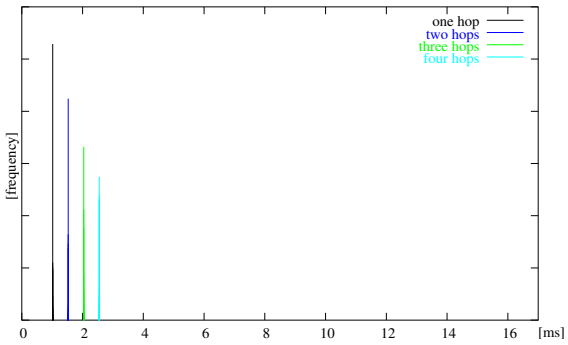


(a) Gamma distribution

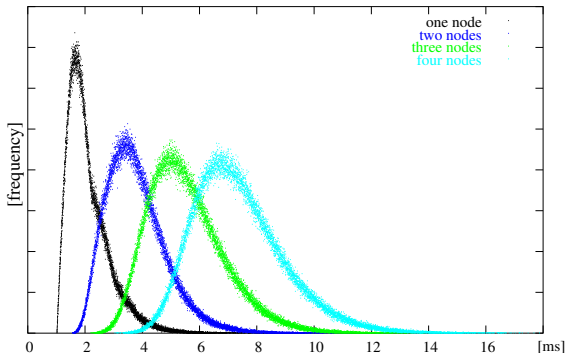


(b) Gauss distribution

Figure 4. Standard distributions with various shapes resp. offset and scale parameters



(a) typical delay distribution of ns2



(b) delay distribution with forwarding delays

Figure 5. Typical ns2 delays in a static simulation and packet delays with the ns2 extension

As shown in Figure 5(a) ns2 generates, without the impact of protocol dynamics, only static delays. Of course, TCP behavior or load changes will result in changing queue lengths and therefore produce a variation of packet delay. However, one of the assumptions of the InterMON project are no packet losses and therefore no congestion for high priority service classes within an AS. Therefore queuing delays can not be used as the only cause of packet delay in our simulations.

In contrast to the standard ns2 simulator, Figure 5(b) shows the delays resulting in a chain of routers with each router causing gamma distributed forwarding delays. The graph clearly demonstrates the convolution of the delay distributions with each hop.

3.2 Training models with Simulation Data

Even if the standard distributions provide a simple model for a router's forwarding delay, the main focus is not so much on providing a realistic model but simply on providing a more complex delay behavior, which is necessary to evaluate the training of models with simulation data.

As was shown in Figure 5 the ns2 link delays have always the exact same value, which does not allow to cause any typical delay distributions for certain nodes or node clusters.

In our experiment a row of six nodes was set up (see left hand diagram in Figure 6). All links between the nodes had the same bandwidth except the one in the middle which has only half the bandwidth and causes packets to be queued and therefore delayed. Each node was configured to cause gamma distributed forwarding delays.

A single CBR traffic source sends packets and fills the bottleneck completely. The simulation was started and the packet delays between the nodes A, B and A, C was measured. Figure 7(a) shows the distribution of the measured delays.

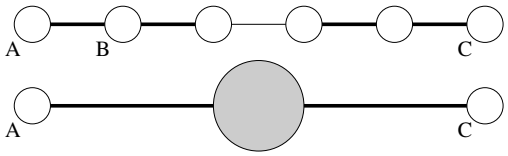


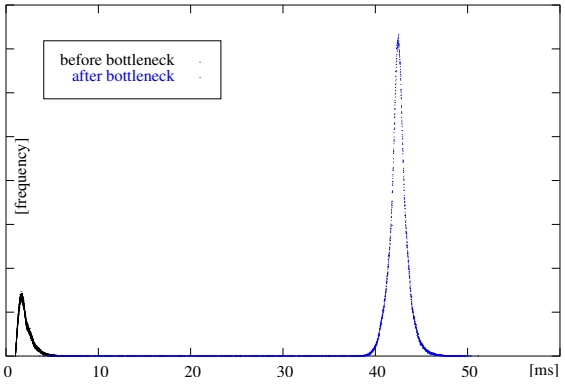
Figure 6. A chain of six `ns2` nodes with a central bottleneck and a similar setup using a domain model

While the delays between A and B show the gamma distribution caused by the first node, the delays after the bottleneck are caused by several reasons:

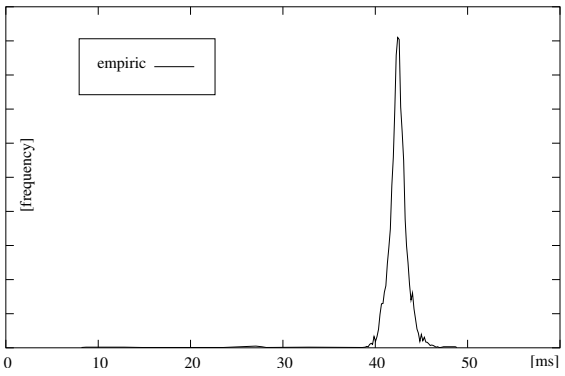
- multiple gamma distributed forwarding delays in the nodes, causing a convolution of the distributions, which alters the shape towards a normal distribution.
- the bottleneck with its queue and the token bucket filter. The queue of packets is responsible for causing the large increase of packet delays, while the token bucket filter modifying the shape.

In the second step we took one of the domain models described in section 2 and trained it the delay and loss measurements between A and C. Then, as shown in Figure 6 the four intermediate nodes in the simulation were replaced with a single node using the domain model for delay and loss prediction. After that the intermediate node causes packet delays and losses according to the training data.

Figure 7(b)) shows the result, which matches the results from the previous simulation quite accurately.



(a) Bottleneck delay distribution



(b) Bottleneck delay distribution (model)

Figure 7. Packet delays by simulation (top) and using a trained domain model (bottom)

4 Processing Simulation Jobs

In this section we describe the mechanisms and formats used to process end to represent simulation jobs.

4.1 Job Description

When the user decides to start a simulation the chosen topology and any changes and additions made to it are written into an XML file to be sent to the simulation manager. Fig. 8 shows the structure of this file. At the top of the file is the BGP topology that was chosen by the user in the graphical user interface. It consists of a list of autonomous systems with their IDs, and possibly IP prefixes, border routers, and recently received BGP path updates. This part is the same for all Intermon simulators.

The `<changes>` section is the second part of a simulation request and contains a simulator dependent list of actions, which are usually actions initiated through the user's GUI, like changing the capacity of a link. Action may also be generated automatically, however. This may serve to supply additional information about the topology to the simulation manager. In the ns-2 hybrid interface, the user can perform the following actions:

- Add a traffic generator to the scenario
- Specifiy transition points between ns-2 and the multi-domain model
- Select the delay behavior for an autonomous system
- Set the capacity of a link

After the changes section an additional `<parameters>` section allows the user to supply general instructions and parameters to the simulator. In the ns-2 hybrid interface the user can choose the data to be returned by the simulator by parameter (delay, packet loss, jitter) and endpoints.

```
<simrequest>
    <BGPTopologyTree>
        ...BGP topology from topology
        detection tool...
        <AS>
            ...AS description, possibly including
            recent BGP path updates...
        </AS>
        ...
        <Link>
            ...Inter-AS link description
        </Link>
        ...
    </BGPTopologyTree>
    <changes>
        ...Simulator dependent changes/additions
        to the scenario...
        <action type="set_link_capacity"
         src="1111" dst="2222">
            1M
            ...Example action changing the link
            capacity between ASs 1111 and 2222...
        </action>
    </changes>
    <params>
        ...General paramenters for the chosen
        simulator...
    </params>
</simrequest>
```

Figure 8. Structure of a simulation request

4.2 Toolmanager and Toolchain

All the components and modules of the InterMON system communicate using the Java Messaging System. Also, the simulators have an JMS fontend to receive simulation jobs and to return the results. Since network simulations require a lot of computing power and also produce a lot of data scalability was a central aspect during the design and the implementation of the architecture. The system supports the processing of simulation requests in parallel and supports multiprocessor computers as well as clusters.

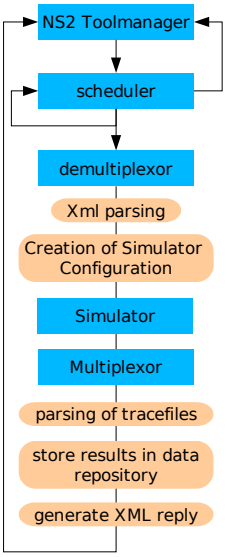


Figure 9. Simulation queue

As can be seen in Figure 9, on top of the toolchain resides the `ns2` toolmanager, which is written in Java and takes care of receiving the simulations job descriptions and which returns the replies. When a simulation job is received, the *toolmanager* automatically returns a short message, signaling the global controller and the graphical user interface that the job was received and will be processed.

After that the job is forwarded to the *scheduler*. The *scheduler* stores the job, initiates the job processing and waits for the next job. After this initiation, the scheduler does not have to wait until the job is finished, but is immediately ready to receive the next job. The scheduler also decides on which computer/processor the particular job is processed and therefore can provide a proper and flexible load balancing.

The processing of the job mainly consists of parsing the xml simulation job description and the creation of the configuration files for the hybrid `ns2` simulator. This is done by the *demultiplexor* component. After the simulation has

been finished, the *multiplexor* components takes care of processing the simulator tracefiles, stores the results in a data repository and produces a message telling the graphical user interface by which URL the results can be obtained.

The data repository is necessary, since the results produced by the simulation may consist of a large amount of data. Due to performance reasons the results are therefore stored on an ftp/web server and merely the URL of the result file is returned to the graphical user interface.

4.3 Processing the Results

The InterMON system includes a module which uses Visual Data Mining techniques to render the results of the simulation. To provide support for different simulators, VDM system does not depend on a specific data format, but requires each simulator to provide an import filter for this simulator's data.

The VDM toolkit is attached to the graphical user interface and receives the simulation result message. It extracts the URL from the simulator reply message, downloads the data from the URL and pipes them through the import filter.

The hybrid simulator returns a single data format which simply lists events between two points A and B in the simulated network. Such an event typically represents a packet passing one of this points. Each event is recorded with the specific simulation. In addition packet losses between the nodes are also recorded.

This allows the VDM system to calculate the delay between specific points within a topology as well as packet losses, packet and data rate. A single simulation job may contain any number of point pairs. Furthermore a granularity can be specified, which causes the simulator toolchain not to list single events in the output data, but to aggregate events. The adjustable granularity allows to provide better performance for large simulations and further improves the scalability.

5 Summary and Outlook

The paper presents the concept of flexible domain models allowing to represent autonomous systems as a single black box model and their integration into a packet based simulation scenario. The simulations are based on `ns2` , which has been extended by a flexible plugin mechanisms allowing to bind external domain models on specific nodes. These nodes then provide the behavior of a whole autonomous system. The models also include internal traffic sources and sinks which allows to internally model huge numbers of VoIP or HTTP streams.

Besides the integration of domain models in `ns2` the simulator was also extended to support forwarding delays. A small set of standard delay distributions have been implemented and can be used to provide a more complex and more realistic delay behavior. A set of simple tests showed the impact of the additional packet delay mechanism and how simulation data can be used to create models for a set of nodes within a simulation and compares the results of the original simulation with the model supported one.

Future work will focus on a more detailed evaluation of the implemented system. Results obtained with the hybrid `ns2` will be compared with other simulators, implemented within the InterMON project.

Acknowlegement

The work presented in this paper was carried out within the IST project "Advanced Architecture for INTERdomain Quality of Service MONitoring, modelling and visualization" (INTERMON), funded by the European Community and the Swiss "Bundesamt für Bildung und Wissenschaft (BBW)".

References

[1] Jong Suk Ahn and Peter B. Danzig. Packet network simulation: speedup and accuracy versus timing granularity. *IEEE/ACM Transactions on Networking*, 4(5):743–757, October 1996.

[2] Florian Baumgartner, Matthias Scheidegger, and Torsten Braun. Enhancing discrete event network simulators with analytical network cloud models. In *International Workshop Inter-domain Performance and Simulation, Salzburg*, pages 21–30, February 2003.

[3] Yang Guo, Weibo Gong, and Don Towsley. Time-stepped hybrid simulation (TSHS) for large scale networks. In *Proceedings of IEEE Infocom*, March 2000.

[4] B. Liu, Y. Guo, J. Kurose, D. Towsley, and W. Gong. Fluid simulation of large scale networks: Issues and tradeoffs. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 2136–2142, Las Vegas, NV, June 1999.

[5] Benyuan Liu, Daniel R. Figueirido, Yang Guo, Jim Kurose, and Don Towsley. A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation. In *Proceedings of IEEE Infocom*, April 2001.

[6] Various. The ns manual. http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf, June 2002.

[7] A. Yan and W. B. Gong. Fluid simulation for high speed networks with flow-based routing. *IEEE Transactions on Information Theory*, pages 1588–1599, 1999.