# Modeling Network Protocol Overhead for Video

John McAlarney and Rami Haddad
*University of Akron*
*Department of Electrical and Computer Engineering*
*Akron, OH*
*Email: {jwm7,rjh28}@zips.uakron.edu*

Michael P. McGarry
*University of Texas at El Paso*
*Department of Electrical and Computer Engineering*
*El Paso, TX*
*Email: mpmcgarry@utep.edu*

***Abstract*—With the gaining popularity of video communication over the Internet, it is important for networking protocol researchers to have tools to analyze and simulate video. We present the results of an extensive study to understand which applications and protocols are used for video communication over the Internet. We obtained an extensive list of sources of video content around the Internet (e.g., Netflix, Hulu, and YouTube) and used Wireshark to capture packets containing video from these various video sources. We found Adobe Flash to be, by far, the most common application used to deliver video over the Internet. Adobe Flash uses its own proprietary application layer protocol called RTMP. We used the results of this study to develop analytical models for the network protocol overhead added to video that is communicated over the Internet. In addition, we have developed simulation tools that we are making available to the research community.**

***Keywords*-Video; Multimedia; Modeling; RTMP; Flash Video;**

## I. Introduction

The simulation of packet video traffic sources is becoming increasingly important for packet network performance analysis. This increasing importance is a result of the proliferation of video content distribution over packet networks (e.g., web video content, and IPTV networks). According to [1], from 2000 to 2005, the amount of available video content on the Internet increased by over 600%. In addition, Cisco is projecting that video will account for 90% of all IP traffic by 2012 [2]. Unfortunately, there is no general model that accurately characterizes packet video traffic. As a result, video traces [3], [4] must be utilized to facilitate the simulation of packet video traffic sources. Video traces are text files that contain information about the playback time and size of video frames of a particular video. To accurately simulate video content delivery over a packet switched network, it is necessary to model how the video frames are segmented into packets and encapsulated in network protocols for transmission.

We have conducted a study to understand how video content is encapsulated in network protocols for delivery over the Internet. This study was facilitated by the use of the Wireshark [5] packet capturing utility. In this article, we present the results of this study.

Awareness of the network protocols in widespread use on the Internet to deliver video will allow the research community to develop accurate models, both analytical and simulation, for the analysis of video communication over packet switched networks.

We use the findings from this study, to develop analytical models of protocol overhead for video communication over the Internet. In addition, we use these findings to develop a tool that turns video traces [3], [4] into packet traces using the protocols we found in widespread use. The dynamics of the various protocol layers (e.g., acknowledgments, and retransmissions) are abstracted out by this tool. The tool models the video frame segmentation and added protocol overhead. For instances in which the dynamics of these protocols need to be modeled, it is advised to use this tool as a reference point. Finally, we have developed an OMNeT++ [6] video traffic model that utilizes these packet traces to simulate a video traffic source. We are making both the video trace to packet trace tool and the OMNeT++ video traffic model available for download at the following URL: http://ee.utep.edu/mcgarry/UTEP_mcgarry/Tools.html.

To our knowledge, we are the first to comprehensively identify the network protocols used for video communication over the Internet. We are also the first to make publicly available, a video traffic simulation model that accounts for network protocol overhead.

In [7], the author briefly outlines the protocol overhead for video with HTTP/TCP and RTP/UDP. In [8] the authors outline the inefficiency of using RTP for the transport of video when coupled with the overhead added by the CTS/RTS collision avoidance mechanism in IEEE 802.11. The authors in [3] proposed three different video traffic models using (NS II, OMNeT++, Ptolemy II).

We found no existing literature that identifies, comprehensively, which protocols are used in practice over the Internet and present resulting analytical models of the overhead. In addition, the publicly available video traffic models that we found did not account for the network protocol overhead. Modeling the network protocol overhead is important for accurately characterizing the bandwidth requirements of video.

This article is organized as follows, in Section II we

978-0-7695-4308-6/10 $26.00 © 2010 IEEE
DOI 10.1109/EMS.2010.68

present our network protocol findings, in Section III we present our analytical models of the network protocol overhead, in Section IV we present the simulation tools we have developed, and we conclude the article in Section V.

## II. Protocols Used for Video Communication

The communication of pre-recorded or live broadcast video is unidirectional and follows a client/server paradigm. This unidirectional video communication is typically implemented by one of the following client/server application pairs:

- Flash Player web browser plugin / Adobe Flash Media Server
- Microsoft Silverlight web browser plug in / Windows Media Server
- Quicktime Player / Quicktime Server
- Real Player / Real Helix Server
- Windows Media Player / Windows Media Server

Interactive live video is a bidirectional video communication that follows a peer-to-peer paradigm. This bidirectional video communication is typically implemented by one of the following peer-to-peer applications:

- Yahoo Messenger
- Skype

Understanding the network protocols used for video communication is equivalent to understanding the protocols used by the video communication applications. We conducted a study to observe the protocols used for both client/server and peer-to-peer video communication.

Sources of pre-recorded and live broadcast video (i.e., client/server video communication) were found by using the Google search engine with the following search strings:

- "video streaming sites"
- "online video"
- "live video streaming"

Sources that contained adult content were excluded from the results. In addition, sources were narrowed down to a per-site selection rather than per-video, excluding multiple videos from the same site. Live interactive video was generated by using peer-to-peer applications within our laboratory.

Video frames are typically much larger than the maximum size packet. Therefore, a video frame is segmented into several maximum size packets and a single packet with the remainder. Each packet contains overhead for network delivery. Additionally, the entire video frame may contain overhead. Figure 1 illustrates the process in which a video frame is segmented for delivery over the Internet.

For each source of video, we observed: 1) the application used, and 2) the protocols used. We used Wireshark [5] to capture packets that allowed us to observe the protocols that were being used for video communication. Table I summarizes our findings for pre-recorded video, Table II summarizes our findings for live broadcast video, and Table III summarizes our findings for live interactive video.

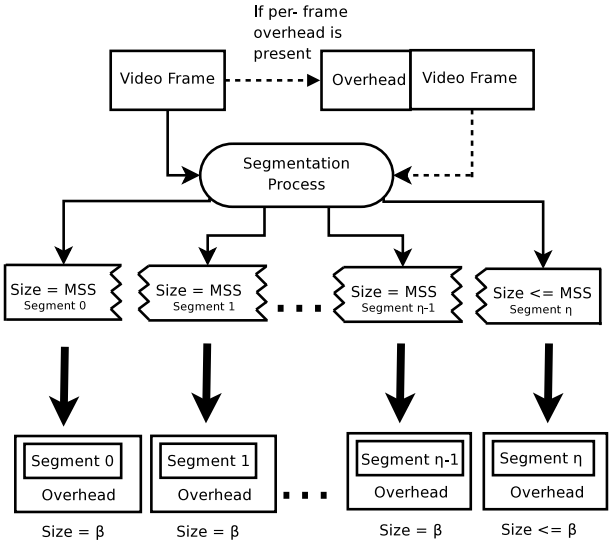


Figure 1. The video frame is *optionally* wrapped in once per video frame network protocol overhead, $O_{vf}$. The video frame is then segmented into $\eta$ pieces, each less than or equal to a maximum segment size, $\beta$. Finally, each of these segments are wrapped in network protocol overhead, $O_{pkt}$, for transport over the Internet.

It is clear by looking at both Tables I and II, that Adobe Flash is, by far, the most widely used application for communicating both pre-recorded and live broadcast video. The popular Netflix (pre-recorded video) defies that trend and uses Microsoft Silverlight. In the next few subsections, we discuss the protocols used by the various video communication applications starting with the popular Adobe Flash.

### A. Adobe Flash

Adobe Flash supports the following modes of communication of pre-recorded video:

- progressive download using HTTP
- streaming using a protocol called Real Time Messaging Protocol (RTMP)

Figure 2 illustrates RTMP which is quite similar to the general process illustrated in Figure 1. In the RTMP documentation, video frame segments are referred to as "chunks". The first "chunk" of the first video frame has its own header; Chunk Message Header Type 0. The first "chunk" of every other video frame; Chunk Message Header Type 1. The second "chunk" of each video frame; Chunk Message Header Type 2. Finally, all of the remaining "chunks"; Chunk Message Header Type 3. Table IV lists the fields in the per video frame header. Table V lists the fields in the different Chunk Message Header Types.

Table I
NETWORK PROTOCOLS USED TO COMMUNICATE PRE-RECORDED VIDEO. WHEN HTTP WAS USED, WE RECORDED THE HTTP HEADER SIZE.

| Source | Application | Protocols | HTTP Overhead (in bytes) |
|---|---|---|---|
| Academic Earth | Flash | HTTP/TCP/IP | 310 |
| Berkeley Webcast | Flash | HTTP/TCP/IP | 332 |
| BYU School of Education | Flash | HTTP/TCP/IP | 257 |
| CBS Video | Flash | RTMP/TCP/IP | |
| Center For Social Media | Flash | HTTP/TCP/IP | 230 |
| CNN Video | Flash | HTTP/TCP/IP | 273 |
| College Humor | Flash | HTTP/TCP/IP | 235 |
| Dog Training Videos | Flash | HTTP/TCP/IP | 218 |
| Edutopia | Flash | HTTP/TCP/IP | 366 |
| eHow Videos | Flash | HTTP/TCP/IP | 224 |
| ESPN Video | Flash | HTTP/TCP/IP | 253 |
| Folk Streams | Flash | HTTP/TCP/IP | 290 |
| Google Video | Flash | HTTP/TCP/IP | 309 |
| Howcast | Flash | HTTP/TCP/IP | 354 |
| Hulu | Flash | HTTP/TCP/IP | |
| Internet Archive | Flash | HTTP/TCP/IP | 277 |
| MSN (Bing) Video | Flash | HTTP/TCP/IP | 347 |
| National Geographic Video | Flash | HTTP/TCP/IP | 359 |
| Nature Online Streaming Archive | Flash | HTTP/TCP/IP | 258 |
| NBC Digital Health Network | Flash | RTMP/TCP/IP | |
| Teacher Tube | Flash | HTTP/TCP/IP | 212 |
| Veoh | Flash | HTTP/TCP/IP | 418 |
| Yahoo! Video | Flash | HTTP/TCP/IP | 262 |
| Youtube | Flash | HTTP/TCP/IP | 309 |
| C-Span | Flash | RTMP/TCP/IP | |
| Channel One | Flash | RTMP/TCP/IP | |
| Discovery Channel | Flash | RTMP/TCP/IP | |
| Disney Online | Flash | RTMP/TCP/IP | |
| EASE History | Flash | HTTP/TCP/IP | 316 |
| Fox News | Flash | RTMP/TCP/IP | |
| History Channel Video Guide | Flash | RTMP/TCP/IP | |
| PBS Video | Flash | RTMP/TCP/IP | |
| Teacher's TV | Flash | RTMP/TCP/IP | |
| TV Guide Video | Flash | RTMP/TCP/IP | |
| Wisconsin Public TV | Flash | RTMP/TCP/IP | |
| Fora.tv | Flash | RTMP/TCP/IP | |
| NBC Video | Flash | RTMP/TCP/IP | |
| Vevo | Flash | HTTP/TCP/IP | 310 |
| Berkeley Webcast | Real | RDT/UDP/IP | |
| PBS The Elegant Universe | Real | RDT/UDP/IP | |
| The Archeology Channel | Real | RDT/UDP/IP | |
| University of Michigan Business School | Real | RDT/UDP/IP | |
| Howard Hughes Medical Institute | Windows Media | HTTP/TCP/IP | 268 |
| Netflix | Silverlight (Windows Media) | HTTP/TCP/IP | 331 |
| The Archeology Channel | Windows Media | HTTP/TCP/IP | 486 |
| AIPCA | Windows Media | RTP/TCP/IP | |
| Apple Movie Trailers | Quicktime | HTTP/TCP/IP | 302 |
| Howard Hughes Medical Institute | Quicktime | HTTP/TCP/IP | 278 |
| PBS The Elegant Universe | Quicktime | HTTP/TCP/IP | 344 |

Table II
NETWORK PROTOCOLS USED TO COMMUNICATE LIVE BROADCAST VIDEO.

| Live Video Content | Application | Protocols |
|---|---|---|
| C-Span Live | Flash | RTMP/TCP/IP |
| EarthTV Live | Flash | RTMP/TCP/IP |
| LiveVideo | Flash | RTMP/TCP/IP |
| Stickam | Flash | RTMP/TCP/IP |
| UStream.tv | Flash | RTMP/TCP/IP |
| C-SPAN Live | Real | RDT/UDP/IP |
| CamStreams | Windows Media | HTTP/TCP/IP |

Table III
NETWORK PROTOCOLS USED FOR LIVE INTERACTIVE VIDEO COMMUNICATION.

| Application | Protocols |
|---|---|
| Yahoo Messenger | TCP/IP |
| Skype | UDP/IP |

### *B. Windows Media, QuickTime, and Real Media*

Windows media uses RTP or HTTP, Quicktime uses HTTP, and Real media uses a protocol called RDT. Unfor-

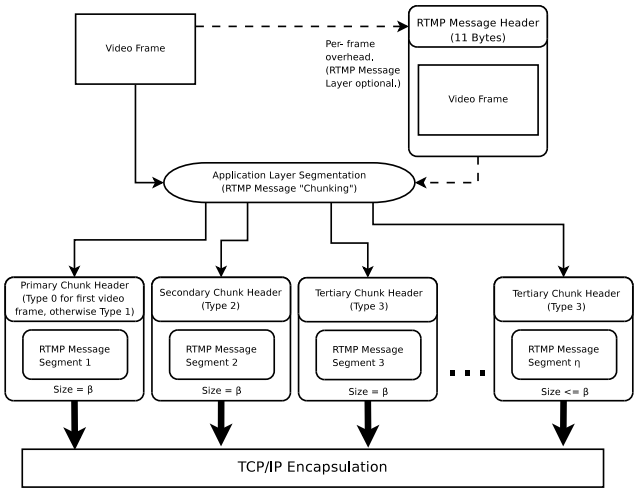


Figure 2. RTMP is a protocol above TCP that operates similarly to the general process presented in Figure 1. However, the per segment overhead is dependent on the segment number. Segments 1 and 2 have a unique RTMP header and segments 3 through $\eta$ have the same RTMP header.

tunately, the RDT protocol from Real Media is proprietary and undocumented. We will model Real Media using the protocol layers below RDT.

### C. Yahoo! Messenger, and Skype

Yahoo! messenger uses an undocumented proprietary protocol over TCP, and Skype uses an undocumented proprietary protocol over UDP. We will model Yahoo! messenger and Skype using the protocol layers below their undocumented protocols.

Table IV
RTMP PER VIDEO FRAME OVERHEAD FIELDS WITH SIZES IN BYTES [9], [10].

| Field | Size |
|---|---|
| Message Type | 1 |
| Payload Length | 3 |
| Timestamp | 4 |
| Stream ID | 3 |
| **Total** | 11 |

## III. MODELING PROTOCOL OVERHEAD

We now use the data we collected in Tables I, II, and III to develop analytical models of the protocol overhead. The structure of the analytical models follow from the illustration in Figure 1.

Let $V$ be a random variable representing the size of a video frame (in bytes), $\beta$ be the maximum size of a packet (in bytes), $\eta$ be the number of packets per video frame, $O_{pkt}$ be the per packet overhead (in bytes), $O_{vf}$ be the once per video frame overhead (in bytes), and $O$ be the total overhead per video frame (in bytes). The total overhead per video frame is expressed as:

Table V
RTMP PER PACKET OVERHEAD FIELDS WITH SIZES IN BYTES [11]. MANY OF THE FIELDS ARE VARIABLE LENGTH. HOWEVER, IN PRACTICE WE ONLY FOUND THE HIGHLIGHTED SIZES IN USE.

| Field | Size |
|---|---|
| **Chunk Message Header Type 0** | |
| Format and Stream ID | {**1**,2,3} |
| Timestamp | 3 |
| Message Length | 3 |
| Message Type ID | 1 |
| Message Stream ID | 4 |
| Extended Timestamp | {**0**,4} |
| **Total** | [**12**,18] |
| **Chunk Message Header Type 1** | |
| Format and Stream ID | {**1**,2,3} |
| Timestamp Delta | 3 |
| Message Length | 3 |
| Message Type ID | 1 |
| Extended Timestamp | {**0**,4} |
| **Total** | [**8**,14] |
| **Chunk Message Header Type 2** | |
| Format and Stream ID | {**1**,2,3} |
| Timestamp Delta | 3 |
| Extended Timestamp | {**0**,4} |
| **Total** | [**4**,10] |
| **Chunk Message Header Type 3** | |
| Format and Stream ID | {**1**,2,3} |
| **Total** | [**1**,3] |

Table VI
THE OVERHEAD FOR VARIOUS NETWORK PROTOCOLS (IN BYTES)

| Protocol | Overhead Variable | Value |
|---|---|---|
| HTTP | $O_H$ | Site dependent |
| RTMP Message | $O_M$ | 11 |
| RTMP Chunk 0 | $O_{C0}$ | 12 |
| RTMP Chunk 1 | $O_{C1}$ | 8 |
| RTMP Chunk 2 | $O_{C2}$ | 4 |
| RTMP Chunk 3 | $O_{C3}$ | 1 |
| RTP | $O_R$ | 12 |
| TCP | $O_T$ | 20 |
| UDP | $O_U$ | 8 |
| IP | $O_I$ | 20 |
| Ethernet | $O_E$ | 18 |

$$O = \eta \cdot O_{pkt} + O_{vf} \tag{1}$$

The number of packets per video frame is expressed as:

$$\eta = \left\lceil \frac{V}{\beta} \right\rceil \tag{2}$$

Combining equations 1 and 2 yields:

$$O = \left\lceil \frac{V}{\beta} \right\rceil \cdot O_{pkt} + O_{vf} \tag{3}$$

Table VI shows the size of the overhead for each protocol.

### A. Adobe Flash

For *progressive mode*, HTTP is used. Therefore, $\beta$ is the TCP Maximum Segment Size (MSS) and the HTTP

overhead is site dependent, $O_{pkt} = O_T + O_I = 40$, and $O_{vf} = O_H$. Let $\beta_T$ be the TCP MSS,

$$O = 40 \cdot \left\lceil \frac{V}{\beta_T} \right\rceil + O_H \quad (4)$$

For *streaming mode*, RTMP is used over TCP. Therefore, $\beta$ = TCP MSS, $O_{pkt} = O_{C1} + O_T + O_I = 48$ for the first segment, $O_{pkt} = O_{C2} + O_T + O_I = 44$ for the second segment, and $O_{pkt} = O_{C3} + O_T + O_I = 41$ for all remaining segments; $O_{vf} = O_M = 11$. This yields,

$$O = 103 + 41 \cdot \left\lceil \frac{V - 2\beta_T}{\beta_T} \right\rceil \quad (5)$$

### B. Windows Media

When Windows media is used with RTP, $\beta$ = UDP MSS, $O_{pkt} = O_R + O_T + O_I = 52$, and $O_{vf} = 0$. Let $\beta_U$ be the UDP MSS,

$$O = 52 \cdot \left\lceil \frac{V}{\beta_U} \right\rceil \quad (6)$$

When Windows media is used with HTTP. Therefore, $\beta$ = TCP MSS and the HTTP overhead size is site dependent, $O_{pkt} = O_T + O_I = 40$, and $O_{vf} = O_H$. This yields,

$$O = 40 \cdot \left\lceil \frac{V}{\beta_T} \right\rceil + O_H \quad (7)$$

### C. Quicktime

Quicktime uses HTTP. Therefore, $\beta$ = TCP MSS and the HTTP overhead is site dependent, $O_{pkt} = O_T + O_I = 40$, and $O_{vf} = O_H$. This yields,

$$O = 40 \cdot \left\lceil \frac{V}{\beta_T} \right\rceil + O_H \quad (8)$$

### D. Real Media

Real media uses a proprietary protocol called RDT. We were unable to find any documentation of this protocol, so we will only model the layers below this protocol. Therefore, $\beta$ = UDP MSS, $O_{pkt} = O_U + O_I = 28$, and $O_{vf} = 0$. This yields,

$$O = 28 \cdot \left\lceil \frac{V}{\beta_U} \right\rceil \quad (9)$$

### E. Yahoo! Messenger and Skype

Yahoo! Messenger uses an undocumented protocol over TCP. Therefore, we will only model TCP and the lower layers: $\beta$ = TCP MSS, $O_{pkt} = O_T + O_I = 40$, and $O_{vf} = 0$. This yields,

$$O = 40 \cdot \left\lceil \frac{V}{\beta_T} \right\rceil \quad (10)$$

Skype uses an undocumented protocol over UDP. Therefore, we will only model UDP and the lower layers: $\beta$ = UDP MSS, $O_{pkt} = O_U + O_I = 28$, and $O_{vf} = 0$. This yields,

$$O = 28 \cdot \left\lceil \frac{V}{\beta_U} \right\rceil \quad (11)$$

## IV. Network Simulation Tools

Using the observations from our video traffic protocol encapsulation study we have designed tools to use video trace data [12] in OMNeT++ network simulations. Figure 3 illustrates how we convert video trace files to packet trace files and then use these to generate packets in an OMNeT++ network simulation. All of the tools described can be found, along with documentation, at the following URL: http://ee.utep.edu/mcgarry/UTEP_mcgarry/Tools.html.

### A. Video Traces to Packet Traces

*pkttrace* is a Python script that produces a packet trace from a video trace. Each video frame represented in the video trace is segmented using a specified maximum packet size and overhead bytes are added to each packet according to the specified protocol. Command line options, that are described in the documentation of *pkttrace*, are used to specify the maximum packet size and protocols to be used. *pkttrace* supports all of the protocols found in our study.

### B. Packet Traces to OMNeT++ Packets

*VideoTrafficGen* is an OMNeT++ module that reads packet information from the packet trace file specified by the "traceFile" module parameter. *VideoTrafficGen* is implemented using an activity() function that reads the packet trace file packet by packet (i.e., line by line). For each packet in the packet trace file, a cMessage object is created. cMessage objects are commonly used by the networking research community to represent packets so we follow the trend in our design. *VideoTrafficGen* expires the appropriate amount of simulation time between packets with different timestamps. The following data are set in each cMessage object: the timestamp of the packet (via setTimestamp()), the length of the packet (via setByteLength()), and the type of video frame data contained in the packet (via setKind()). The type of video frame data is 1 for an I frame, 2 for a P frame, and 3 for a B frame.

## V. Conclusion

For a vast majority of the sources of video content found on the Internet we have observed and recorded both the applications and protocols used for video communication. With this information we developed analytical models of the protocol overhead for the different video communication applications. These models can be used by multimedia networking researchers to accurately characterize the overhead for communicating video over the Internet. Further, we have developed simulation tools that use these models to facilitate discrete event simulation of video traffic sources in

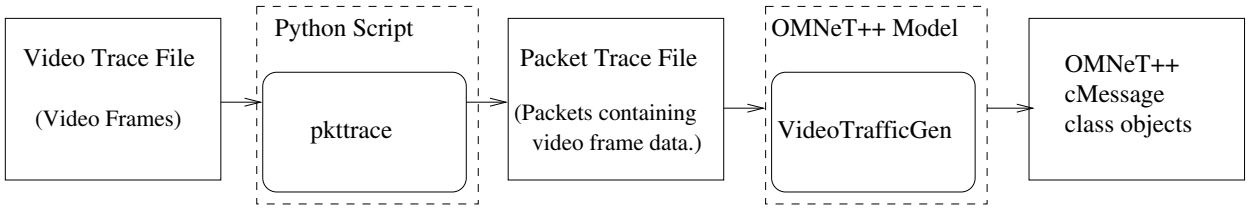


Figure 3. The process of converting video traces into sequences of packets modeled as OMNeT++ cMessage objects in an OMNeT++ simulation. The solid boxes represent inputs and outputs.

OMNeT++ [6]. We are making these tools publicly available at the URL listed in Section I.

Extending our study to IPTV networks such as AT&T's U-verse would be beneficial to allow accurate modeling of broadcast and on demand video delivered over these private packet switched networks. Since these IPTV networks are private networks, coordination with the service provider would be required to obtain the necessary data for our study.

## REFERENCES

[1] M. Li, M. Claypook, R. Kinicki, and J. Nichols, "Characteristics of streaming media stored on the web," *ACM Transactions on Internet Technology*, vol. 5, no. 4, pp. 601–626, November 2005.

[2] Cisco, "Approaching the zettabyte era," *Cisco Visual Networking Index*, 2008.

[3] F. H. Fitzek, P. Seeling, and M. Reisslein, "Using network simulators with video traces," Arizona State University, Tech. Rep., 2003. [Online]. Available: http://trace.eas.asu.edu/publications/tracesim.pdf

[4] F. H. Fitzek and M. Reisslein, "Mpeg-4 and h.263 video traces for network performance evaluation," *Network, IEEE*, vol. 15, no. 6, December 2001.

[5] "Wireshark foundation," http://www.wireshark.org.

[6] "Omnet++ simulation library," http://omnetpp.org.

[7] M. Johanson, "An rtp to http video gateway," in *WWW '01: Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 499–503.

[8] R. D'Haenens, J. Doggen, D. Bakker, and T. Dams, "Transmitting scalable video with unequal error protection over 802.11b/g," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2008. WIMOB '08.*, Oct. 2008, pp. 638 –643.

[9] "Real time messaging protocol (rtmp) message formats," Adobe Systems inc., Tech. Rep., June 2009.

[10] "Video file format specification version 10," Adobe Systems inc., Tech. Rep., November 2008.

[11] "Real time messaging protocol chunk stream," Adobe Systems inc., Tech. Rep., June 2009.

[12] "Arizona state university video trace library," http://trace.eas.asu.edu.