[HPSSt03] *Oliver Heckmann, Krishna Pandit, Jens Schmitt, Ralf Steinmetz,* **KOM ScenGen - The Swiss Army Knife For Simulation And Emulation Experiments.** Multimedia Interactive Protocols and Systems (MIPS 2003), Neapel, November 2003, S. 91-106.

KOM ScenGen
The Swiss Army Knife For Simulation And Emulation Experiments

Oliver Heckmann, Krishna Pandit, Jens Schmitt, Ralf Steinmetz

KOM Multimedia Communieations Lab
Department for Eleetrieal Engineering and Information Teehnology
& Department for Computer Seienee
Darmstadt University of Teehnology
Merekstr. 25, 64283 Darmstadt, Germany

{heekmann,pandit,sehmitt,steinmetz}@kom.tu-darmstadt.de

Abstract Multimedia networking involves complex collections of protocols, in particular protocols that support the inherent quality of service (QoS) requirements of multimedia applications. Most often analytical treatment falls short in being able to assess the overall system behaviour or performance. However, also simulation and testbed experiments alone often leave uneasiness with the results they deliver. The combination of simulation and testbed experiments promises to avoid most disadvantages that their isolated usage bears.
In this paper, we discuss the KOM Scenario Generator, a tool that supports the integration of simulation and testbed experiments for system-wide assessment of design alternatives in particular in the complex environment of distributed multimedia systems.
This paper also systematically analyses the different steps in ereating a research seenario. Even if one is not interested in eombining simulations and testbed experiments our scenario generator is a helpful tool because it systematically integrates and supports all the different steps in creating a eomplex network research scenario from topology creation over traffic generation to evaluation.

1 Introduction

In this paper we introduce the so-called KOM scenario generator (KOM ScenGen) that we developed in the context of the LETSQoS project (www.letsqos.de). Within the LETSQoS projeet we compare different QoS technologies (Intserv/RSVP [1], Diffserv in different flavors [2], ABE [3], Price Controlled Best Effort [4], Load Control Gateways [5], etc.) using simulations and testbed experiments. We conduct different experiments. For eaeh QoS technology we estimate the overprovisioning factor that a best-effort network must be designed with to match the QoS teehnology. Further, we measure the utility for a wide range of different application mixes when different QoS technologies are switched on. Finally we estimate scaling behavior by repeating the utility measurement experiments with searce router resources. To support the wide range of technologies and experiments we developed a seenario generator that supports the manual and automatic creation of experimentation scenarios for network research from the topology creation over traffic generation to evaluation. We believe our approach is very general and that our seenario generator can be helpful for other researchers and in other areas of multimedia network research as well.
We next continue motivating our approach. An overview of the scenario generator, definitions of the terminology used and related work are discussed in the third seetion. In the fourth seetion we diseuss the different steps of seenario generation and how they are supported by the scenario generator in more detail. We conclude with a summary and a pointer

to a video that demonstrates the KOM ScenGen at work.

2 Motivation

Multimedia network research ean be eonducted using analytieal methods, simulation, testbed and real-world experiments. All these methods have their advantages and disadvantages.

Simulations for example are relatively cheap to earry out. However, realistic simulations depend on the eorrect and realistic setting of simulation parameters and models. For analyzing a protocol the simulator cannot be used with the original protoeol without reimplementing the protocol. The eosts of eertain operations (e.g. a routing lookup) arc hard to estimate by simulation. This is even harder if the code basis for the simulation is different from the eode of a real-world router - something whieh is almost always the ease. Beeause of this if we want to analyse the performanee of e.g., RSVP, this ean hardly be done by solely a simulation [6].

For simulation models a eertain level of abstraction is neeessary, often it is hard to judge how the realism of the results suffers by these abstraetions. Though researchers have to rely on the correctness of the simulation models and protoeol implementations (as for example the TCP implementation), verification of those models is very hard [7], espeeially as they typically have a eompletely different code basis than real-world implementations.

The realistic setting of parameters is easier in testbed experiments. Also, the eosts of operations ean be measured far better in testbed experiments - however one has to admit that these measurements are still only valid for the hardware and software platform used in the testbed. Other types of routers may behave differently.

While a testbed is still not the real world at least it is possible to use real-world applications and protocols (e.g. TCP, eurrent web browsers, FTP server and clients) for the testbed experiments whieh increases the realism and decreases the ehanees of unrealistic and misleading results beeause of wrong models, bad implementations or negleeted details.

So testbed experiments do not share the disadvantages of simulations but they have their own set of disadvantages. Testbeds are generally relatively expensive. Also testbeds are typically difficult to eonfigure and reconfigure. And they are limited in seale. Experiments with hundreds of nodes ean usually not be performed in a testbed. The realism of results based on experiments with very few nodes, whieh are typical for testbed experiments, is hard to judge.

Based on these observations it makes sense to combine simulations and testbed experiments, this is also reeommended in literature [8, 9, 10] but not often found in actual research work. The main reason is that most tools are specialized for simulation or for emulation and that thus doing simulation and emulation leads to nearly twiee the effort than using either simulation or emulation. This was our motivation for KOM ScenGen which supports in an integrated fashion simulation and emulation experiments at the same time.

With KOM ScenGen testbed experiments ean be conducted to establish realistie simulation parameters and to ercate referenee data with which the later simulation results ean be compared. The eosts of operations can be estimated in the testbed results. Only small testbed experiments have to be eonducted, whieh decreases the costs of the testbeds as larger experiments with larger topologies, higher bandwidth or more flows can be eonduetcd using simulation.

Even if one is not interested in combining simulations and testbed experiments KOM ScenGen and this paper are useful. Seientists should be awarc of the different steps undertaken when doing a simulation or testbed experiment. In this paper we analyse and discuss the different steps in creating and testing a network research scenario from topology ercation over traffic generation to evaluation. KOM ScenGen supports all the steps without mixing them up.

3 Overview

In this section, we give an overview over the terminology we use in this paper and over the different steps in scenario generation. Those steps will be discussed in the following sec-

tion in detail. We also discuss related work.

3.1 Terminology

Traffic The term "traffic" is used to describe the amount of bits that are sent over one link or are output by a node. Traffic can be described in several ways with an increasing level of abstraction (see section 4.3.1). With the term traffic we always mean Internet (IP) traffic.

Network simulation In network simulation computer models of real network components are used to estimate the behavior of the network to some input with regard to typical networking parameters like loss, delay, throughput. Network simulators like NS2 [11, 8], JavaSim [12], OpNet [13] etc. are used for network simulation. We use NS2 for our simulations.

(Real-world/Testbed) experiment In a real-world or a testbed experiment the behavior of a network to some input is estimated based on measurements made in a real physically existing computer network, either a testbed, research network or production network.

(Simulation/Experiment) Scenario By the term "scenario" we describe the simulation and/or experiment setup, execution and evaluation. The scenario includes all parameters needed for the simulation and the experiments, e.g. topology, link and node properties, traffic mix, simulation/experiment parameters, measurement points, etc.

Traffic simulator A traffic simulator delivers traffic input for a network simulator (e.g. for NS2).

Traffic emulator A traffic emulator emulates traffic by sending real packets using a network interface like an ethernet card. Both the traffic simulator and emulator contain a module that generates the data structures that resemble traffic, we call this the traffic generator.

Traffic generator The traffic generator artificially generates traffic. Traffic is derived from so called traffic models. The generated traffic can be used for simulation and/or emulation.

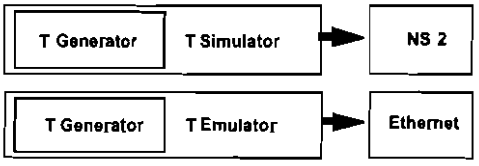

Figure 1. Traffic Generators as part of SimulatorsandEmulators

Traffic model A traffic model describes in a general way how traffic of one kind can be generated. Different kinds of traffic models are used to describe different types of traffic like: Voice over IP traffic, Web traffic, Telnet traffic, etc. Traffic models are discussed in section 4.3.1.

Network load With "network load" we characterize the traffic for each node of a given network topology.

Load generator A load generator generates network load that is traffic for all nodes of a given topology.

3.2 Generating Scenarios

The different steps in generating a scenario are depicted in figure 2. All of them are supported by the KOM scenario generator.

In the first step, a topology is created manually or automatically. Then the properties of the links and nodes (e.g. bandwidth, queuing algorithm) are set manually or automatically. Also the traffic parameters for the scenario have to be set. Next the network load which is the traffic of all nodes is created. This step can be followed by a plausibility check where several things critical for the scenario can be checked for plausibility. An example would be estimating the bandwidth necessary for the generated traffic and comparing it with the available bandwidth. If much more bandwidth is needed than offered, the operator might want to change the scenario parameters. After the plausibility check the scenario is exported to NS2 for simulation and/or to a collection of scripts and configuration files that are used to setup the scenario in a testbed. The next step can be to manually adapt the NS2 files or the scripts and configuration files for specific needs. After that the simulation or experiment can be conducted and in the last step be evaluated.

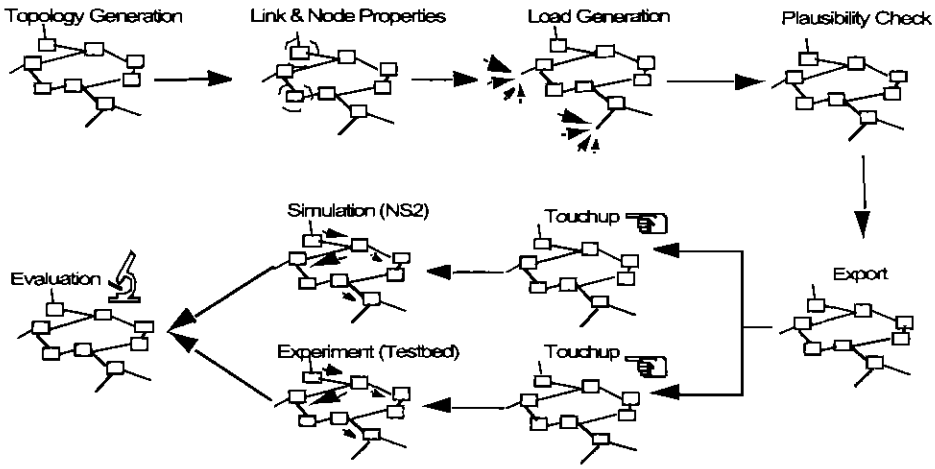

Figure 2. Scenario Generation

3.3 Related Work

For scenario generation many people use small self-written and usually non-published scripts. We believe that being aware of the different steps discussed above helps writing better scripts. I also helps developing tools for one step which can be more easily reused and combined with other tools. The tools which KOM ScenGen consists all have clear interfaces and are all focused on one specific task and can thus be reused easily in different contexts. Using the concept of KOM ScenGen eases the writing of scenario-creating scripts and understanding / reusing third-party scenario-creating scripts.

For the NS2 simulator [11] a simple seenario generator exists [14] plus modifications for QoS scenarios [15]. The NS2 scenario generator lacks many features we deem important like support for testbed experiments and for other topology generators apart from GT-ITM. Also it only supports the built-in (low-level) traffic models of NS2 while we aim for a different (more application oriented) approach to traffic generation (see section 4.3).

For mobility scenarios some scenario generators for NS2 exist [16], [17]. They however focus on mobility models and are for NS2 only while we aim for an integrated approach for non-mobility scenarios. Also they do not offer support for emulation.

With respect to combining network emulation and simulation there are also efforts in NS2 [56], which however rather aim at combined experiments where some part of the scenario is simulated and other parts are emulated whereas we focus on parallel, yet mutually supporting simulation and testbed experiments.

In the context of adhoc routing protocols [10] allows to share the codebase between simulation (NS2) and emulation (Linux/FreeBSD with the Click modular router [18]).

The network emulation testbed (NET) projeet [19] defines a detailed network scenario deseription language based on XML for link-based emulation. We use a similar but much less complicated deseription format in our scenario gencrator and concentrate on support for all steps of scenario generation. Also we eoneentrate on smaller lab testbeds while NET is focused on the 64 machine testbed of the university of Stuttgart. Opposite to us they offer no simulation support.

In our experiments for the Market Managed Multiservice Internet project (M3I, www.m3i.org) we successfully integrated simulation and emulation experiments [5].

There is a lot of work that is related to the individual steps of scenario gencration. These works will be presented when the relevant step is diseussed in the next section.

4 Scenario Generation

The KOM Scenario Generator is a collection of integrated tools and file format specifica-

tions and supports all steps for generating networking scenarios for simulation and testbed experiments. We now discuss the individual steps and how they are supported by the KOM Scenario Generator.

4.1 Topology Creation

When setting up a scenario, first the underlying network topology has to be created. We have started to collect a library of real-world router and POP level topologies which is publicly available at www.kom.e-teehnik.tu-darmstadt.de/~heckmann/topologies/. Instead of using a topology from the library the topology can be created manually with the scenario generator GUI or imported from one of the following topology generators:

- TIERS Random Network Topology Generator [20]
- BRITE - Boston University Representative Internet Toplogy Generator [21]
- GT-ITM - Georgia Tech Internetwork Topology Models [22]
- Inet - AS Level Network Topology Generator [23].

The converter written to import topologies from these generators can be used independently from the scenario generator, it can also read NLANR topology files. It is written in Java and available at http://www.kom.e-technik.tu-darmstadt.de/~heckmann/topologies/.

We have also investigated how to choose the parameters of the topology generators in order to obtain realistic topologies. The results show that the topology generators above can indeed produce realistic topologies with respect to outdegree distribution, the hop-plot and some other metrics, for details see [24].

4.2 Setting the Node & Link Properties

After the basic topology is created the properties of the nodes and links have to be specified. Example properties are:

- Bandwidth
- Propagation Delay
- Queue Length
- Queuing Algorithm, RED parameters, ...

Depending on the kind of scenario other properties are important, too. For a QoS scenario DiffServ or IntServ/ RSVP parameters have to be set for the node.

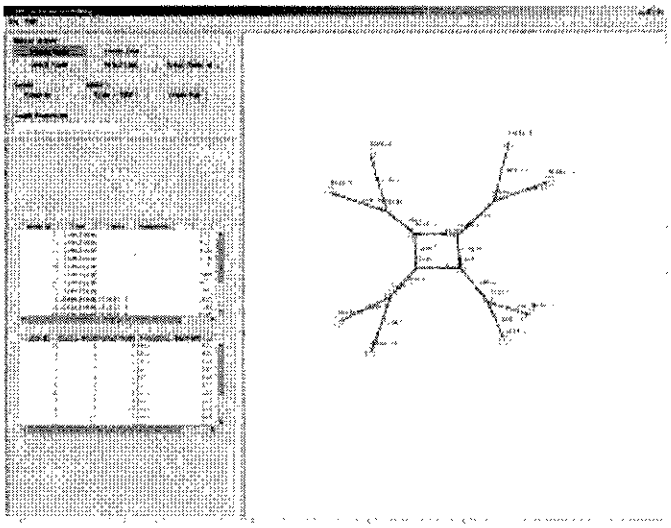

Figure 3. Screenshot

KOM ScenGen supports the automatic and manual setting of these parameters. For the manual setting a comfortable GUI is available (see figure 3). Algorithms to automatically identify and modify edge and core nodes and links are included in a library and allow an automatization of this step with a seript. Instead of specifying all node/link parameters for every node and link, also link and node types which all share the same parameters can be used.

Apart from setting the node and link properties, also the parameters for generating traffic output by the nodes ean be set in this step (application mix layer, see section 4.3.1).

We speeify an extended topology file format in [25] that is used by the scenario generator to store the topology, node & link properties and the traffic generation paramcters.

4.3 Load Generation

The general structure of the load generator is depicted in figure 4. Traffic models are used to generate traffic in edge nodes, the partner nodes for the traffic are selected depending on the sink model. Each traffic model models traffic of one kind (e.g. IP Telephony traffic, single WWW traffic or aggregated WWW traffic). A viewer to visualize the traffic and a test tool to test the traffic for self-similarity are useful in this step. We implemented a tool to estimate the Hurst parameter of the packet level traffic with a variance time plot. For a more detailed analysis SELFIS [26, 27] can be used as an independent package. We plan to better integrate SELFIS in a later version.

The generated network load can be exported to a simulator or testbed traffic emulator.

We now first discuss what traffic is and how it can be modeled. This allows us a very systematic and clear approach to traffic generation. We then explain our traffic generation module, the sink models and compare our approach with related work.

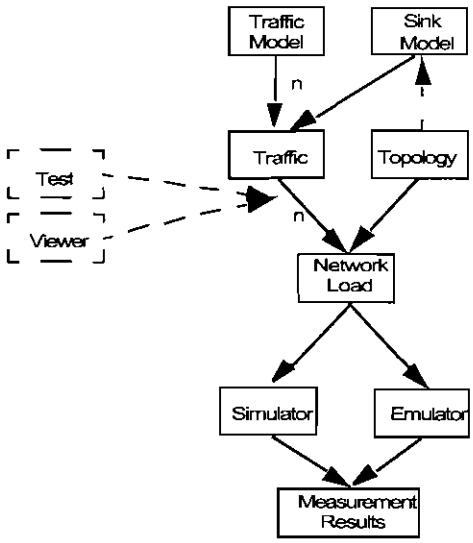

Figure 4. Load Generator Structure

4.3.1 Modeling Traffic. Traffic can be modeled on different layers with different degrees of abstraction. For ATM traffic we can distinguish between cell, burst and flow layer [28]. For IP traffic we think that the 5 layers of figure 5 are appropriate.

On the lowest layer IP traffic can be modeled as a series of **packets**. Each packet is specified by a generation time and size plus source and target node and port plus protocol number.

Traffic can also be modeled on higher more abstract layers. If traffic is aggregated in time we call this the **intensity layer** which specifies traffic as the number of bytes transmitted between a source and destination(s) or on one link in a single period of specified length. The information about the individual packet sizes is lost this way. It is non-trivial to split up an intensity into individual packets again. Traffic matrices are an example that typically use traffic intensities. Also some trace files specify traffic intensities and some self-similar traffic models specify how to generate traffic intensities.

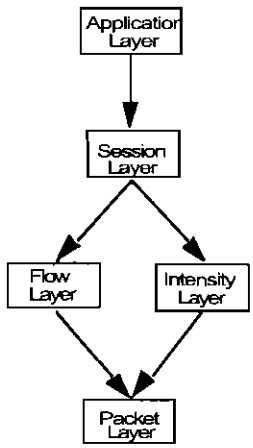

Figure 5. Traffic Layers

If traffic is not aggregated in time but instead by context we speak of the **flow layer**. Each flow generates a series of packets with a flow-type specific algorithm. A CBR flow transmits packets of fixed size in constant intervals. A greedy TCP Reno flow transmits packets as fast as possible using the TCP Reno flow and congestion control algorithm. The advantage of flow layer traffic is that it is obviously very powerful and memory efficient as a lot of packets can be described by a few flow parameters. However each flow type (CBR, greedy TCP, ...) has a very different set of parameters and the flow algorithm has to be implemented both in the simulator and traffic emulator.

All flows have a start time and a node/port pair. The greedy TCP source has the following additional parameters:

- Packet size

- Amount of data to be transferred
- TCP algorithm parameters

A CBR flow for example is charaeterized additionally by the following parameters:

- End time
- Packet size
- Interval between two paekets

The next highest layer is the **session layer**. A session consists of a number of elosely related flows or intensities. A simple IP telephony session for example might contain a number of CBR flows following each other with switching direetions. A session can be seen as the runtime instanee of one application.

The highest layer - the **application mix layer** - models how many sessions of which traffie model respective application are generated in one edge node (e.g. 40 IP Telephony, 20 Peer-to-Peer and 100 WWW sessions). The applieation mix is specified in the node & link property step and used as input for the load generator.

4.3.2 Traffic Generation. The modular and structured approach of our load generator allows it to easily develop and plug in traffie models. A traffie model instance generates sessions of one type (e.g. aggregated WWW traffic) consisting of flows, intensities or directly packets.

Currently the load generator contains the following traffic models:

- Single WWW model (modeling a single WWW user)
- Aggregated WWW model (modeling the aggregate of many WWW user's traffie) based on the traffic generator by Kramer [29].
- A simple IP telephony model
- A Peer-to-Peer model
- A model that allows traee-files to be played back. With this model we ean for example generate video conference or stored video streaming sessions.

It is work in progress to add further traffic models. Also, we do not want to reinvent the wheel and as there is much work about traffic modeling and many good tools and algorithms exist we aim for as many reuse as possible. Our architecture is open and allows to plug-in 3rd party tools and algorithms.

4.3.3 Sink Models. Generating paekets is not enough. Complex scenarios involve a larger number of nodes that can act as source and sink for traffic flows. As every session is generated in one node, this node acts as the source node for the session[1]. For most sessions a second node participates in the session, sometimes more than one (e.g. multicast sessions). An algorithm is neeessary to determine the partner nodc(s). We call this algorithm the sink model and eurrently investigate the influence of differcnt sink models.

Example: Our LETSQoS scenarios are from the point of view of a single ISP. In these scenarios we mark nodes as home user aeeess nodes (H), company aceess nodes (B), interconnection nodes (I) and core nodes (C). Core nodes are not the source and sink of traffic. Peer-to-Peer traffic uses a sink model that ehooses sink nodes from the set of H and I nodes modeling the fact that Peer-to-Peer traffic is mostly exchanged betwcen private end users. The WWW model on the other hand uses a sink model that connects B or I nodes with H or B or I nodes modeling the fact that most WWW servers are connected to company access nodes and not home uscr aeeess nodes.

We are currently investigating whether this distinction creates more realistic results than a purely random selection of partner nodes.

Support for multieast can be added easily also for existing traffic models with the appropriate multicast sink model. This is a strength that comes from the explicit separation betwcen traffic and sink models.

4.3.4 Related Work. As mentioned before a lot of traffic generators, simulators, emulators

[1] This does not mean it is also the source node of all flows belonging to that session, but the exact decision about the direction of the flows belonging to a single session is given by the traffic model.

and traffic models exist. However, we are not aware of any tool that generates traffie for both testbed experiments and simulations simultaneously as our tool does. We are also convinced that our approaeh of distinguishing between the different abstractions layers of traffic and the separation of traffic and sink models is a strong methodological improvement.

Using our terminology the eombination of a traffic generator and emulator for testbed experiments are very eommon. Commereial solutions like Chariot [30] and Ixia [31] inelude a number of traffic models. The Java based traffic emulator GenSyn [32] inspired our work. GenSyn models individual user's behavior with state machines for different applications (Web, FTP, MPEG, VoIP). Opposite to our approaeh, traffie is generated online and the feedbaek of the network can influence the traffic generation (if the throughput is too low a HTTP session might end earlier because the user gives up).

The scalable URL reference generator SURGE [33] specializes on aggregated WWW traffie as does [29]. Another project from our lab [34] focuses on generating realistic VoIP flows with control flows for testbed experimcnts.

The netperf benchmark tool [35] and NetSpee [36] are also often used to generate test traffic.

The combination of some fixed traffic models, a traffic generator and an export module for NS2 is quite common. NS2 itself contains several traffie models that can be easily used, e.g. with the NS2 seenario generator [14].

[37] provides detailed support for persistent and pipelined HTTP 1.1 connections and a SURGE-like load model implementation. RAMP [38] can convert measurements from a tcpdump-format file into eumulative distribution functions for simulation models which can then be used to generate realistie synthetic traffie in NS2. The pre-WWW tcplib model [39] can also be used for NS2 simulations.

fft_fgn [40] and RMD_nn [41] can be used to generate self-similar traffic on intensity layer. The algorithms of both tools are also integrated into the KOM load generator.

Apart from the traffic generators mentioned above that use one or more traffic models there is an enormous amount of work about traffie models. We ean only diseuss a small amount of those works here. [42] eontains a Telnet, FTP and SMTP/NNTP model, [43] eoneentrates on detailed models for WWW traffic and [44] can be used for FTP traffie models. For a very detailed single WWW user model with packet level details [45] is very useful. [46] contains an exeellent literature overview and specific information about TCP based traffic models. For ISP level simulations [47] ean also be handy, it contains a BGP traffic model and describes a tool for ereating realistie routing tables for testbeds.

Instead of using explieit models traffic is also often generated from trace files although one should be careful to use trace files in an environment different from the one they were recorded in [7]. Tracefiles are available e.g. at [48, 49, 50, 51] and can be played back with KOM SeenGen.

4.4 Plausibility Check

After generating the network load plausibility ehecks ean be started to control whether certain scenario aspects are sensible. An example would be comparing the bandwidth of the links with the bandwidth needed by the traffic. KOM SeenGen can estimate the bandwidth requirements of the generated network load. For the TCP connections the TCP formula [52] is used to predict the rate. The estimated bandwidth requirement can be compared with the offered bandwidth. If there is a large mismatch one might want to adapt the bandwidth of some links or the traffic before conducting the experiment.

4.5 Export

If the seenario setup passes the plausibility ehcck it can finally be exported. Currently two export modules exist, one for NS2 and one for our testbed. Export modules can be easily adapted to support other simulators or testbeds.

4.5.1 NS2. The NS2 export module can automatically create an OTel file for NS2 called *run.tcl* that sets up the topology and the traffie sources and starts them. To allow the user to finetune the setup process for her needs we do not directly eonfigure NS2 in the *run.tcl* script but instead call setup funetions that are defined in a seeond OTcl file *header.tcl*. Usually, the operator only has to adapt the *header.tcl* to her specific scenario's needs while the

run.tcl file can be generated automatically and does not have to be changed. In our LETSQoS scenario for example we have different *header.tcl* files for scenarios with IntServ, where e.g. RSVP has to be set up, DiffServ, where e.g. the PHBs have to be defined, and best effort. Each seenario dependent *header.tcl* file has to implement a fixed set of functions (e.g. "create-node"). For more details see [25].

4.5.2 Testbed. The export module of the scenario generator is written for our testbed in the LETSQoS project. It should not be too difficult to adapt it to other lab testbeds.

4.5.2.1 Description of our Testbed. The heart of our lab testbed are 16 PCs. Each is equipped with a Intel Pentium 850 Mhz processor, 256 MB RAM, a 20GB hard disk and 4 network interface cards. Further there are 3 24-port Allied Telesyn AT-8326GB switches which are staeked. We chose FreeBSD 4.6 as operating system since it has proven itself as a reliable operating system for our former testbeds. Administrating the testbed is always tedious as operations have to be performed on 16 machines. Therefore we wrote scripts that automate many tasks, e.g. we can eompletely install FreeBSD plus all needed applications automatieally.

For larger experiments we ean eonnect our old 8 machine testbed to the new one which leaves us with 24 test machines. The clocks of all testmachines are synchronized by a GPS receiver. This e.g. allows to do one-way delay measurements. The time stamps necessary for these kind of measurements are added by a Kernel module developed by Martin Karsten.

As control maehine and gateway to the external world (and the Internet) we use a separate PC that also runs a DNS and DHCP server for the testbed.

4.5.2.2 Automatic Configuration of the Testbed. The export module of the scenario generator creates a number of eonfiguration files and seripts. When the masterscript is started it sets up the testbed completely automatic. When a seeond script is started the experiment is started automatieally.

First SSH host keys on the machines are exchanged. Next the DNS and DHCP server on the control maehine are configured and restarted, then all maehines in the testbed are rebooted. The IP addresses of their interfaees are distributed by the DHCP server, the DNS server allows us to dress the maehines with the same names as in the scenario file.

Next the switch is configured automatically using an "expect" script addressing its telnet interfaee. Alternatively SNMP could be used[1]. The VLANs are set up to represent the links of the topology. Unused network interfaces are put into dummy VLANS. Because VLAN headers will be added to every packet we had to modify the Ethernet network drivers because otherwise full-size ethernet paekets eould not be sent.

We use a shortest path algorithm to ealeulate the routes and set up statie routing in all nodes.

After that ALTQ [53] eonfiguration files are distributed to all nodes and ALTQ is started. ALTQ is a traffic management software that enables certain QoS meehanisms on PC-based routers.

Further we plan to ineorporatc a modified version of NIST Net [52] to emulate a wide variety of network conditions and dummynet [53] to apply bandwidth and queue size limitations and emulate delays and losses.

Then the configuration files for our traffic emulator tool are distributed to all nodes (see next section).

Finally, a scenario dependent configuration seript can be exeeuted. Depending on the scenario KOM RSVP [54] is started on each machine or ALTQ is configured for DiffServ etc. The scenario dependent seript has to be written by the researcher himself.

We also have a video available showing the eonfiguration of the testbed at http://www.kom.e-technik.tu-darmstadt.de/letsqos/scengen/.

4.5.2.3 Traffic Emulator. After experimenting with some open souree tools that can emulate traffic on an ethernet interface we decided to develop our own tool. We had some prob-

[1] We experienced severe problems with SNMP and our switch.

lems with the timing of other tools. On FreeBSD, netperf [35] for example does not have a fine grained timer resolution. Netperf will send 128 paekets per second for a CBR UDP Flow with a paeket size of 80 bytes if the interarrival time is set to 8ms, 10ms, 12ms or 15ms and 64 if it is set to 16ms.

Our traffic cmulation tool has a more finegrained resolution and will really send 125 packets for an interarrival time of 8ms and 100 for one of 10ms. This tool was originally written by Martin Karsten and uses the efficient timer library of the KOM RSVP engine [54] (the exeellent timer management is one of the reasons why the KOM RSVP engine performs so well).

The traffic emulator runs on the sender and receiver side and can send diverse TCP and UDP flows. Information about the received paekets (e.g. the eurrent rate) is recorded and ean be written to an evaluation file after the experiment (file aecess during the experiment can disturb the timing of the network operations).

Because all clocks are synchronized by a GPS receiver the traffic emulators on all machines can start sending at the same point in time.

4.6 Touchup

Sometimes, not all possible steps and measurements ean be foreseen and therefore automated. Although it is the explicit goal of KOM ScenGen to avoid manual intervention as much as possible it might sometimes be neeessary to take a manual touchup step before the simulation/experiment in which the researeher checks, finetunes and possibly modifics parts of the scenario file. Note that for all our experiments with KOM ScenGen no touchup activities were necessary.

4.7 Simulation or Testbed Experiment

Finally, the simulation or the testbed experiment can be condueted by running NS2 with the generated OTcl file or by running the start script on the testbed control machine.

4.8 Evaluation

The last step is analyzing and evaluating the results of the simulation. Several already existing tools can be used for this step. We use Gnuplot [55] and Microsoft Exeel for evaluation purposes. For demonstration purposes the network animator NAM [11] can be used. For future work we plan to support the automatic statistical analysis of the measured data.

4.9 Implementation of KOM SeenGen

KOM ScenGen is implemented in Java, the NS2 parts are written in OTcl and C++, the testbed export code in Python. The traffic emulator software was originally written by Martin Karsten as part of the KOM RSVP engine [54].

5 Summary and Conclusions

In this paper we presented a systematic approach to simulation and testbed experiments and the KOM scenario generator. We discussed the different steps in generating a network researeh scenario. They are all supported by KOM ScenGen which contains many helpful tools like a topology file format converter, an applieation oriented and topology awarc traffic generator and seripts to automatieally configure a lab testbed. Apart from this it supports simulating and emulating (in a lab testbed) the created scenario. The combination of simulation and testbed experiments avoids most of the drawbacks and pitfalls of those methods if used alone. As a methodological improvement to traffic and load generation KOM ScenGen uses different abstraction layers for traffic and the separation between traffie and sink models.

The first version of thc scenario generator is finished and already being used for QoS experiments in the LETSQoS (www.letsqos.de) project. More information about the sccnario generator and a video demonstrating the scenario generator at work are available at www.kom.tu-darmstadt.de/letsqos/scengen/.

Acknowledgments

This work is partly sponsored by the German research network providcr DFN (www.dfn.de). We would like to thank the DFN for their funding, deep insights and valu-

able operational data. Martin Karsten helped us in many aspects, his valuable tools are used in many parts of KOM ScenGen. We would further like to thank the students Ian Hubbertz and Martin Jess for their work on the scenario generator and our testbed, Tobias Boll for his work on the scenario generator and NS2 and last but not least Peter Larem and Thomas Pfeiffer for their work on the scenario generator.

References

[1] R.Braden, D.Clark, and S.Shenker. Integrated Services in the Internet Architecture: an Overview. Informational RFC 1633, June 1994.
[2] D.Black, S.Blake, M.Carlson, E.Davies, Z.Wang, and W.Weiss. An Architecture for Differentiated Services. Informational RFC 2475, December 1998.
[3] P.Hurley, M.Kara, J.Y. Le Boudec, and P.Thiran. ABE: Providing a Low-Delay Service within Best Effort. *IEEE Network Magazine*, 15(3), May 2001.
[4] F.Kelly. Models for a self-managed Internet. *Philosophical Transactions of the Royal Society*, A358:2335–2348, 2000.
[5] M.Karsten and J.Schmitt. Admission Control based on Packet Marking and Feedback Signalling. Mechanisms, Implementation and Experiments. Technical Report TR-KOM-2002-03, Darmstadt University of Technology, May 2002.
[6] M.Karsten, J.Sehmitt, and R.Steinmetz. Implementation and Evaluation of the KOM RSVP Engine. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2001)*, pages 1290–1299. IEEE, April 2001.
[7] S.Floyd and V.Paxson. Difficulties in Simulating the Internet. *Transactions on Networking*, pages 392–403, Feb 2001.
[8] 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*, 33(5):59–67, May 2000.
[9] A.Hafid, J.D. Meer, A.Rennoch, G.V. Bochmann, and R.Dssouli. Quality of Service Verification Experiments. In *Proceedings of the Workshop on Distributed Multimedia Applications*, 1994.
[10] M.Neufeld, A.Jain, and D.Grunwald. NSCliek: Bridging Network Simulation and Deployment. In *Proceedings of the 5th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 74–81. ACM Press, 2002.
[11] Network Simulator NS2. http://www.isi.edu/nsnam/ns/.
[12] JavaSim Network Simulator. http://www.javasim.org/.
[13] OpNet Network Simulator. http://www.opnet.com/.
[14] NS2 Scenario Generator. http://www.isi.edu/nsnam/dist/scen-gen.tar.
[15] NS2 Scenario Generator Modifications for QoS Experiments. http://keskus.hut.fi/tutkimus/iro-net/ns2/ns2.html.
[16] MANET Scenario Generator. http://www.comp.nus.edu.sg/liqm/scengen/.
[17] BonnMotion: Java Mobility Scenario Generator and Analyser. http://www.cs.uni-bonn.de/IV/BonnMotion.
[18] E.Kohler, R.Morris, B.Chen, J.Jannotti, and M.F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.
[19] D.Herrscher, A.Leonhardi, and K.Rothermel. Modeling computer networks for emulation. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, pages 1725–1731, June 2002.
[20] TIERS. Tiers Topology Generator. http://www.isi.edu/nsnam/ns/ns-topogen.html#tiers.
[21] BRITE. Boston University Representative Internet Topology Generator. http://www.cs.bu.edu/brite/.
[22] GT-ITM. Georgia Tech Internetwork Topology Models. http://www.cc.gatech.edu/projects/gtitm/.
[23] Inet Topology Generator. http://topology.eecs.umich.edu/inet/.
[24] O.Heckmann, M.Piringer, J.Schmitt, and R.Steinmetz. On realistic network topologies for simulation. In *Proceedings of ACM SIGCOMM MoMeTools*, Karlsruhe, 2003.
[25] O.Heckmann, K.Pandit, J.Schmitt, M.Hoffmann, and M.Jobmann. LETSQoS Milestone 2. http://www.letsqos.de, June 2002.
[26] T.Karagiannis and M.Faloutsos. SELFIS: A Tool for Self-Similarity and Long-Range Dependence Analysis. In *Proceedings of the 1st Workshop on Fractals and Self-Similarity in Data Mining*, 2002.
[27] T.Karagiannis. SELFIS: A Short Tutorial. http://www.cs.ucr.edu/tkarag/Selfis/Selfis.html, 2002.
[28] J.Roberts, U.Mocci, and J.V. (Eds). *Broadband Network Teletraffic (Final Report of COST 242)*. Springer Verlag LNCS 1155, 1996.

[29] UC Davis Generator of Self-Similar Traffic. http://wwwcsif.cs.ucdavis.edu/kramer/code/trf_gen2.html.
[30] NetIQ Chariot Traffic Generator. http://www.netiq.com/products/chr/default.asp.
[31] Ixiacom Ixia Traffic Generator. http://www.ixiacom.com/.
[32] P.Heegaard. GenSyn - a generator of synthetic Internet traffic used in QoS experiments. In *Proceedings of 15th Nordic Teletraffic Seminar*, 2000.
[33] P.Barford and M.Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Measurement and Modeling of Computer Systems*, pages 151–160, 1998.
[34] KOM Call Generator. http://www.kom.tu-darmstadt.de/KOMtraffgen/.
[35] Netperf Network Benchmark Tool. http://www.netperf.org/netperf/NetperfPage.htm.
[36] A Tool for Network Experimentation and Measurement (Netspec). http://www.ittc.ukans.edu/netspec/.
[37] NSWEB HTTP Traffic Generator. http://www.net.uni-sb.de/jw/nsweb/.
[38] K.Lan and J.Heidemann. Rapid Model Parameterization from Traffic Measurements. http://www.isi.edu/kclan/paper/ramp.pdf.
[39] P.B. Danzig and S.Jamin. tcp-lib: A library of TCP/IP Traffic Characteristics. *USC Networking and Distributed Systems Laboratory TR CS-SYS-91-01*, October, 1991.
[40] C.Schuler. fft_fgn: fraetional gaussian noise generator. ftp://ita.ee.lbl.gov/software/fft_fgn_c-1.2.tar.Z.
[41] I.Norros, P.Mannersalo, and J.Wang. Simulation of fractional Brownian motion with conditionalized random midpoint displacement. *Advances in Performance Analysis*, 1999.
[42] V.Paxson and S.Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.
[43] G.Abdulla. *Analysis and Modelling of World Wide Web Traffic*. PhD thesis, Virginia Polytechnic Institute and State University, 1998.
[44] D.J. Ewing, R.S. Hall, and M.F. Schwartz. A Measurement Study of Internet File Transfer Traffic. Technical Report CU-CS 571-92, January, 1992.
[45] J.Charzinski. HTTP/TCP Connection and Flow Characteristics. *Performance Evaluation*, 42(2-3):149–162, Sep. 2000.
[46] A.Feldmann. Characteristics of TCP Connection Arrivals, 1998. Technical report, AT&T Labs Research, 1998.
[47] O.Maennel and A.Feldmann. Realistic BGP Traffic for Test Labs. In *Proceedings of ACM SIGCOMM*, 2002.
[48] O.Rose. Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems. Technical Report Technical Report No. 101, University of Wuerzburg, Institute of Computer Science, 2 1995.
[49] Internet Traffic Archive (ITA). http://ita.ee.lbl.gov/html/traces.html.
[50] NLANR/NZIX Traces. http://pma.nlanr.net/Traces/.
[51] Waikato Internet Traffic Storage Traces. http://wand.cs.waikato.ac.nz/wand/wits/.
[52] J.Padhye, V.Firoiu, D.Towsley, and J.Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of the ACM SIGCOMM*, 1998.
[53] K.Cho. The Design and Implementation of the AltQ Traffic Management System. PhD thesis, Keio University, January 2001.
[54] KOM RSVP Engine. http://www.kom.tu-darmstadt.de/rsvp/.
[55] Gnuplot. http://www.gnuplot.info/.
[56] K. Fall. Network Emulation in the Vint/NS Simulator. In *Proceedings of the 4th IEEE Symposium on Computers and Communications*, 1999.