

## GenSyn - a generator of synthetic Internet traffic used in QoS experiments

Poul E. Heegaard  
Telenor R&D, N-7004 Trondheim, Norway\*  
phone: +47 7354 3845  
fax: + 47 7354 3700  
e-mail: Poul.Heegaard@telenor.com

### Abstract

*For Quality of Service (QoS) testing of new applications and network mechanisms in the Internet, a generator of controllable, scalable, synthetic but realistic IP traffic is required. A traffic generator called GenSyn has been developed for this purpose. GenSyn has defined a flexible and scalable modelling framework. The stochastic user behaviour is described by state diagrams. The model is scalable because it allows a composition of users in each state, instead of creating a new instance of the process for every user. The stochastic user behaviour model controls the creation of TCP connections and UDP streams through interface modules that links the GenSyn process to the underlying Internet protocol stack on the workstation. This means that on transitions between specific states in the stochastic model, an interface process will be instantiated and IP packets are sent and received through the network.*

*To demonstrate the applicability of GenSyn this paper shows an example of a scenario created for testing QoS in an experimental test net. The test net is an IP based communication platform that will provide differentiated services. To test its QoS mechanisms, a controllable, and reproducible, mixture of traffic streams with different characteristics is essential. This traffic mixture is generated by GenSyn using the source models that are currently available and described by the framework of GenSyn. This includes models of web and ftp clients that are generating tcp traffic, and models that generate udp traffic from a video server (using MPEG), from voice over IP (VoIP), and in a Constant Bit Rate (CBR) stream.*

*The scenario example includes description of test objectives, QoS requirements, measurement method, traffic load profiles, traffic mixtures, and test topology. The scenario is offered to an IP test network both with (DiffServ) and without (pure best effort) QoS mechanisms. The work is an ongoing activity but preliminary results will be presented.*

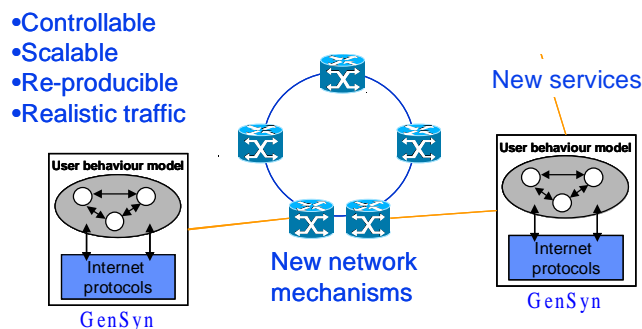
**Keywords:** traffic generator, user behaviour, real Internet traffic, TCP, UDP, IP, workload benchmark, web, MPEG, Java, QoS testing.

---

\* Phase one of the GenSyn development was carried out as an employee in SINTEF Telecom & Informatics.

## 1 Introduction

For the purpose of Quality of Service testing of new applications and network mechanisms in the Internet, a generator of controllable, re-producible, scalable, synthetic but realistic traffic is required. The generator will typically produce traffic in a controlled testbed environment where there are few real users and a corresponding low traffic load. Realistic, controllable, and re-producible traffic load is of great importance in order to test the Quality of Service of new applications, i.e. interactive video. In addition, to increase the understanding and get experience with the configuration of new QoS network mechanisms, the network must be offered a high load of heterogeneous traffic mixture.



**Figure 1 Load generation in an Internet test environment**

Different source modelling approaches can be considered to describe a typical Internet source:

1. *Trace* - replay of a recorded stream of IP packets, i.e. a stream of IP packets obtained from measurements is replayed and offered to the test network (e.g. replay of a tcpdump-file). If the recorded stream contains traffic from elastic sources (e.g. TCP connections) the replay will not be representative unless the congestion situation in the network is exactly the same. This will rarely be the case in a network with a mixture of traffic streams.
2. *Aggregate generator* (“black box”) - generation of IP packets according to a parametric stochastic process. If a recorded aggregate of packets from elastic sources is used to determine the parameters of the model, the same problem as described in *Trace* above will still be present.
3. *User behaviour model* (“white box”) - generation of IP packets from physically based source models. More detailed measurements than for the two other approaches are required. However, the parameters in the model reflect the user behaviour and hence it is straight forward to change the model if e.g. the number of sources will be changed.

For generation of traffic in Internet testbeds both user behaviour models, see e.g. [MTW99], [ChLi99], [Vic98], [Ake99], and bulk-transfers, e.g. ttcp [MTW99], are being used. GenSyn combines the user behaviour approach with bulk-transfer of data via communication streams. Application of stochastic user behaviour models described by state diagrams introduces flexibility, scalability, physically interpretable model parameters. This part of the modelling framework is adapted from the ideas in [Helv95] which were used in an ATM traffic generator called ATM100 or Synthesised Traffic Generator (STG) [HMM93]. However, instead of

developing a specialized hardware instrument, GenSyn applies modern Web- and Java-technology and exploits the Internet protocol suite (TCP/IP) that is already available. This means that the generator is only a software process that imitates the user behaviour and dynamically controls the creation and deletion of one or more links (threads) to physical HTTP- and TCP-connections. The generator is not only a simulator, it generates real IP packets that flow through a real (test) Internet.

In this paper the modelling framework of GenSyn is described. The general framework is given in Section 2, while Section 3 contains an overall description of a QoS performance test in a IP based communication platform. This is included to demonstrate the use of GenSyn. Finally, the paper is closed in Section 4 with some general comments of application of GenSyn. For more details about the application models available in GenSyn, see [Heeg00].

## 2 Modelling framework

### 2.1 Multi-level stochastic behaviour

The model described using the GenSyn framework will attempt to reproduce the inner workings of the physically source. This includes many stochastic processes, both human, environmental, and communication equipment. An example of the variety of activity levels in a source is illustrated in Figure 2. The example is originally from ATM, but this general multi-level behaviour is independent of the communication technology. Consider for instance a telephony user. The user is present (at the office) {end-user present level}, he is making a phone call {connection level}, during the call he is speaking and listening {dialogue level}, when he is speaking he sends voice and take short breaks {burst level}, when voice is sent it is wrapped in packets {packet level}, each packet is segmented in cells {cell level}. The last two levels are technology dependent. The typical time constants involved on various levels are indicated in the table in Figure 2. Hence, the aggregated packet or cell stream that can be observed on the transport or cell level will have a communication pattern that is generated as a result of many interacting stochastic processes with different time constants. In [HeHo95] and [WTSW97] this multilevel superposition of stochastic processes with different time constants is considered to be an explanation of the observed self-similar behaviour observed in aggregated traffic streams (packet or cell level) on the Internet [LTWW94].

The aggregated stream will be a heterogeneous mixture of traffic from various sources where each source will be influenced by both user behaviour, equipment and protocols, see Figure 2.

#### *User behaviour:*

- the end user behaviour - setup/disconnect a session, application mixture (web browsing, software downloads, chat, games, email, streaming (video, audio))
- user-network interaction (slow variation) - interest/impatience, takes a break and returns later due to congestion, cost, etc.
- variation in information stream - variable video coding (MPEG).

#### *Equipment and protocols:*

- end user equipment constraints - access capacity, processor capacity, disk, video coding processing
- communication system constraints - buffer space, transmission capacity, router capacity

- network mechanisms - routing strategies, priorities (diffserv), weighted fair queuing, resource reservations (RSVP, intserv)
- protocols, e.g. TCP congestion control and avoidance.

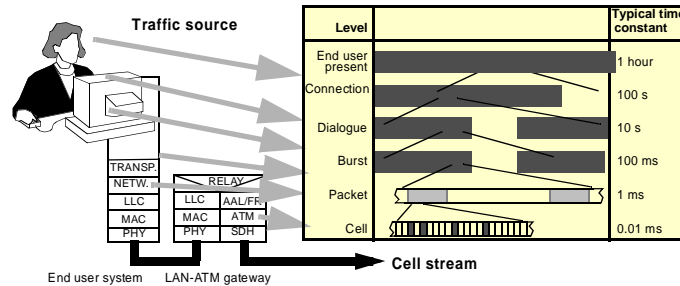


Figure 2 Illustration of different activity levels in a source

## 2.2 Linking stochastic user behaviour models to real communication streams

GenSyn models the user behaviour in a state based source model, while the communication systems are not modelled. The equipment constraints and protocol behaviour are automatically included through the linking of the stochastic processes to the built-in protocol stack on the workstation. This means that no incorrect assumptions about the protocols or network mechanisms will be made, it is for instance not necessary to know the details about the MPEG coding or the TCP slow start mechanisms.

In general it can be said that the modelling framework combines the better of two worlds, it gets the flexibility and scalability of state diagram description with composition of users combined with the accuracy of protocol and network behaviour by using the actual protocol instead of a model.

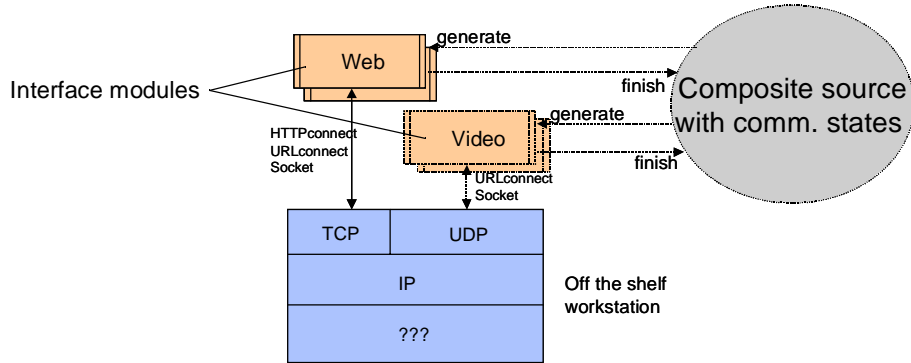
### 2.2.1 Division of state space

Figure 3 shows a principal sketch of the modelling framework to illustrate the linking of a composite state space description and the protocol stack. This efficient combination is accomplished by dividing the modelling state space into:

- *stochastic state space*,  $\Omega_S$ , where the stochastic user behaviour is described by a state machine where each state can contain a collection (composition) of many users\*, and
- *communication state space*,  $\Omega_C$ , which is a “placeholder” for the users that are waiting for response from the communication system.

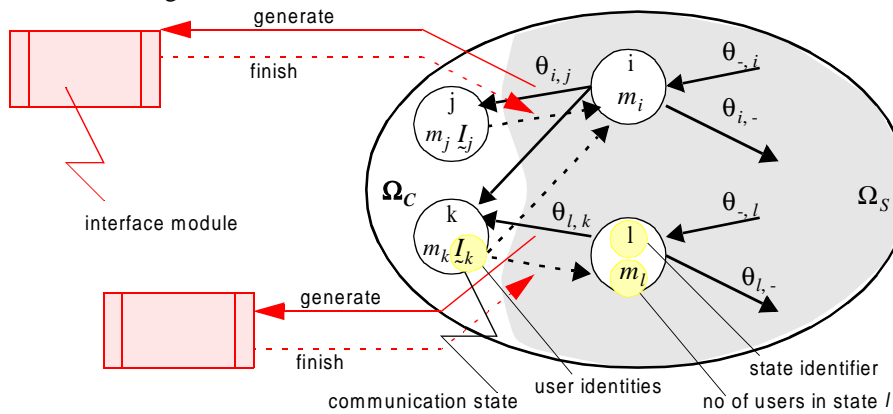
All transitions from the stochastic to the communication state space,  $\Omega_S \rightarrow \Omega_C$ , will instantiate an *interface module* that creates a communication stream to and/or from the workstation. A communication stream is either a TCP connection or a UDP datagram stream. The user that instantiated the communication stream will be placed in a *communication state* and stay there until the communication is completed. Hence, a transition from the communication state space back to the stochastic state space,  $\Omega_C \rightarrow \Omega_S$ , is initiated on completion of communication and the corresponding pointer to the interface module is removed.

\* The state model has to be (semi-)Markovian, i.e. each state sojourn time must be negative exponentially distributed, in order to make the composition of users in each state.



**Figure 3 Interface modules - linking user behaviour to built-in protocol stack**

The number of states will not change during the evolution of the generator process, only the number of users in each state will change. All states need an attribute for the current number of users in each state, and the communication states need an additional attribute for the storage of information about the user (or process) identities that awaits a communication stream to complete. Hence, only a modest increase in the generator process is observed as the number of users increases. This solution is chosen to optimise the scalability against the accuracy in the protocol modelling.



**Figure 4 The modelling framework: division of state space**

### 2.2.2 Notation

The following notation will be used in this paper:

$\Omega$  - global state space,  $\Omega = \Omega_S \cup \Omega_C$

$\Omega_S$  - stochastic state space

$\Omega_C$  - communication state space

$\underline{m}$  - state vector,  $\underline{m} = \{m_i\}_{i \in \Omega}$

$m_i$  - number of users in state  $i$

$I_i$  - identity vector of users with an open communication stream in state  $i \in \Omega_C$

$\theta_{i,j}$  - state transition rate from state  $i$  to state  $j$ ,  $i \in \Omega_S$

$\theta_i$  - total transition rate in state  $i$ ,  $i \in \Omega_S$ ,  $\theta_i = \sum_{j \in \Omega} \theta_{i,j}$

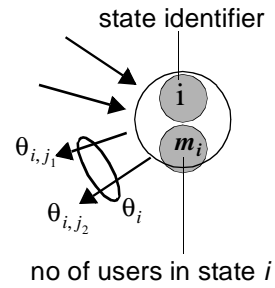
$E(T_i)$  - expected sojourn time  $E(T_i) = 1/(\theta_i m_i)$ ,  $i \in \Omega_S$

In Figure 4, the notation is shown in an example demonstrating the division of state space and the links to the interface modules.

### 2.2.3 Stochastic state model - the behaviour of a single source

The user behaviour is described by a finite state continuous time Markov process. A general state has the following attributes:

- a state identifier,  $i$ ,
- number of users  $m_i$  in state  $i$ ,
- a state sojourn time distribution (negative exponential distribution),
- list of transition rates,  $\theta_{i,j}$ , and probabilities,  $p_{i,j} = \theta_{i,j}/\theta_i$
- list of neighbour states, i.e. states that can be reached in one transition from state  $i$ .



A state model is semi-Markovian if all states have state sojourn times that are negative exponential distributed, or if neither of the states in the model have more than one non-exponential sojourn time. In case of a state with non-exponential time distribution, an approximation by a phase-type distribution is feasible by substituting this state with a combination of states that have negative exponential time distributions. This means it is possible to model state sojourn times that follows a hyper-exponential, hypo-exponential, or Coxian distribution. All these distributions is a combination of states with negative exponentially distributed state sojourn times.

When all state sojourn times are exponential, the procedure described in Figure 5 can be applied to determine the next stochastic event. Observe, however, if a communication stream is completed while waiting for the next scheduled stochastic event to occur, a state change caused by this completion, see Figure 6.

1. Sample the time  $T$  to next event in  $\Omega_S$ , the expected value is  $E(T) = 1/(\sum_{j \in \Omega_S} \theta_j m_j)$
2. Wait  $T$
3. Sample which state  $i \in \Omega_S$  where the next event took place, the probability is  $\theta_i m_i / \sum_{j \in \Omega_S} \theta_j m_j$
4. Sample the next state  $j$  from state  $i$ , the probability is  $p_{i,j} = \theta_{i,j} / \theta_i$
5. Move a user from state  $i$  to state  $j$  by updating the  $m_i + 1$  and  $m_j - 1$ .

**Figure 5 Update the state vector when the next event is a stochastic event in  $\Omega_S$**

### 2.2.4 Communication State model - link to the network

The communication states are the “placeholders” for all users that currently have an open communication stream. The states sojourn times,  $T_i$ ,  $i \in \Omega_C$ , for all users in the communication states are fully determined by the behaviour of the underlying communication system, i.e. the

performance of the end user equipment, protocols, network mechanisms. A user will be “locked” in the communication state as long as the communication stream is open, and immediately be removed when the stream is closed. Hence, for user  $x$  the transition from state  $i$  in  $\Omega_C$  to state  $j$  in  $\Omega_S$  is considered to be a *conditional transition*, i.e.

$$\theta_{i,j}(x) = \begin{cases} \infty & \text{comm. stream opened by } x \text{ is closed} \\ 0 & \text{comm. stream opened by } x \text{ is open} \end{cases} \quad (1)$$

where state  $i \in \Omega_C$  and state  $j \in \Omega_S$ .

In the communication states a relation, e.g. a process identity, to all opened communication streams must exist. When a communication stream is opened and a new user enters state  $i$ ,  $i \in \Omega_C$ , the process identity of the interface module related to this state transition need to be stored. For this purpose an identity vector  $I_i$  is added as a new attribute to the communication states in addition to the list in Section 2.2.3. When a communication stream is closed, e.g. when a file is downloaded, an instantaneous state transition will occur, and a user leaves this communication state ( $m_i - 1$ ). Observe that the number of user in state  $i$  equals the number of elements in the identity vector in state  $i$ ,  $m_i = |I_i|$ . The identity vector,  $I_i$ , is implemented as a list of pointers (process identities) to open communication streams.

In Figure 6 the addition to Figure 5 is given to handle both stochastic events and events triggered upon completion of a communication stream.

1. Sample the time  $T$  to next event in  $\Omega_S$  as described in step 1 of Figure 5.
  2. If  $(\exists k \in \Omega_C) \wedge (T_k < T)$ , i.e. a communication stream is closed before the time  $T$  elapses, then replace step 3 in Figure 5 with the following:
  3. The next event takes place in state  $k$  in  $\Omega_C$  where  $k$  is where the first stream closed i.e.
$$k = \{i | i \in \Omega_C \wedge (T_i = \min_{j \in \Omega_C} (T_j))\}$$
- Step 4 and 5 is the same as in Figure 5 except that the  $p_{i,j}$  are given as parameters because they can not be derived from the conditional rate  $\theta_{i,j}(x)$  which is now either  $\infty$  or 0.

**Figure 6 Update the state vector when the next event is an event in  $\Omega_C$**

The state sojourn time in a communication state may depend on the current congestion situation in the underlying network. In the case of downloading web pages, the congestion, the size and location of the requested page will contribute to the sojourn time. Furthermore, for web- and ftp-downloads an *impatience factor* is defined for each communication state. This enables the definition of an upper limit on the time spent in the communication state. The impatience factor is formulated as an condition of  $\theta_{i,j}(x)$  in (1), and its randomness is defined in the stochastic part of the source model.

### 3 The use of GenSyn in QoS testing

To demonstrate how to use GenSyn in a test environment, this section describes a test of QoS end-to-end performance in an IP based communication platform with variable degree of support for QoS guarantees. GenSyn is currently being deployed in such a testbed with DiffServ and MPLS enabled routers and a variety of applications. Some preliminary results from these measurements will be presented at the seminar.

This section describes the scenario example, including test objectives, testbed topology, GenSyn applications, GenSyn deployment, load level and mixture, and the measurement approach.

### 3.1 Test objective

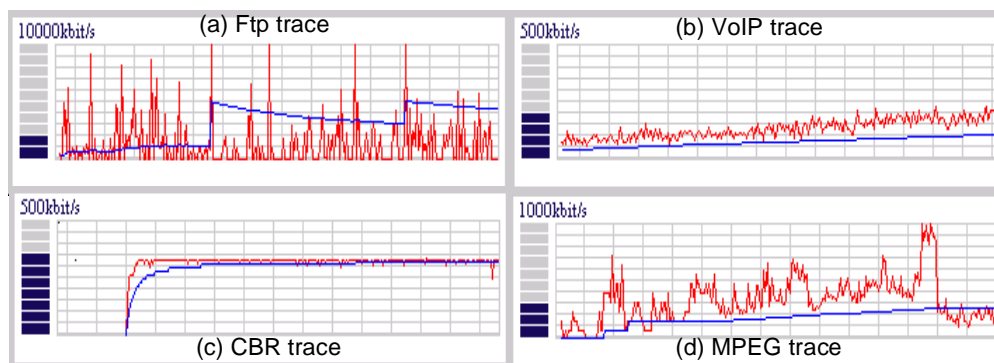
The objective of the experiment is to study the sensitivity on end-to-end performance of changing e.g. the tcp/udp mixture, increasing the total load, changing the traffic characteristics (burstiness), and changing the underlying mechanisms providing support for QoS differentiation.

### 3.2 The GenSyn application models

A few source models are now available as described in the modelling framework of GenSyn. New models can be defined, and the current models can easily be changed if this is requested.

1. **TCP traffic** - adjusts to the network performance (slow-start window mechanism)
  - i. *Web* - a model of users (clients) that are downloading web-pages with all their content (inclusive applets and images) from real web servers all over the world. The url-addresses are found in a parameter list of predefined addresses that may dynamically be updated as the experiment evolves.
  - ii. *ftp* - a model of users (clients) that download real files from a server. The files are specified in a parameter list of files.
2. **UDP traffic** - no network performance adaptation at transport level
  - i. *VoIP* - a model of the information/media stream from VoIP users. It sends a deterministic stream of packets (fixed size and inter packet arrival time) from each of the active users. The model does not include the call setup and disconnection phases.
  - ii. *MPEG* - a model of a video server that is sending MPEG-1 coded video sequences [Rose95]. All video frames are converted to a number of fixed sized IP packets sent back to back. The interframe distance is a parameter with a default value as recommended by the MPEG-1 codex standard. The clients are implicitly modelled only as incoming requests.
  - iii. *CBR* - a model of a multiplex of deterministic streams of packets with phase shifts.

In Figure 7 snapshots from GenSyn visualization module are included. The figure illustrates that GenSyn can generate packet flows with very different traffic characteristics.



**Figure 7 Snapshot of 4 traces captured from the visualizer in GenSyn**

### 3.3 QoS mechanisms and service differentiation

It is a trend towards building communication platforms for integration of a great variety of applications with different traffic characteristics and users with different Quality of Service. The next generation network will be Full Service Networks - one network for all type of services. The various services and applications need to be treated differently, and hence some means for service differentiation is required with different support for traffic management and control. In an IP based network such differentiation and management techniques are still a research topic. The proposed, and implemented, mechanisms and methods (e.g. DiffServ) need to be studied carefully.

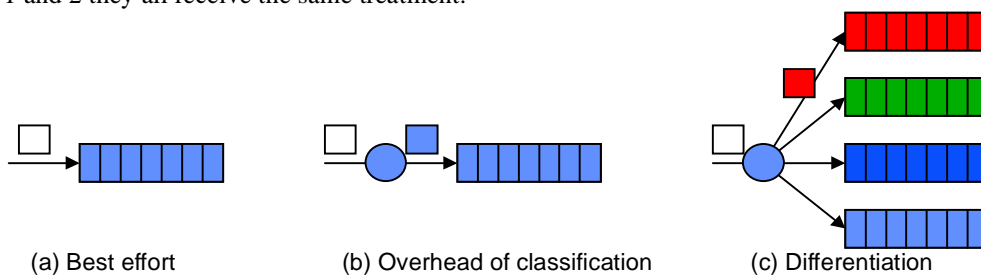
In the test scenario described in this paper, a differentiation of services is done according to

1. *Real time requirements* - no (best effort), weak (audio/video streaming), and hard (telephony, interactive video)
2. *Willingness to pay* - economy (free/cheap, only connectivity requirements), business (inexpensive, minimum guaranteed level on QoS requirements) and first class (expensive, hard QoS requirements)

The service differentiation is part of a Service Level Agreement (SLA) that will exist between end-users, network providers, service providers, and content providers. In order to provide (and observe) different QoS performance, some mechanisms and methods are required in the network. To test the implemented mechanisms of commercial routers it is proposed to carry out the same experiment under 3 different network configurations, see Figure 8.

1. *Best effort* - to establish a reference system
2. *Classification without differentiation* (best effort) - to study overhead of the mechanism
3. *Classification with differentiation* in service classes - to study effect of differentiation

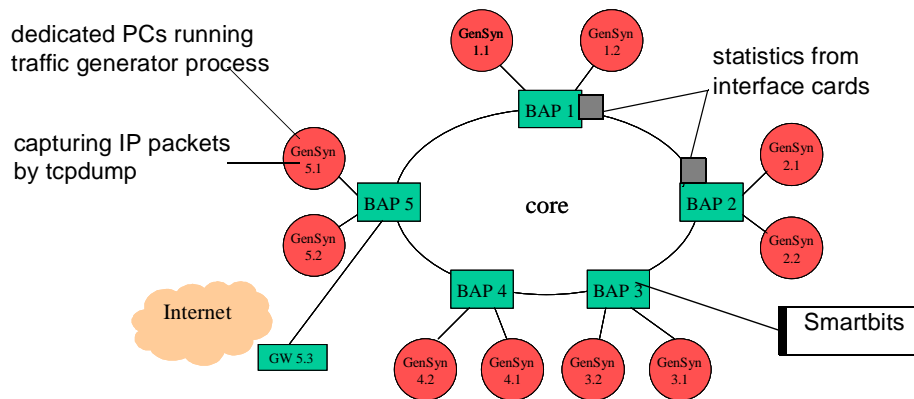
In all 3 cases the QoS performance is studied separately for all service classes although in case 1 and 2 they all receive the same treatment.



**Figure 8 Support for QoS in test**

### 3.4 Test topology

In the figure below a principal sketch of the topology of the test network is given. The test network is an IP based communication platform that will provide differentiated services. GenSyn is running on 10 dedicated standard PCs with Linux OS, 2 PCs connected to each Broadband Access Point (BAP). Other measurement and load generation equipment like Smartbits [Sbit] can be combined with GenSyn.



**Figure 9** Equipment for testing end-to-end QoS performance

### 3.5 GenSyn model deployment

Before the GenSyn processes can be distributed on the PCs and started, it is necessary to decide where to run the various processes. Furthermore, it has to be specified what number of users to run for each type of model to produce the requested traffic mixture and load.

#### 3.5.1 The tcp - udp mixture and load level

All traffic scenarios defines their load level  $\alpha$  as the total offered load from all GenSyn processes relative to the network capacity,  $C$ , and hence the total load is  $\rho = \alpha C$ . As a start, the following load levels will be defined:  $\alpha = 0.2, 0.4, 0.6, 0.8$ . As an option, other load levels will be defined. This depends on the outcome of the experiments.

Each load level has defined different mixtures of tcp and udp:

- *today* - in bytes, 85% tcp and 15% udp traffic
- *tomorrow* - in bytes, 40% tcp and 60% udp traffic
- *near future* - in bytes, 10% tcp and 90% udp traffic, under the assumption that applications like video on demand or interactive video become popular

The latter is expected to be a worse case with respect to tcp performance. The reason is that udp has no feedback mechanism that adjusts the bitrate according to the congestion in the network similar to what tcp does. This implies that under heavy load, tcp will reduce its sending window to a minimum, while udp traffic sources continue to send at the same rate. For some applications that are running on top of udp a rate adaptation or control are implemented, but this is not considered in this example.

The 3 different tcp / udp mixtures are constructed by the use of the application models that are available in GenSyn, see Section 3.2.

#### 3.5.2 Heuristics for specifying the number of users

To scale up the models to generate a specific load level the traffic generated by one single user must be determined, and the time spent in the send states of the state models describing the user

behaviour must be calculated. In the case of udp models, this is a trivial task because the calculations are based on model parameters. However, tcp behaviour depends on the network congestion:

1. the traffic generated by  $X$  users are not  $X$  times the traffic generated by one single user except in a very lightly loaded network
2. the time spent in a send state depends on the load on the server, the network congestion, etc.

This means that to come up with an exact mapping of the number of tcp users and an observed load level, a detailed control over the interaction of traffic flows and other running processes is required. In a real network this level of control is unrealistic, even in a relative small and closed test environment. Instead, it is proposed to use an approach based on conducting a few test experiments combined with a simple heuristics.

$\hat{t}$  - observed download time with only one user active in send state

$C_1$  - observed TCP throughput on download with only one user active in send state

$C$  - total link rate available

$N$  - number of users

$\pi_s(t)$  - probability of being in send state (dependent on  $t$ , the download time)

$\rho(x) = \min(C/x, C_1)$  - the throughput estimate per user with  $x$  users active

The following two approaches are considered to determine the expected total load for  $N$  users

1. *Independent scaling* - an upper bound where the tcp flow control and congestion mechanism are not considered. This approach is appropriate for scaling of udp models.

$$\alpha_1(N) = \max(N \cdot \pi_s(\hat{t}) \cdot \rho(1), C) \quad (2)$$

2. *Scaling considering simultaneous connections* - based on a ideal sharing of the available link rate (one objective of the tcp flow control and congestion mechanism) and measurement of average throughput and download time for a single user. The scaling assumes that the number of user in the send state follows a Binomial distribution.

$$\alpha_2(N) = \sum_{x=1}^N \binom{N}{x} \cdot \pi_s(\hat{t})^x \cdot (1 - \pi_s(\hat{t}))^{N-x} \cdot x \cdot \rho(x) \quad (3)$$

The number of tcp users for a specific load level  $\alpha$  is determined by  $\alpha_i(N) = \alpha$ , ( $i=1$  or  $2$ ). In Figure 10 a plot of the two approaches is illustrated where  $C = 100$  [Mbit/s],  $C_1 = 60$  [Mbit/s], and  $\hat{t} = 37$  [ms]. Observe that under light load ( $\alpha < 0.2C$ ) then the  $\alpha_1(N) \approx \alpha_2(N)$ .

### 3.6 Measurements

Very little measurement functionality are included in the GenSyn. Instead the measurements are based in external measurement processes like tcpdump (on all PCs, time synchronized, in the test net to monitor the end-to-end performance), counters at the interface cards (to monitor the routing and load balance), or specialized equipment like Smartbits [SBit]. At the seminar a few preliminary results will be presented.

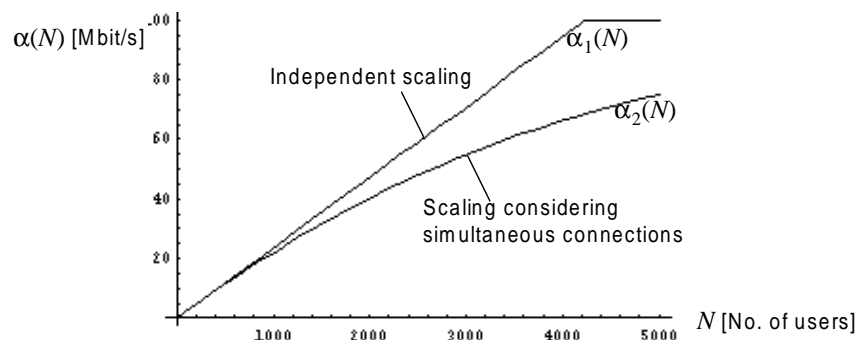


Figure 10 Theoretical bounds on the TCP throughput over a 100 Mbit/s interface

#### 4 Closing comments

The GenSyn is a Java process that generates IP traffic using a flexible, scalable, stochastic, modelling framework for describing the user behaviour of Internet sources. This stochastic behaviour model is linked to the underlying protocol stack and generates real packets into the network. This is a novel modelling approach that picks the best of two worlds; flexibility and scalability of composite state models, and accuracy in the protocol behaviour by use of the underlying protocol stack instead of making a model of it.

#### Acknowledgements

I would like to thank Manyi Lu, SINTEF Telecom and Informatics, and Per Christian Bjelke, Telenor R&D for assistance on program design and for excellent Java programming to make GenSyn come alive. I would also like to thank my former employer SINTEF Telecom and Informatics for supporting phase one of the project.

#### References

- [Ake99] Å. Arvidsson. *On traffic models for TCP/IP*. 16th International Teletraffic congress, ITC'16, Edinburgh, UK, June 7-11 1999.
- [BaCr98] P. Badford and M. Crovella. *Generating representative Web workloads for network and server performance valuation*. In Proceedings of ACM SIGMETRICS'98.
- [ChLi99] H.-K. Choi, J.O. Limb. *A Behavioural Model of Web Traffic*. 7th International Conference on Network Protocols (ICNP'99), October 1999, Toronto, Canada. extended version: <http://users.ece.gatech.edu/~hkchoi/paper/model.pdf>
- [HeHo95] B.E. Helvik, L. Hofseth. *Self-similar traffic and multilevel source models*. In Ilkka Norros and Jorma Virtamo, editors, The 12th Nordic Teletraffic Seminar (NTS-12), pages 407-420, Espoo, Finland, 22 - 24 August 1995. VTT Information Technology.
- [Heeg00] P. E. Heegaard *GenSyn - a Java-based Generator of Synthetic Internet Traffic Linking User Behaviour Models to Real Network Protocols*. Accepted for presentation at ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Sept. 18-20, 2000, Monterey, CA (USA).
- [Helv95] B. E. Helvik. *Synthetic Load Generation for ATM Traffic Measurements*. *Elektronikk*, vol. 91, no. 2/3, pp. 174 - 194, October 1995.
- [HMM93] B. E. Helvik, Ole Melteig and Leif Morland: *The synthesized traffic generator; objectives, design and capabilities*. In Integrated Broadband Communication Networks and Services (IBCN&S). IFIP, Elsevier, Copenhagen, Denmark, April 20-23 1993. (STP40 A993077)
- [LTWW94] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. *On the self-similar nature of Ethernet traffic* (extended version) *IEEE/ACM Transaction of Networking*, pp 1-15, 1994.
- [MTW99] G.J. Miller, K. Thompson, and Rick Wilder. *Performance Measurements on the vBNS*. In proceedings from Interop '98, Las Vegas, NV, May 1998.
- [Rose95] O. Rose. *Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems*. Technical Report 101, University of Wurzburg, Institute of Computer Science, February 1995. The traces obtained from: <ftp://info.informatik.uni-wuerzburg.de/pub/MPEG>.
- [SBit] SmartBits info: <http://www.netcomsystems.com/>
- [Vic98] N. Vicari. *Models of WWW-Traffic: a Comparison of Pareto and Logarithmic Histogram Models*. University of Wurzburg, Institute of Computer Science. Research Report Series. Report No.: 198 March 1998
- [WTSW97] W. Willinger, M.S. Taqqu, R. Sherman, D.V. Wilson. *Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level*. *IEEE/ACM transactions on networking*, vol 5, no 1, Feb 1 1997, pp 71-86.