A realisitic VoIP traffic generation and evaluation tool for OMNeT++

Mathias Bohge
Telecommunication Networks Group, TU Berlin
Einsteinufer 25, 10587 Berlin, Germany
bohge@tkn.tu-berlin.de

Martin Renwanz
Telecommunication Networks Group, TU Berlin
Einsteinufer 25, 10587 Berlin, Germany
renwanz@tkn.tu-berlin.de

ABSTRACT

The fraction of voice over Internet Protocol (VoIP) based telephone calls among the totality of voice based communication acts has been significantly growing during the last years. In wired, as well as wireless communication applications, VoIP is expected to completely replace former circuit switched telephony approaches, and is, thus, a major factor to be considered when designing sophisticated communication networks. The C++ based simulation library OMNeT++ has gained a lot of attention among the communication networks research community. It is being used as a design tool for next generation networks and their performance evaluation. In this paper, we present an OMNeT++ based VoIP traffic generator that creates realistic VoIP packet streams thanks to the utilization of real audio data and an existing VoIP standard codec. Moreover, by applying ITU-T's perceptual evaluation of speech quality (PESQ) approach at the sink, the perceived quality of a transmitted VoIP stream can be determined. The process of creating, transmitting, receiving and evaluating VoIP streams is carefully explained. Some examples are presented and an exemplary quality evaluation is done. [1]

Keywords

Voice over IP, VoIP, G.726, PESQ, OMNeT++, Simulation, Quality Evaluation

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

1. INTRODUCTION

The group of voice over Internet protocol (VoIP) related applications belongs to the fastest growing Internet applications today. Compared to the traditional circuit switched telephone service, packet switched VoIP telephony requires significantly less system bandwidth per call, and, thus, is expected to completely displace the plain old telephone system (POTS). Moreover, VoIP has entered the domain of wireless applications. It is being used over wireless local area networks (WLANs), as well as in global cellular systems, such as the Universal Mobile Telecommunications System (UMTS). However, especially in WLAN [1], but also in cellular [2] and even wired applications, the use of VoIP poses a high number of challenges due to its special transmission requirements, such as low delay, or low jitter. Even though VoIP has been around for quite some time, there is a crucial need for an improvement in VoIP transmission performance in almost all networks. Moreover, the creators of next generation communication networks have VoIP as one of the major factors that determine the network design. As a consequence, there is a lot of VoIP related communication networks research going on at the moment.

Simulation tools are a frequently used opportunity to explore communication network behavior. Simple models of upcoming sophisticated networks are relatively easy to be implemented. Moreover, simulation models that can be used for testing performance increasing ideas already exist for a high number of different network types. The C++ based simulation library OMNeT++ [3] has gained a lot of attention among the communication networks research community. At the simulator's project page, there is a high number of standard network implementations, modules, items, as well as simulation tools freely available. However, to the best of our knowledge, there is no VoIP traffic generator implementation among the available tools. Hence, so far each OMNeT++ user that wanted to test his simulated network on VoIP traffic behavior had to implement its own VoIP traffic generator. As a consequence, there is no unique way of treating VoIP in OMNeT++ simulations.

In this paper,we present an OMNeT++ based VoIP traffic generation and evaluation tool that creates realistic VoIP packet streams thanks to the utilization of real audio data and the application of a VoIP standard codec G.726 [5]. It takes an arbitrary sound file (e.g. WAV or MP3 format) as input, creates according packet streams that are sent via standard OMNeT++ connections. Moreover, it tracks packet errors, and reassembles the sound file at the receiver side, such that the original can be compared to the received version using ITU-T's perceptual evaluation of speech quality (PESQ) standard [6]. In the following, the process of creating, transmitting, receiving and evaluating VoIP streams

[1]This work has been supported by the German Ministry of Education and Science (BMBF) and Ericsson Research, Germany, in the context of the project ScaleNet.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
OMNeT++ 2008 March 3, 2008, Marseille, France
Copyright 2008 ACM 978-963-9799-20-2 ...$5.00.

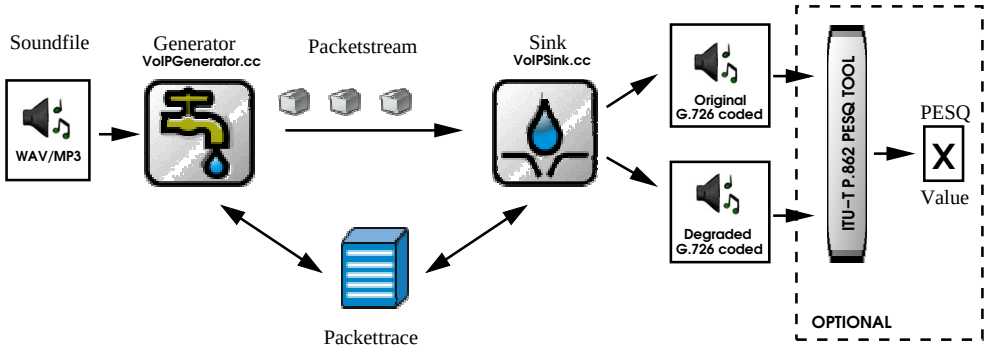

Figure 1: The presented tool consists of the packet generator, and a sink. Optionally the transmission results can be used as input for ITU-T's perceptual evaluation of speech quality (PESQ) tool to obtain the according PESQ value.

MOS	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

Table 1: ITU-T's Mean opinion score (MOS).

is carefully explained. Some examples are presented and an exemplary quality evaluation is done.

The remainder of the paper is structured as follows: In the next chapter, we briefly review the PESQ idea. Then, in Chapter 3, the general design and functionality of our tool is introduced. In Chapter 4, we explain how to download and install the required components. Moreover we introduce the VoIP related simulation parameters in use. Afterwards, in Chapter 5, we show some example studies that are easy to be reproduced. Finally, in Chapter 6, we present our conclusions.

2. PESQ

PESQ stands for 'Perceptual Evaluation of Speech Quality' and is an enhanced perceptual quality measurement for voice quality in telecommunications. PESQ was specifically developed to be applicable to end-to-end voice quality testing under real network conditions, like VoIP, POTS, ISDN, GSM etc.

Each PESQ test results in the mean opinion score (MOS), which expresses voice quality. The PESQ MOS as defined by the ITU recommendation P.862 [6] ranges from -0.5 (worst) up to 4.5 (best). This is different from ITU's original MOS scale, which ranges up to 5.0 (see Table 1). The explanation, however, is simple: PESQ simulates a listening test. It, thus, is optimized to reproduce the average result of all listeners. Statistics show that the best average result one can expect from a listening test is ca. 4.5 (it seems like human beings are always cautious to score a 5, meaning "excellent", even if there is no degradation at all). In order to differentiate between the original MOS and the PESQ-MOS value, we will simply refer in the following to the PESQ-MOS value as PESQ value.

3. VOIP TOOL DESIGN

The general design of the VoIP traffic generation and evaluation tool is shown in Figure 1. It mainly consists of a VoIP packet generator and a VoIP sink.

3.1 VoIP packet generator

The VoIP packet generator's task is to create a stream of VoIP packets out of an arbitrary sound file. Before the actual simulation, the generator parses the input sound file and fragments it into pieces that fit into a VoIP packet, whose size is determined by the combination of the VoIP parameters *samplesPerPacket* and *codeRate* (for a detailed description of all VoIP related simulation parameters see Section 4.4). For the audio-file analysis, the free C++ audio/video library *avcodec* of the FFmpeg project is used [7]. The packet-size audio snippets are individually parsed. Depending on the individual maximum signal amplitude, the packet generator decides on whether an actual VoIP packet, or a silence packet should be sent to represent the related samples at the sink. The decision is based on a comparison between the maximum signal amplitude and the selected *silenceThreshold* parameter (see Section 4.4). The packet creation information is stored in a packet trace list (see Section 3.3 for detailed information about the trace).

Once the actual simulation is started, the generator parses the packet trace list for packet generation: at initialization, it reads the transmission time of the first packet to be sent and schedules an according event (in OMNeT++ this relates to self-message scheduling). If the event execution time is reached, the related packet is sent to the sink, the transmission time of the next packet is read from the trace, and an according event is scheduled. Note that the audio contents are not actually sent. Instead, a standard OMNeT++ IP message of according size is transmitted. All audio data handling is done at the sink.

3.2 VoIP sink

For each packet that arrives at the sink, the arrival time is logged in the packet trace. In addition, using the standard OMNeT++ *cMessage* error handling functionality, the packet is checked for transmission errors. An according entry is added to the trace. The trace is then used to create two different wave files:

1. a wave file representing the original sound file at a sampling rate of 8000Hz, and

2. a wave file representing the transmitted G.726 encoded VoIP stream, including silence packets and packet errors.

Note that all erroneous packets are omitted. They are replaced by silence in the reconstructed sound-files. For human subjective evaluation, the created sound files can be played in standard wave file players.

Also note that the avcodec library contains a number of different codecs that might be used to replace the chosen G.726. Since our implementation is not very G.726 specific, the migration to another codec should be quite easy.

If the PESQ evaluation switch $pesqValueComputation$ is set to 1, the wave files are piped into ITU-T's PESQ evaluation tool. The resulting PESQ value is piped back and stored, together with the number of silence packets and the number of transmission errors, in the results log-file. Note, however, that the ITU-T's PESQ tool needs to be individually downloaded and properly installed (for details on prerequisites and installation see Section 4).

3.3 VoIP packet trace

The VoIP packet trace is created at the generator. Additional information about transmission errors and the packet arrival time is added at the sink. It is held in memory, such that both entities have direct access to it. However, if the $writeTracesToDisk$ switch is set to 1, the trace is written to disk and can be used for further processing and analysis. For each packet belonging to a VoIP stream, the trace holds the following information:

- transmission time
- arrival time
- type (VoIP/silence)
- size (packet size in bit)
- pos/wav (position in the related wave-file)
- packetNo (VoIP stream packet number)
- error (packet erroneous true(1)/false(0))
- wavfile (the wave file the packet belongs to)

Recall that a single VoIP stream consists of multiple concatenated sound files, if random sound file selection is chosen (see Section 4.4) and the selected simulation time is greater than the duration of the randomly selected sound file.

If, due to simulation time limits, some sound packets are not send, those packets are marked with the packet number '-1'in the trace file.

4. USING THE VOIP TOOL

In this section we explain how to set up and use the tool.

4.1 Prerequisites

In order to use our VoIP traffic generation and evaluation tool you need to download and install a working OMNeT++ version (3.1 or higher) from [3] (note that for OMNeT++ to run, you will also have to download the appropriate TK and TCL libraries). If you want to evaluate the VoIP transmission using ITU-T's P.862 PESQ tool, you need to download and install it as well. The P.862 standard in combination with the evaluation tool source code is available at the ITU homepage [4]. After having OMNeT++ (and optionally the PESQ tool) up and running, you can download our VoIP traffic generation and evaluation tool from [8]. Note that the download file *OMNeT_VoIPTool.tgz* is a zipped tar-archive file, which can be decompressed calling 'tar xvzf OMNeT_VoIPGenerator.tgz' on a Linux shell, or using an arbitrary archive manager.

4.2 Installation

Once the archive file is downloaded and decompressed, switch to the just created *VoIPGenerator* directory. Then execute the shell script *makemake.sh*, which generates a Makefile that includes all necessary libraries (using the opp_makemake command). Calling *makemake.sh* will also set links to the codec libraries, which are stored in the *libs* folder. After that, type *make* to compile the generator code. If the compilation was successful, a *VoIPGenerator* executable has been created. The generator is started by executing it. Note that if you want to use ITU-T's PESQ evaluation tool, the executable file *pesq* must be stored in the *VoIPGenerator* directory.

4.3 OMNeT++ Example Network Setup

The present VoIP traffic generation and evaluation tool can be used in arbitrary OMNeT++ simulation environments. It can e.g. be used in combination with the OMNeT++ standard framework INET (see [3] for more details). An exemplary combined application of the VoIP tool and the INET framework is described in the Appendix. However, for simplicity reasons, in the following we assume the tool to be used as a stand-alone tool. We provide an according simple example network setup (stored in the *SimpleNet.ned* file) consisting of a single source, a single sink, and a wireless channel. Note, that in OMNeT++ the wireless channel characteristic is determined by the following properties:

- propagation delay (in seconds)
- bit error probability
- datarate (in bits/second)

These wireless channel characteristics can be changed by modifying the according values in the *SimpleNet.ned* file.

4.4 VoIP Simulation Parameters

This section provides an overview of the VoIP generation and evaluation related simulation parameters we use in our implementation. All parameters can be accessed and modified in the *omnetpp.ini.* file.

Samples per packet.

The number of audio samples that are sent in a single packet. Note that G.726 works with 8000 Hz as sampling frequency. Thus, the length t_{packet} (in seconds) of each sound

snippet that corresponds to a single VoIP packet can be computed as follows:

$$t_{packet} = \frac{samplesPerPacket}{8000} \quad (1)$$

Consequently, the length of a standard 64 samples packet can be computed to $t_{packet} = 8ms$.

Coding rate.

The G.726 codec supports 4 different coding rates (in bits/second): 16000, 24000, 32000, 40000. Coding rates different from these four are not supported. Note that the VoIP packet payload size is determined by the combination of the two parameters *samples per packet* and *coding rate*.

Voip header size.

The VoIP header size in *bits*. A common size is $32bits$ ($= 4bytes$), which is the size of the standard real-time transport protocol (RTP) header. A regular VoIP packet consists of the header and the coded audio data.

Silence threshold.

Our generator, as most VoIP generators, owns the opportunity to create silence packets that omit the VoIP payload, if the amplitude of the sound signal is low. In this particular case, the highest absolute amplitude value among the current $samplesPerPacket$ samples is compared to the threshold. If it is smaller, the samples are omitted. Consequently, silence packets solely consist of the VoIP header. Thus, their utilization significantly increases the network's efficiency. To disable silence packet generation in the simulation, the value can be set to 0.

Write-traces-to-disk switch.

The tool provides the possibility to store the information about the transmission by creating traces and store them on the disk. The traces are written to disk, only if this parameter is set to '1'.

Tracefile base-name.

A string variable that is used as the first part of the name of created trace-files, if $writeTracesToFile$ is set to 1.

Results file name.

A string that holds the name of the file that logs the simulation results, such as transmission errors, silence packets and total number of packets, and (optional) the PESQ values.

Sound file directory.

Another string variable that holds the path for the sound files to be processed by the generator.

Sound file selection.

This string holds the name of the sound file to be used. If it is set to "random", the generator randomly selects a sound file from the sound file directory.

Converted sound file names.

The sink creates two wav files once all packets of a VoIP stream are received:

1. a down-converted (8000Hz) wave file of the original sound file, and
2. a (degraded) wave file that corresponds to the transmitted VoIP stream.

The two wave files serve as input for the PESQ tool, if PESQ evaluation is switched on.

Parameter	**Value**
Nr of Soundfiles	4
Coderate	16; 32; 40 kBit/s
Silence Threshold	0; 100
OMNeT++ channel delay	0 s
OMNeT++ channel error (BEP)	0; 10e-3; 10e-4
OMNeT++ channel data rate	2 MBit/s

Table 2: Example VoIP stream and OMNeT++ channel parameter settings.

PESQ value computation switch.

If set to 1, the two G.726 coded wav files (original and by transmission degraded version) constructed at the sink are piped to ITU-T's PESQ tool. Recall that it needs to be correctly installed (see Section 4.2). The PESQ value is computed and piped back to the sink for furhter processing.

5. EXAMPLE SIMULATION

Using the simple OMNeT++ example network setup introduced in Section 4.3, we have conducted a number of example test runs. We have tested our tool by utilizing four different input sound files, three different code rates, two different silence threshold values, as well as three different OMNET++ channel error (bit error probability - BEP) settings (for details see Table 2).

5.1 Results

In Figure 2 the simulation results are shown. The two graphs on top represent the perceptual evaluation of speech quality (PESQ) values of the four files in an ideal channel scenario (channel error is set to 0). By comparing the values on the left side, which represent the quality for the G.726 encoded audio files without silence packet generation ($silenceThresh$ set to 0), with the right side values, which correspond to the adequate files assuming silence packet generation with a $silenceThresh$ set to 100, it can be seen that the use of silence packets can significantly degrade the perceived VoIP stream quality. Note that the degradation of the music sound files (*music_loud.mp3* and *music_soft.mp3*) is smaller, because the fraction of silence packets among all generated packets per stream is smaller than for the voice files. In all cases, the PESQ value increases with the encoder's code rate (16kBit/s, 32kBit/s, 40kBit/s). In some cases, the optimal PESQ value of 4.5 is almost reached.

The lower two graphs show the results of the simulated non-ideal channel scenarios. In all cases, silence packet generation with a $silenceThresh$ set to 100 is considered. At the left side, the packet errors subject to the encoder's coderate and the channel error setting (BEP) are presented. As expected, in all cases the order of magnitude of the packet errors decreases with the order of magnitude of the BEP

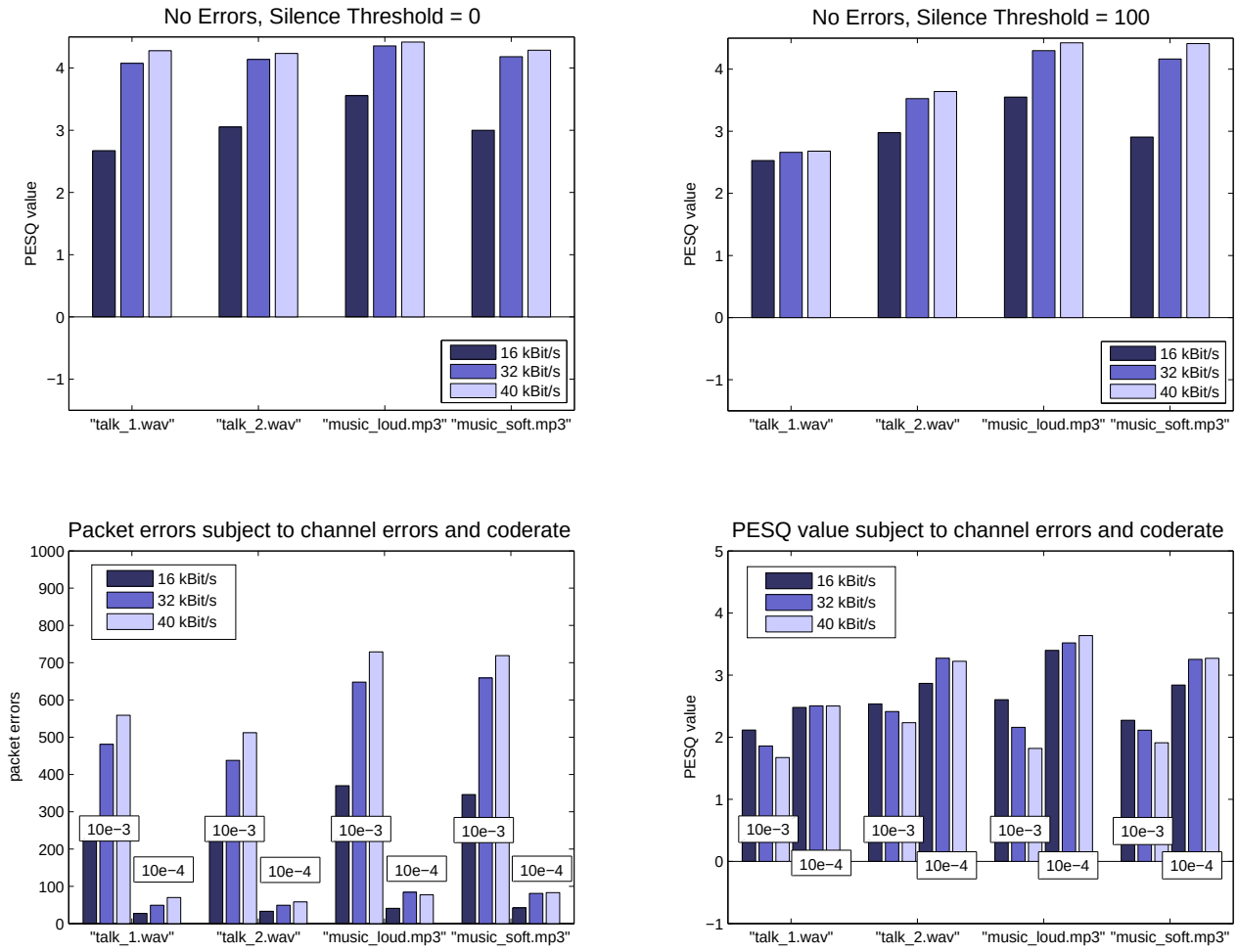

Figure 2: The simulation result graphs.

(10e-3, 10e-4). However, note that the number of packet errors per file increases with the encoder's code-rate. This is due to the fact that higher code-rates yield larger VoIP packets, and large packets are more susceptible to bit errors than small packets for the same BEP.

The related PESQ values are shown at the lower right side of Figure 2. Comparing the PESQ values to the values in the two graphs on top, it can be seen that in all cases a significant degradation due to channel errors has happened. Note that the PESQ values in the low channel error setting (BEP=10e-4) mostly increase with the encoder's code-rate, while they decrease in the high channel error scenario (BEP=10e-3). This is due to the fact that the difference in packet errors is much smaller in the low channel error scenario, and, thus, a higher code rate accounts for a better sound quality. In the high channel error scenarios, the additional amount of erroneous packets when switching to a higher code-rate is so large that the quality loss due to packet losses is larger than the quality gain due to the higher encoder code-rate.

6. CONCLUSIONS

The increasing importance of VoIP related traffic in today's communication networks makes it one of the most important factors when it comes to the design of future communication network architectures. A major tool for designing communication networks are network simulators, such as the C++ based network simulation library OMNeT++. However, so far no OMNeT++ based VoIP generation and evaluation tool has been standardized and made available to the OMNeT++ community. In this paper we have presented an according tool that offers the opportunity to generate, transmit, receive and evaluate realistic VoIP packet streams within the standard OMNeT++ simulation environment. It uses a standard VoIP codec (G.726) and ITU-T's PESQ evaluation tool to process arbitrary sound files, convert them to VoIP packet streams, and evaluate the stream's quality in terms of ITU-T's perceptual evaluation of speech quality (PESQ) metric. We believe that our VoIP traffic generation and evaluation tool will significantly ease and improve the application of OMNeT++ in the context of VoIP related communication network research.

7. REFERENCES

[1] W. Wang, S.C. Liew, and V.O.K. Li, "Solutions to performance problems in VoIP over a 802.11 wireless LAN," *IEEE Transactions on Vehicular Technology*, vol. 54, pp. 366–384, Jan. 2005.

[2] W. Bang, K.I. Pedersen, T.E. Kolding, and P.E. Mogensen, "Performance of VoIP on HSDPA," in *Proc. of the 61st IEEE Vehicular Technology Conference (VTC 2005-Spring), Stockholm, Sweden*, May 2005, vol. 4, pp. 2335–2339.

[3] "Omnet++ 3.3," Project Homepage: http://www.omnetpp.org [status:12/07/2007].

[4] Telecommunication Standardizatoin Sector of ITU (ITU-T), "Perceptual evaluation of speech quality (PESQ) tool, available at: http://www.itu.int/rec/t-rec-p.862-200102-i/en [status:12/07/2007]," .

[5] The International Telegraph and Telephone Consultative Committee (CCITT), "Recommendation g.726: 40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM)," Oct. 1990.

[6] Telecommunication Standardization Sector of ITU (ITU-T), "Recommendation P.862: Perceptual evaluation of speech quality (PESQ): An objective mehtod for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," Feb. 2001.

[7] FFmpeg Project, "AVCODEC: the leading audio/video codec library," Project web-page: http://ffmpeg.mplayerhq.hu [status:12/12/2007].

[8] M. Bohge and M. Renwanz, "A realistic VoIP traffic generator for OMNeT++, source code," available at http://www.tkn.tu-berlin.de/research/omnetVoipTool [status: 2007/12/12].

APPENDIX

A. COMBINED USAGE WITH INET

The VoIP tool presented in this paper was originally implemented as a stand- alone simulation. However, only a few modifications are necessary to use the tool in combination with other simulation models, such as the INET framework (a sophisticated TCP/IP protocol suite for OMNeT++), which is freely available at the OMNeT++ homepage [3]. To ease the usage of the VoIP tool with this frequently used framework, an additional source code archive file that includes all necessary modifications is provided on our project homepage [8].

A.1 Additional Installation Steps

In addition to the regular installation process (described in Section 4.2), the following steps need to be taken: In order to use the INET enabled version of our tool (that, of course in an initial step needs to be downloaded from our project homepage [8]), you need to download and build the INET framework.

After unzipping the VoIP toll source code and before executing the shell script *makemake.sh*, open it with any text editor, and edit the following line concerning your system:

INET="/path/to/your/INET".

Afterwards execute it. In addition to creating links to all necessary libraries, calling *makemake.sh* will also create the script you need to execute in order to run the simulation: *InetVoIPTest*.

Then, type 'make' to compile the generator code. The dynamic library *OMNeT_VoIPTool_INET.so* has been created, if the compilation was successful. The generator can now be started by executing the *InetVoIPTest* script. Note that, *opp_makemake* names the output file (here, the name of the dynamic library) according to the name of the folder the command is executed in. Thus, if you move the files to another folder, the library will have a different name. If you choose to move the files to another folder, you need to modify the file *omnetpp.ini*: The name of the library to load must be specified (without the .so extension) in the line "load-libs =" in the General section.

Recall that if you want to use ITU-T's PESQ evaluation tool, the executable file 'pesq' must be stored in the 'VoIP-Generator' directory.