Modeling Computer Networks for Emulation

Daniel Herrscher, Alexander Leonhardi, Kurt Rothermel
University of Stuttgart
Institute of Parallel and Distributed High-Performance Systems (IPVR)
Breitwiesenstr. 20-22
70565 Stuttgart, Germany

Abstract -- The Network Emulation Testbed project provides a configurable network environment for the performance analysis of distributed applications and protocols. An emulation scenario consists of a network topology, link parameters, and dynamic change models. In order to set up a scenario automatically, it has to be modeled in an appropriate description language. In this paper, we analyze the requirements for an emulation scenario description language and propose a possible solution.

Keywords: Network Emulation, Scenario Modeling, Network Topologies

1 Introduction

Performance measurement of distributed systems and network protocols requires the target environment of the respective system. In most cases, though, this environment is not available to the performance analyst. To allow a wide variety of measurements, a synthetic, *emulated* network environment is needed. The Network Emulation Testbed (NET), which is currently being set up at the University of Stuttgart, consists of a large number of computing nodes (40+) connected by a flexible network infrastructure. The network traffic on each link can be affected by special *traffic shaper* modules.

To set up a network scenario, an appropriate specification language is required. The scenario specification has to describe both the network topology and certain link properties. The topology description concept has to include point-to-point connections as well as multipoint-connections like LANs. Regarding the connections, all properties affecting the network performance in any way have to be modeled. Since in real networks, some properties will change over time, a realistic network model has to reflect such changes as well.

In this paper, we investigate the essential properties of a network modeling language to describe an emulation scenario. As a possible solution, we propose a modeling language that includes these parameters and offers concepts for dynamic changes. The proposed language will be the basis for further research in building a comprehensive network emulation testbed.

2 Related Work

Modeling the properties of computer networks is the first step for both simulation and emulation. Because network simulation predates emulation approaches by a number of years, there are already sophisticated network models for simulation engines.

The most important general-purpose network simulator today is ns-2 [1]. Since ns-2 is widely used in network simulation, its modeling language is extensive. Instead of using a general modeling language, the ns-2 developers chose to model their scenarios using a programming language. A program in a script language (OTcl) is used to instantiate the required simulation objects and to set up their parameters and behavior. The parameters can be either fixed or determined by a variety of dynamic models. As a result, the execution of a scenario declaration script yields a specialized simulator instance for the desired purpose. If new types of simulation objects are added to ns-2, they can easily be referenced by scenario scripts just like any predefined object.

Compared to simulation, network emulation approaches are much less common. So far, the majority of emulation projects just propose a tool to intercept and affect network traffic. These projects consider only small scenarios consisting of a few machines. Therefore, it is sufficient for

them to set up each instance of the respective tool manually. In this case, there is no comprehensive network model but only the combination of several setup commands for each machine.

Existing emulation tools differ in the parameters they can emulate. Delayline [4] can introduce propagation delay and allows to specify a packet loss probability. In addition to that, Dummynet [5] comes with bandwidth limitation and an adjustable queue size. NISTNet [2] has a fixed queue size, but adds delay variation, packet reordering, and packet duplication.

The Utah Network Testbed [5] is an emulation project that provides a larger testbed based on a number of machines running traffic shaping tools. Several local Dummynet instances are used to emulate a comprehensive scenario. The desired topology and the respective link parameters are specified in a script language similar to ns-2 OTcl-scripts. In fact, they use a subset of the ns-2 setup language, extended by some special commands. However, the dynamic models of ns-2 are not supported.

There are a number of general network topology description languages that do not aim specifically at emulation or simulation. Most of them are limited to node-link-structures, again with various properties for the links. Some include the modeling of LANs, but transform them into stars or complete connected graphs to fit in the node-link-model [9]. Since these approaches aim at describing the state of a network at a specific time, they do not include dynamic changes at all.

3 The Network Emulation Testbed

To make it clear in which context our network modeling language will be used, we first give a brief overview of the Network Emulation Testbed, which is currently being set up at the University of Stuttgart.

The testbed basically consists of a 40+ node PC cluster system running Linux (see fig. 1). The node PCs are connected with both a Gigabit Ethernet switch and a Fast Ethernet switch. While the Fast Ethernet connects the system for administrative purposes, the Gigabit switch can be used to establish a number of VLANs (virtual LANs), each containing arbitrary node PCs. A VLAN with only two participating nodes would be equivalent to a single connection between these two nodes. A VLAN with more than two nodes is best viewed as a separate, private LAN connecting these nodes. If a node is part of several VLANs, Linux can address each VLAN by a separate virtual network device, transparent to protocols and applications. Through this concept, VLANs can be used to set up any virtual topology.

In addition to the desired topology, the properties of each connection have to be emulated. This is done by traffic shaper modules running on the nodes. Once configured, they introduce the specified link properties like delay, bandwidth limitation, packet loss etc. Since they are running in

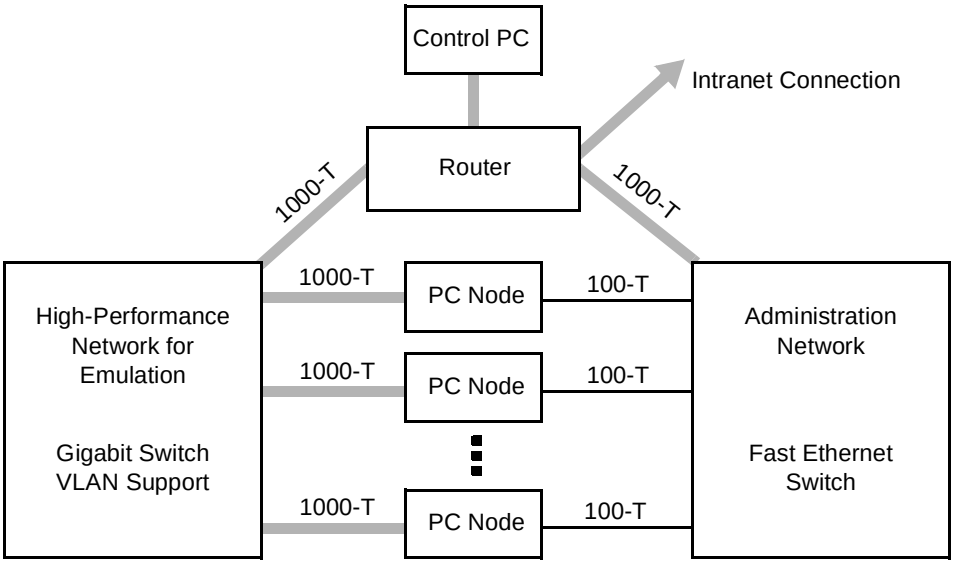


Figure 1: The Network Emulation Testbed Hardware.

kernel space, the user space software on the nodes is not aware of them. For the time being, we use NISTNet [2] to shape traffic. Because of certain limitations of existing tools, however, we are currently developing a traffic shaper on our own.

A central administrative node is responsible for setting up the network scenario and controlling the experiment. Once a network model is provided to the administrative node, the scenario creation will run automatically.

4 Modeling Language Requirements

In this paragraph, we will consider the requirements for a modeling language to describe a link-based emulation scenario.

4.1 Topology

Topology modeling for emulation has to provide node-link structures, of course. However, since the common media in multipoint connections (like LANs) have significant effects on performance, it is not sufficient to model them as stars or complete connected graphs. They have to be modeled in their original connection topology to emulate the performance properly. Therefore, the model has to provide both point-to-point links and multipoint links.

4.2 Parameters

The parameters considered in a network modeling language for emulation depend on the granularity of the emulation environment. Some tools aim at emulating subnets, consisting of several links, routers, etc. In this case, the emulation has to reproduce parameters like packet reordering, delay variation because of changing router queues, etc.

Because the NET project provides each emulated link with a separate traffic shaper, we can concentrate on the parameters of a single network connection, including multipoint connections like LANs. This consideration leads to the following set of parameters.

1) Bandwidth Limitation

The maximum bandwidth of a connection is one of the most important parameters affecting network performance. A reasonable approach to introduce bandwidth limitation is a “leaky bucket” [8]. A leaky bucket limits outgoing traffic to the specified bandwidth. Surplus traffic is stored in a FIFO queue. If the queue is full, incoming packets are dropped. Leaky bucket behavior is determined by the parameters *bandwidth* and the *queue length.*

Note that there can be different types of bandwidth limitation regarding a two-way connection. The bandwidth can be limited for each direction *separately* (with equal or unequal limits) or there can be a *common* bandwidth limitation for both directions. The bandwidth might also be shared by all participants of a multipoint connection.

2) Delay

The delay a packet experiences between two adjacent hosts consists of several components: propagation delay, serialization delay, medium access delay, and queueing delay.

The *propagation delay* depends on the type and the length of the propagation media. It may also include some delay introduced by additional network components that are part of a link (e.g. repeaters). Unless a communication partner is moving, the propagation delay will remain constant.

The *serialization delay* is introduced by the limited bandwidth of a link and represents the time it takes to send and receive a packet. It depends on the *bandwidth* and the respective *packet size*.

Some link layer protocols may introduce an additional *medium access delay* because they have to wait before they can start sending bits. This time may vary. For all common protocols, though, stochastic models exist that describe this behaviour.

Queueing delay is the time packets wait in FIFO queues to get on an occupied line. Queue lengths vary over time and so does the queueing delay. The queueing delay does not have to be modeled separately, because the queueing mechanisms we use for the bandwidth limitation include the delay of packets (see above).

Summarizing the delay types to be modeled for emulation, there is a *fixed delay* (the propagation delay) and an additional *variable delay* (the medium access delay). The other delay types can be either computed from other parameters (serialization delay) or come for free (queueing delay).

Note that, unlike the packet delay variation that can be measured in subnets, the variable delay on link level does never lead to packet reorderings. Emulation tool implementations must take care of this characteristic.

3) Packet Loss

Packets can be lost on network links for two reasons: They are dropped because of link congestion, or they are damaged due to transmission errors. Our bandwidth limitation approach already drops surplus packets in case an emulated link is congested. Packet loss because of transmission errors highly depends on the type of the network link and has to be included in the emulation model. The packet loss probability is a very dynamic parameter: Since some lines tend to burst losses, it may be not sufficient just to include a single loss probability value in the model. A realistic emulation model has to provide a more flexible approach.

4) Parameters Summary

One could think about including further parameters for the emulation of network links. *Packet reordering* and *duplicate packets* happen in subnets, as a result of packet loss or dynamic routing algorithms. However, these effects do not appear on link level. If an emulation scenario is set up properly on link level, reordering and duplication will automatically emerge from the interaction of higher-level protocols and emulated packet loss.

Summarizing the set of parameters to be modeled for a point-to-point or multipoint link:

- *bandwidth limitation:* maximum bandwidth (bit/s), queue length (bytes or packets), type of bandwidth sharing (simplex, half-duplex, or duplex)
- *delay:* fixed delay (ms), variable delay (stochastic function leading to a ms value)
- *packet loss:* probability value, additional burst loss model

5) Dynamic Parameters

The variable delay component always has to be modeled by a dynamic model. Though, the other parameters could also be subject to change: The packet loss, which may be a dynamic model itself, can change significantly over time, especially in wireless networks [6]. The bandwidth limitation may also change (consider the adaptive bandwidth in 802.11b). Even the propagation delay can change if a communication partner is moving.

Therefore, a realistic emulation model has to support both fixed values and dynamic models to specify parameters.

5 XML-Based Approach

In this paragraph, we will propose a specification language that includes the parameters stated above. While the only existing modeling language for network emulation [5] uses an ns-2-based programming language, we decided to use an XML-based description language rather than a programming language for several reasons:

- XML code is easy to understand, process, and extend.
- Descriptions in XML are more likely to be interchanged with other projects than scripts in a special language.
- The main reason why the ns-2 network specification is based on a programming language is the seamless integration of the setup code with the simulator code. This benefit does not hold for the emulation case, because the emulation tools are distributed on a number of machines. There is no single “emulator engine.”

5.1 Topology and Parameters

Because of space limits, we are not able to present the whole XML DTD here. The documented DTD can be found at our project webpage.[1]

Figure 2 shows the basic structure of the topology description: A number of network links is defined by a unique ID and the respective parameters. Each parameter has a predefined unit (e.g., “Kbps” for the bandwidth). Units can be overridden by explicitly specifying a unit. If parameters are omitted, they are given reasonable default values (e.g., zero delay, no loss). Some parameters may be given for each direction separately (cp. the different bandwidths of an ADSL link). In addi-

[1] http://www.informatik.uni-stuttgart.de/ipvr/vs/en/projects/net/

tion to the specified parameters, the network type can be specified optionally (e.g., “Token Ring”). For some network types, certain properties can be emulated by special algorithms rather than described by general dynamic models (e.g., emulated token passing can emulate the medium access delay of a token ring better than a stochastic approach).

Subsequently, hosts with one or more network interfaces are defined. Each network interface is given a symbolic network address and is linked to a network by its unique ID. The relative position of a host within a network is defined by an optional attribute, if necessary (e.g., for asymmetric links).

5.2 Dynamic Parameters

As stated above, a realistic network model must include dynamic parameters. The variable delay component is always a dynamic parameter (if present). The other network parameters can be either set to a fixed value, or be defined by a dynamic model. There are two different approaches to include dynamic parameter models:

- Time based: At certain times, the respective value is changed, according to a change model. This approach is suitable for most trace-based models.
- Packet based: Dynamic models that describe packet-level behavior (e.g. medium access) may need to be evaluated for each packet separately.

Since both approaches have reasonable uses, our dynamic models can be specified both time and packet based.

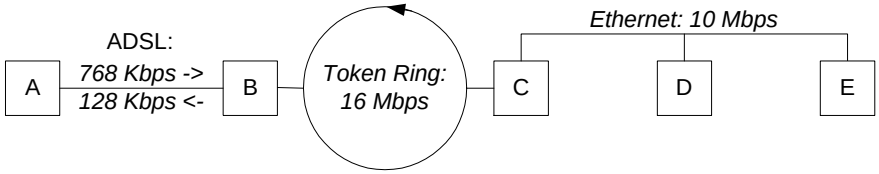
There is a variety of conceivable models to introduce dynamic changes. We propose three basic models. Further models will be added when new requirements arise. So far, our description language supports a table model, the Gaussian distribution, and Markov state models.

1) Table

The simplest dynamic model is the *table* model. Instead of a static value, a table is provided that consists of tuples with triggers and actions. Each action can be a parameter change or another dynamic model. Tables are especially suited to replay parameters gathered from measurements. For example, the changing bandwidth of a mobile 802.11b node can be easily modeled that way (see fig. 3). The triggers can be either time values or packet counts. In most cases, a time triggered table will be appropriate.

2) Gaussian Distribution

Especially for the variable delay component, a stochastic function is needed to specify the properties of the variation. The Gaussian distribution will cover basic needs and is specified by the combination of an expected value and the mean



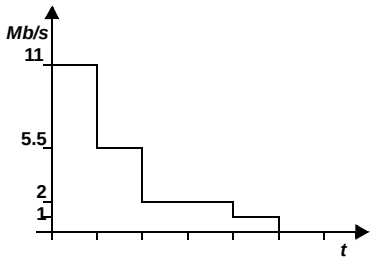

```
<topology>
  <network id="ADSL">
    <bandwidth type="simplex" direction="up">
      <value>128</value>
    </bandwidth>
    <bandwidth type="simplex" direction="down">
      <value>768</value>
    </bandwidth>
  </network>
  <network id="Ring" type="tokenring">
    <bandwidth type="half-duplex">
      <value>16 Mbps</value>
    </bandwidth>
  </network>
[...]
```

```
<host id="A">
  <adaptor>
    <dnsname>ModemA</dnsname>
    <memberof position="0">ADSL</memberof>
  </adaptor>
</host>
<host id="B">
  <adaptor>
    <dnsname>ModemB</dnsname>
    <memberof position="1">ADSL</memberof>
  </adaptor>
  <adaptor>
    <dnsname>RingB</dnsname>
    <memberof position="0">Ring</memberof>
  </adaptor>
</host>
```

```
<host id="C">
  <adaptor>
    <dnsname>RingC</dnsname>
    <memberof position="1">Ring</memberof>
  </adaptor>
  <adaptor>
    <dnsname>EtherC</dnsname>
    <memberof>Ether</memberof>
  </adaptor>
</host>
[...]
</topology>
```

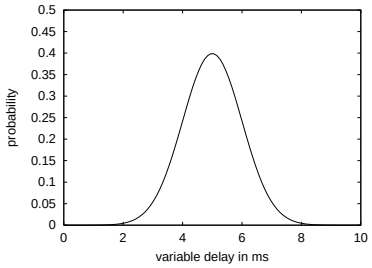
Figure 2: Topology modeling example.

deviation (see fig. 4). Note that it is reasonable to compute the variable delay per packet and not per time unit.




```
...
<bandwidth type = "half-duplex">
  <table base="ms">
    <action at="0"> <value>11000</value> </action>
    <action at="2000"> <value>5500</value> </action>
    <action at="4000"> <value>2000</value> </action>
    <action at="8000"> <value>1000</value> </action>
    <action at="10000"> <value>0</value> </action>
  </table>
</bandwidth>
...
```

Figure 3: Modeling the bandwidth of a wireless node moving away from a base station.

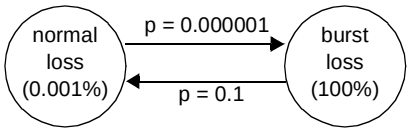



```
...
<variable_delay>
      <gaussian base=”packets” mean="5" deviation="1" />
</variable_delay>
...
```

Figure 4: Modeling variable delay by a probability variable with Gaussian distribution.

3) Markov State Model

Single probability values are not sufficient to describe all nondeterministic behavior, especially traffic and loss patterns [3]. Markov state models are much more powerful, yet easy to understand. The example in fig. 5 uses an ON-OFF-model, the simplest form of Markov state models, to describe a burst loss behavior. Again, the state transition probabilities can be either packet based (e.g., for loss models) or time based (e.g. for changing bandwidth over time).




```
...
<loss>
  <markov initial="normal" base="packets">
    <state id="normal">
      <value>0.00001</value>
      <transition id="burst" probability="0.000001" />
    </state>
    <state id="burst">
      <value>1</value>
      <transition id="normal" probability="0.1" />
    </state>
  </markov>
</loss>
...
```

Figure 5: Markov state model to switch between sporadic and burst loss.

4) Combination of Dynamic Models

Of course, the modeling possibilities become even more interesting if several dynamic models are combined. In our language, every occurrence of a numerical value can be replaced by a dynamic model. Therefore, it is possible to create combinations like a table that switches between different Markov models that define the packet loss behavior, one loss model for a static and another one for a moving node.

5.3 Dynamic Topologies

The dynamic models investigated above varied the properties of existing connections only. We did not address the modeling and emulation of dynamic topologies so far. For the time being, changes in the connection topology (e.g. a node moving from one network to another) can be modeled with our approach by a workaround: A node moving from network A to B is set up with connections to both networks initially. To cut off inactive connections, a table is defined that simply sets the bandwidth to zero at the appropriate times.

6 Conclusion and Future Work

In order to set up and run a network emulation environment, a comprehensive, dynamic network model is needed. We identified the parameters that have to be modeled for emulations at link level. We proposed an XML-based approach to model a network scenario with these parameters,

and completed the model with a concept to model dynamic changes during the emulation process. Our modeling language will be the basis for further research on building a comprehensive network emulation environment for performance measurement.

We named the parameters needed for realistic link-based network emulation above. However, existing traffic shaper tools are not yet able to emulate all of these parameters. Models including the dynamic change of parameters are also missing in current emulation tools. Therefore, a new emulation tool has to be developed that suits the needs for realistic link-based network emulation.

To ease the parallel use of simulation and emulation tools, it would also be desirable to have a compiler that generates XML-based emulation topologies from ns-2 simulation scripts. For easy (i.e. static) topologies, this should be feasible.

We presented a workaround to facilitate dynamic connection topologies. However, a more general approach to model and emulate mobility in networks is needed. This approach has to consider both deterministic and nondeterministic movement models, managed either centralized or distributed.

7 References

[1] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in Network Simulation. *IEEE Computer,* Vol. 33, No. 5, pp. 59-67, 2000.

[2] M. Carson. NISTNet Network Emulation. *http://www.antd.nist.gov/itg/nistnet/*

[3] D. Eckhardt and P. Steenkiste. Measurement and analysis of the error characteristics of an in-building wireless network. In *Proceedings of the ACM SIGCOMM '96,* pp. 243-254, Stanford, 1996.

[4] D. Ingham and G. Parrington. Delayline: A Wide-Area Network Emulation Tool. In *Computing Systems,* Vol. 7, No. 3, pp. 313-332, 1994.

[5] J. Lepreau, C. Alfeld, D. Andersen, K. van Maren. A Large-Scale Network Testbed. *SIGCOMM '99 New Research Session,* Cambridge, 1999.

[6] B. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz. Trace-Based Mobile Network Emulation. In *Proceedings of the ACM SIGCOMM '97,* pp. 51-61, Cannes, 1997.

[7] L. Rizzo. Dummynet: A simple approach to the evaluation of network protocols. In *ACM Computer Communication Review,* Vol. 27, No. 1, pp. 31-41, 1997.

[8] J. Turner. New directions in communications (or which way to the information age). *IEEE Communications Magazine,* Vol. 24, No. 10, pp. 8-15, 1986.

[9] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of the IEEE INFOCOM '96,* pp. 594-602, San Francisco, 1996.