Using Network Simulators with Video Traces

F.H.P. Fitzek P. Seeling M. Reisslein*

acticom GmbH – mobile networks
R & D Group
Germany
fitzek@acticom.de

Arizona State University
Department of Electrical Engineering
USA
[reisslein|seeling]@asu.edu

May 8, 2003

Video traces for the evaluation of network performance are publicly available today. With the content–dependence of video sequences, the generation of an universal video source model is not possible. As the video models to date all use the video frame sizes for the adjustment of the model parameters, video traces are needed for the application of these models. The design and layout of networks, services, and their respective QoS requires early simulations, discrete event simulators are applied in early development stages. As video traffic is of high importance for current and future networks, the need for video sources in simulator packages arises. In this report, we introduce interfaces from different video traces to three major simulators.

*The work of M. Reisslein is supported in part by the National Science Foundation through Grant No. Career ANI-0133252 and Grant No. ANI-0136774. Any opinions, findings, and conclusions or recommendations expressed in this material are these of the authors and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Figures

List of Tables

1 Introduction

Video traffic is supposed to account for a large portion of future wired and wireless network traffic. The evaluation of different video coding standards (e.g., H.263 [1, 2, 3], MPEG–4 [4, 5, 2, 3], or H.264 [6, 7]), the resulting video traffic, and the resulting requirements for networks have attracted great interest in the research community. As encdoded video traffic is dependent to the content [8, 9], the encoding standard, and the encoder settings, no independent video source model can be developed. In order to facilitate the network performace evaluation, quality of service (QoS) categorizations, and service designs, the prospective video traffic has to be evaluated. The difficulties in modeling the behavior of video sequences with different encoding modes and different contents is driving the utilization of network simulators for performance analysis puposes. A large portion of research in the field of video traffic analysis is thus on video traces. The variety of different video stream characteristics, user behavior, and the properties of the transporting network is huge. Models of video sources such as presented in [9, 10, 11, 12, 13, 14] can greatly improve the possibilities to test final systems with regard to some real–life applications. However, these models require several to several tenths parameters. The creation of such a model is therefore only possible if the video itself has been statistically evaluated before. To evaluate an encoded video sequence, the video trace files (i.e., the video frame sizes over time) are utilized. However, these video traces can also be used for the use in different network simulators. The main purpose of using network simulators with video traces is to facilitate the layout of the network with respect to its physical form, QoS, and other parameters. As modeling the video sources always requires the original trace to be fully evaluated first, a simpler approach is to incorporate these traces directly into the network simualor. Another benefit from incorporating the video traces directlky into network simulators is the vast amount of video source models, each with a specific point of view. The definition of an interface between the network simulator and the video trace is necessary. An overview of the necessary steps for incorporation of video sources is illustrated in Figure 1. Clearly the direct utilization of video traces in network simulators facilitates the fastest method to incorporate video sources into existing network models. It is thus necessary to have clearly defined interfaces to facilitate a timely approach to any incorporation into existing models. In the following we explain how to

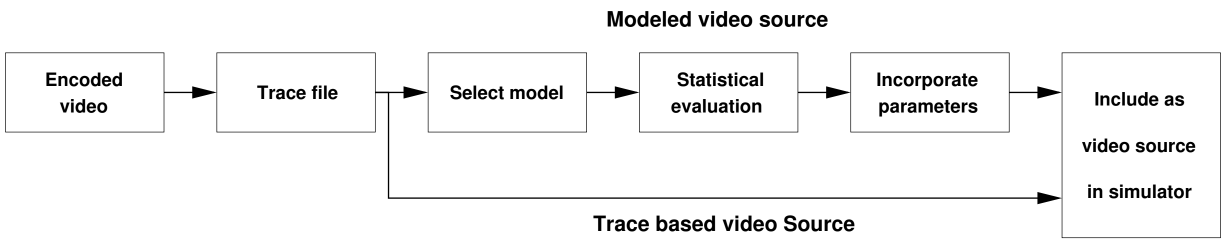


Figure 1: Overview of modeling vs. trace approach.

use existing interfaces for the most popular network simulators. These interfaces were designed by various people. All of these interfaces are made available through our video trace site and its mirrors [15].

2 Trace Files

Video trace files were generated as early as 1995, started by Rose [16]. As the encoder generations moved onward to the currently widespread MPEG–4 standard [4], the trace files generated also started to differ. As the frame rate per second is usually fixed within a single encoded video stream, the frame numbers or the playout time can be regarded as redundand. The trace files presented on our webpage [15] use different column positions to present the values needed for the different simulator interfaces. The presented tracefiles also differ with respect to the information presented. This results from the evolutionary development in the generation of video encoding standards and related software encoders. Additionally, the depth of research in the video coding domain has increased. In the beginnning of the research based on video frame traces, the focus was solemnly on the network performance. Today, the issues of quality and their correlation to the video frame sizes is also of importance, driven by the need for differential QoS. For utilization of the trace files in the simulators shown in Section 3, some of the interfaces have to be adapted. It is therefore necessary to redefine the routines of the interfaces that read the trace files. The following groups of frame traces are using the same column layout.

Table 1: Different video trace file formats.

Group	**Entries in file**						
MPEG–4 (new)	`frame#`	`time`	`type`	`size[byte]`	`PSNR-Y`	`PSNR-U`	`PSNR-V`
MPEG–4 FGS	`frame#`	`type`	`size[`*bit*`]`	`PSNR-Y`	`PSNR-U`	`PSNR-V`	
H.26L MPEG–4 (old)	`frame#`	`type`	`time`	`size[byte]`			
H.263 H.261	`time`	`type`	`size[byte]`				

Please note that up to now, we only incorporate the single layer verbose trace files. As most of the interfaces are programs written in different scripting languages, the positions to change within are easy to find and in all cases single or few lines. The *Omnet++* interface is an exception since it automatically recognizes the two different groups of MPEG–4 trace file formats.

3 Trace Based Video Sources for Network Simulators

Several simulator packages for utilization in network performance analysis exist. Mostly of interest in this field are discrete evenet simulators. In the following, we introduce the interfaces by which the video trace files can be used in these simulators as video sources. Three major distributions of simulators are shown with an exemplary demonstrator for each of them. With the demonstrator incorporating the video trace based sources, anybody can easily derive advanced configurations from these.

3.1 NS II

The interface for using the video traces with the *Network Simulator — ns–2* [17] was written by Michael Savoric from the Technical University of Berlin, Germany. The example provided at [15] is using the trace file model as provided with the different groups of video traces. The *Network Animator* package is used to generate a visual representation of the frame sending mechanism. As the interface was written for an earlier version of the *Network Simulator*, a few warnings may appear in newer versions. Nevertheless, the provided *TCL*–scripts are tested to work with the version to date. With some experience in *TCL*–scripting, the interface can easily be modified to work with additional, such as fine grain scalability, trace file versions. In this demontrator, all links between entities shown are set to a capacity of 1Mbit. The running simulator and visualization package is illustrated in Figure 2. For a successful compilation of the demonstration package, you need to:

1. Extract the package: `tar xvzf nsexample.tar.gz`.
2. Change into the created directory: `cd nsExample`.
3. Invoke the demonstration: `ns video_trace_`*`type`*`.tcl`, where *type* denotes the group of trace files introduced above.
   (Please note that it may take some time for the script to parse the trace file and create the *ns* input file.)

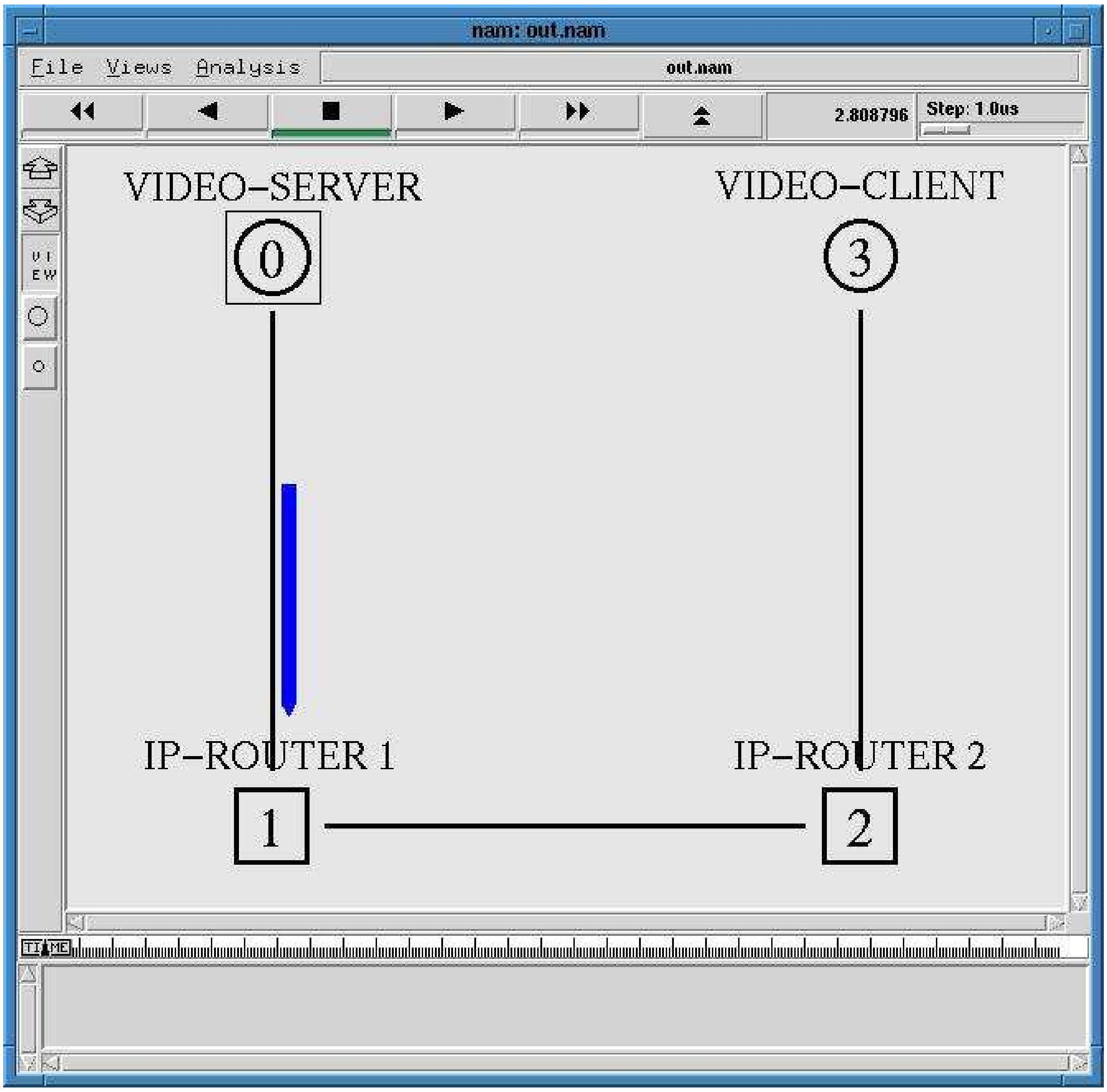


Figure 2: NS–2 trace based video source with NAM sample output.

3.2 Omnet++

The interface for the *Omnet++* simulator was written by Luca Signorin from the University of Ferrara, Italy. The interface is capable of detecting the different video trace file formats and to feed the data into the *Omnet++* simulator accordingly. The trace files above use different columns for the representation of the data. The *Omnet++* simulator package is available from [18]. The current stable *version 2.2* is the version the interface was written for and tested with (Note that the currently available version 2.3 beta is not compatible with the interface). The following steps are neccessary for a successful compilation of the interface and demonstration package:

1. Extract the downloaded file: `tar xvzf OMNET++VideoInterface.tar.gz`.
2. Change into the created directory: `cd OMNET++VideoInterface`.
3. Edit the `Makefile` and change the paths `NEDC=`*your_omnet_dir*`/src/nedc/nedcwrapper.sh`, `OMNETPP_INCL_DIR=`*your_omnet_dir*`/include`, and `OMNETPP_LIB_DIR=`*your_omnet_dir*`/lib` to the Omnet++ directory according to your setup.
4. Call `make`.
5. Invoke `./OMNET++VideoInterface`.

For speed up the simulation and skipping the autodetection it is possible set the "type_video" parameter in omnetpp.ini file. The following output and main *Omnet++* windows are shown in Figures 3 and 4. With adjustments to the header and source files, this example can easily be expanded to suit different needs.

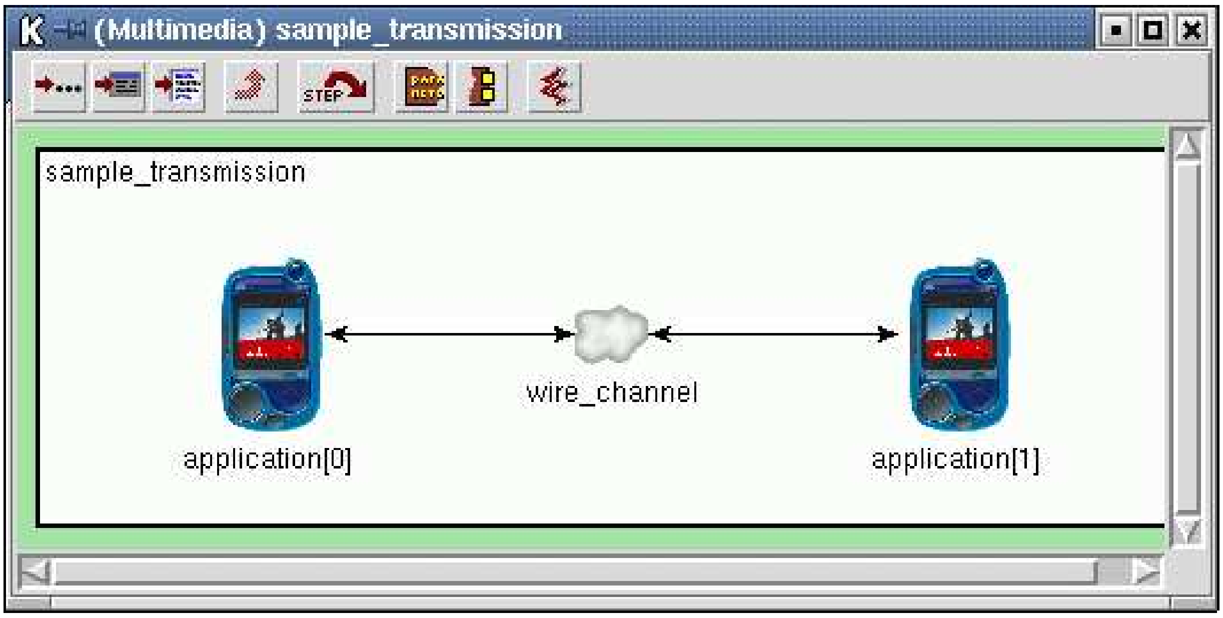
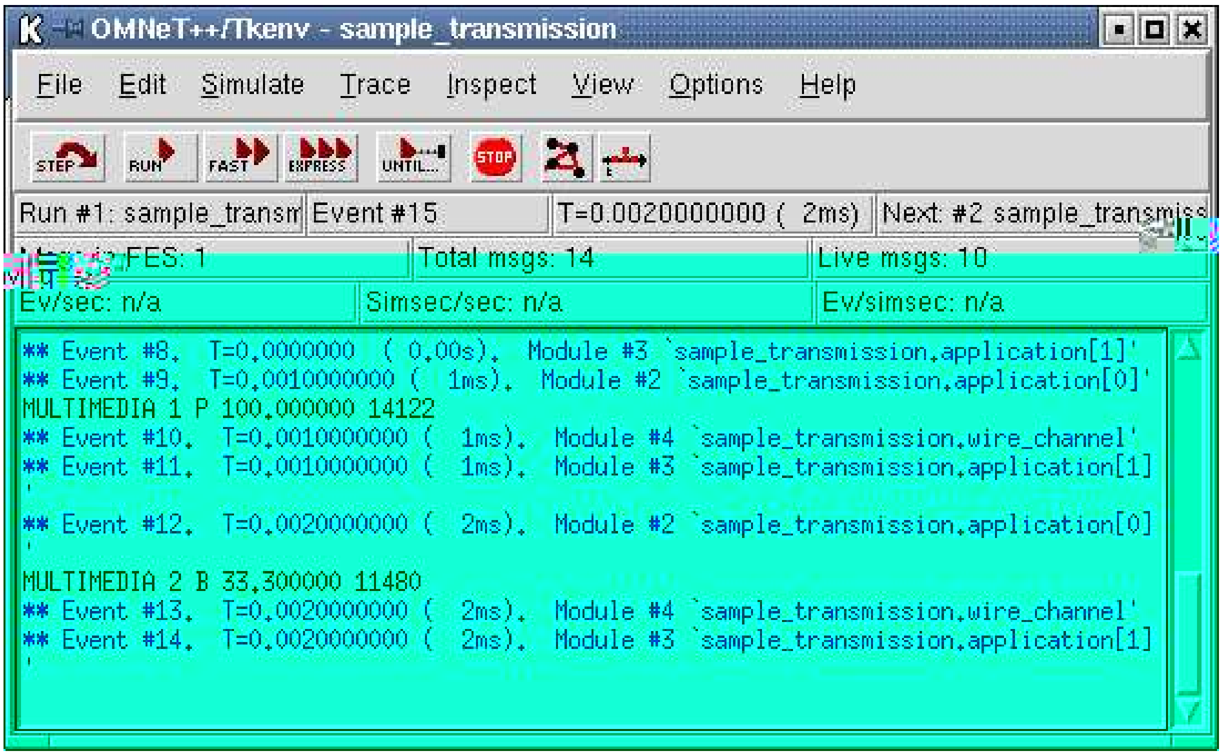


Figure 3: Omnet++ visualisation window for provided example simulation.


OMNeT++/Tkenv - sample_transmission
File Edit Simulate Trace Inspect View Options Help
STEP RUN FAST EXPRESS UNTIL... STOP

```
Run #1: sample_transm  Event #15  T=0.0020000000 (  2ms)  Next: #2 sample_transmiss
Msgs in FES: 1  Total msgs: 14  Live msgs: 10
Ev/sec: n/a  Simsec/sec: n/a  Ev/simsec: n/a
** Event #8.  T=0.0000000  ( 0.00s).  Module #3 `sample_transmission.application[1]'
** Event #9.  T=0.0010000000 (  1ms).  Module #2 `sample_transmission.application[0]'
MULTIMEDIA 1 P 100.000000 14122
** Event #10.  T=0.0010000000 (  1ms).  Module #4 `sample_transmission.wire_channel'
** Event #11.  T=0.0010000000 (  1ms).  Module #3 `sample_transmission.application[1]
'

** Event #12.  T=0.0020000000 (  2ms).  Module #2 `sample_transmission.application[0]
'

MULTIMEDIA 2 B 33.300000 11480
** Event #13.  T=0.0020000000 (  2ms).  Module #4 `sample_transmission.wire_channel'
** Event #14.  T=0.0020000000 (  2ms).  Module #3 `sample_transmission.application[1]
'
```

Figure 4: Omnet++ main window for provided example simulation.

3.3 Ptolemy II

A third network simulation package, *Ptolemy* [19], was originally used for simulation in the beginning of the video trace project. The interface was written by Frank Fitzek during his time at the Technical University of Berlin. At this time only trace files of the format H.261, H.263, and MPEG (old) were used. Regarding Table 2 also the trace files for H.26L (H.264) should work as they have the same format as the form of the previous H.26x coders. In case newer versions of trace files with different formats are used, minor changes to the source code have to be done. The interface is mainly based on the PTOLEMY star `DEVideo.pl`. The `DEVideo.pl` star is derived from the DERepeatStar domain of the Ptolemy simulator package. A set of parameters as in Table 2 can be used to adapt the interface to the users' needs.

Table 2: Parameters for Ptolemy interface.

Parameter	Type	Info	default
CodingType	StringState	MPEG4 and H263x is supported	H263
TraceList	StringState	file name were trace files are specified	H263TraceList.dat
InputDir	StringState	location of trace files and this→TraceList	$TRACE_PT_DIR/
framescale	FloatState	scaling the frame size	1.0
Offset	FloatState	uniform random phase for sequence start	0.0
maxFrameSize	IntState	larger frames will be skipped	1000.0
Duration	IntState	uniform random trace duration in sec	1000.0
DEBUG	IntState	different DEBUG level for stdout	3

The CodingType specifies which codec is used. This important to read the trace data properly. The TraceList parameter points top a file where all trace file names are specified that should be used for the simulation. Examples:

`Verbose_ARDNews_64.dat`	0.245
`Verbose_ARDTalk_64.dat`	0.745
`Verbose_Aladdin_64.dat`	1
`Verbose_BoulevardBio_64.dat`	1
`Verbose_DieFirma_64.dat`	1

Besides the name also a time scale factor is given. The video trace files differ in their length. The factor multiplicated with 60 minutes gives the real trace length. The files itself has to be stored in InputDir. In case the video frames has to be scaled the parameter framescale has to be set properly. The star choses randomly a video trace file out of the trace file set given in TraceList and starts generating frames with an random offeset (maximum offset given by Offset). This is done to avoid synchronisation effects between different terminals. Frames that are larger than maxFrameSize will be skipped (this option was not used to date, but might be helpful for other researchers). The Duration parameter is used to set the duration time of the video trace. Such as the offset even the duration is set randomly but never larger than the video trace length. If the trace data is fully consumed the star choses a new trace file with new values for the offset and the duration. The DEBUG level is used to check the proper functionality of the star. It should be disactivated for simulations as it produces a lot of out.

4 Acknowledgements

We would like to thank the following people and Instituitions for their efforts on behalf of the video trace libraries and generation of interfaces for popular network simulators:

Telecommunications Network Group
Technical University of Berlin, Germany
http://www-tkn.ee.tu-berlin.de

Adam Wolisz
wolisz@ft.ee.tu-berlin.de
Michael Savoric
savoric@ft.ee.tu-berlin.de

FUTURE Group
University of Ferrara, Italy
http://www-tlc.ing.unife.it/

Michele Zorzi
zorzi@cwc.ucsd.edu
Luca Signorin
luca.signorin@inwind.it

References

[1] D. Turaga and T. Chen, *Fundamentals of Video Coding: H.263 as an example Compressed Video over Networks*, ser. The Signal Processing Series. MarcelDekker, 2000, ch. Fundamentals of Video Coding: H.263 as an example. 3

[2] F. Fitzek and M. Reisslein, "MPEG–4 and H.263 traces for network performance evaluation," Technical University Berlin, Dept. of Electrical Eng., Germany, Tech. Rep., Oct. 2000. [Online]. Available: http://trace.eas.asu.edu 3

[3] ——, "MPEG–4 and H.263 video traces for network performance evaluation," *IEEE Network*, vol. 15, no. 6, pp. 40–54, November/December 2001. 3

[4] R. K. (Editor), "Overview of the MPEG–4 standard, ISO/IEC 14496," May/June 2000. 3, 4

[5] F. Pereiera and E. Touradj, *The MPEG–4 Book*. Prentice Hall, 2002. 3

[6] T. Wiegand, "H.26L Test Model Long–Term Number 9 (TML-9) draft0," ITU-T Study Group 16, Dec. 2001. 3

[7] K. Dovstam, "Video Coding in H.26L Video Coding in H.26L," Ph.D. dissertation, 2000. 3

[8] M. Wu, R. A. Joyce, H.-S. Wong, L. Guan, and S.-Y. Kung, “Dynamic resource allocation via video content and short–term traffic statistics,” *IEEE Transactions on Multimedia*, vol. 3, no. 2, pp. 186–199, June 2001. 3

[9] P. Bocheck and S. Chang, “A content based video traffic model using camera operations,” in *Proceedings of IEEE International Conference on Image Processing (ICIP ’96)*, Lausanne, Switzerland, Sept. 1996, pp. II–817–820. 3

[10] K. Chandra and A. R. Reibman, “Modeling one– and two–layer variable bit rate video,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 398–413, June 1999. 3

[11] M. Frey and S. Nguyen-Quang, “A gamma–based framework for modeling variable–rate MPEG video sources: The GOP GBAR model,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 710–719, Dec. 2000. 3

[12] J. Gao and I. Rubin, “Multiplicative multifractal modeling of long–range–dependent network traffic,” *International Journal of Communication Systems*, vol. 14, pp. 783–801, 2001. 3

[13] D. P. Heyman and T. V. Lakshman, “Source models for VBR broadcast video traffic,” *IEEE/ACM Transactions on Networking*, vol. 4, pp. 40–48, Jan. 1996. 3

[14] H. Liu, N. Ansari, and Y. Q. Shi, “A simple model for MPEG video traffic,” in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, New York, NY, 2000, pp. I–553–556. 3

[15] “Video traces for network performance evaluation,” Online Website. [Online]. Available: http://trace.eas.asu.edu 3, 4, 5

[16] O. Rose, “Statistical properties of MPEG video traffic and their impact on traffic modelling in ATM systems,” University of Wuerzburg, Institute of Computer Science, Tech. Rep. 101, Feb. 1995. 4

[17] “The network simulator — ns–2.” [Online]. Available: www.isi.edu/nsnam/ns/index.html 4

[18] A. Varga, “Omnet++,” *IEEE Network Interactive*, vol. 16, no. 4, 2002. [Online]. Available: www.hit.bme.hu/phd/vargaa/omnetpp 7

[19] E. L. J.T. Buck, S. Ha and D. Messerschmitt, “Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems,” *Int. Journal of Computer Simulation*, vol. 4, pp. 155–182, Apr. 1994. [Online]. Available: ptolemy.eecs.berkeley.edu/index.htm 9