

# An Example of a Simple Cache system for a Video Streaming Implementation within a Network Simulation

Patrick Mulumba, Peter Clayton and George Wells

**Abstract**—The modern expansion of public streaming has played a considerable part in the growth of distributed multimedia, with special reference to video. However, since the Internet is an inter-connected web of different network speeds, streaming video is still a difficult service to implement to all locations. This paper illustrates two simple simulated network streaming environments. The first models a broadcast network and the second model represents a multicast network. A simple cache system is then implemented into these models to investigate whether a significant difference exists between the cache size and the different network bandwidths of these two optimization systems. This caching system will then be used to compare the impact of videos (characterised by different resolutions and degrees of picture motion) on the cache performance.

**Index Terms**—Network simulation, Cache, Multicasting, Video streaming, NS-2, Distributed Multimedia, Evalvid,

## I. INTRODUCTION

Distributed media streaming, over the past few years, has grown significantly in terms of its functional capabilities. This growth was driven by a combination of an increase in the availability of bandwidth and improved media compression techniques. These advances have led to the development of various streaming frameworks, which currently provide the majority of the streaming media available throughout the Internet.

However, the Internet is a vast inter-connected web of various network configurations with different bandwidths. This has led to a complex field for media streaming to be deployed, especially with reference to streaming video. Jitter, lost and dropped packets and the non-deterministic nature of the Internet are just some of the numerous problems faced by the streaming video environment. Various techniques and strategies have been presented to combat the errors, be it from various configurations to mathematical algorithms that either enhance the signal strength or limit the probability of a particular signal being lost. Network simulation has also been a successful tool in determining problems in various network deployments before actual physical implementations have been installed.

This paper analyses two streaming video media distribution scenarios within a simulated network environment. The first scenario consists of a single server distributing a broadcast stream to multiple clients (broadcasting), whilst the other scenario has the server send a stream which is publicised

This work was partly sponsored by the Distributed Multimedia Centre of Excellence at Rhodes University, with sponsorship from Telkom SA, Business Connection, Comverse, Amatole Telecommunication, Mars Technologies, Openvoice, Storetech, Tellabs, THRIP and with financial support from National Research Foundation.

to a particular “mask” which clients can join (multicasting). This paper attempts to demonstrate the application of a simple caching system to these scenarios. Following this, this paper determines whether a significant difference exists between the cache size and the different network bandwidths of these two optimization techniques (multicasting and broadcasting) for a given simulated video streaming network environment. This caching system then compares the impact of videos (characterised by different resolutions and degrees of picture motion) on the cache performance.

## II. BACKGROUND

The Internet has become an ever evolving tool for communication, characterised by various formats of available media. With a continuously increasing demand for an efficient means of communication, video is becoming a primary source of relaying information to peers, the public and other recipients. Traditional downloading was the most common way of distributing this media. However due to the need for easy content control, the provision of real-time video and audio and the prevention of piracy, an alternative means of distribution has become necessary to relay this media to the public.

Streaming Media is a suitable alternative to traditional downloading. It “enables real-time access to media via the Internet or on an Intranet” [1]. It is currently the most popular form of providing live broadcasts to the public [2]. One of its chief advantages is that it removes the need for file downloads, as no file is downloaded during the stream. In order for streaming to take place however, the media is generally played by client software, and in almost all cases, a media server is required to provide the streaming.

Streaming serves as an excellent means for live broadcasts [2] whilst providing some form of copyright protection [3]. However, since the Internet is not designed for real-time streaming, streaming is plagued with the obstacles of limited bandwidth constraints [2], latency of signal transmission [4] and both noise and packet loss [5]. These difficulties make it difficult to deploy streaming media systems into the real environment; hence this project aims to provide a simulated environment which can aid the design and implementation of a streaming media system. Before we can investigate the different streaming deployments however, we must first provide some technical background.

Streaming media can be classified according to three different distribution schemes [2]:

- 1) Live broadcast: involves “encoding content straight from the event ‘live’ to a server to be streamed”.
- 2) On-Demand: files are streamed at the client’s request and are encoded at the source with the opportunity of being accessed at a later stage.
- 3) Progressive Download: consists of a mixture between traditional downloading and streaming. It enables content to be viewed whilst being downloaded, by utilizing temporary memory known as a Buffer. This streaming method is generally used in conditions where the content is short.

This paper explores live broadcasts as the main distribution scheme.

Media streaming encompasses such a large variety of encoding techniques and various algorithms that for the purposes of this paper, we decided to base our common themes on those presented by other academic research papers. We aim to highlight the various strategies and algorithms that are employed in order to overcome these problems or theories.

#### A. Load Management

Bandwidth is one of the important factors in the application of the streaming media [2]. Since the Internet has a multitude of different bandwidths and various connection speeds, it is important for the media to adapt to the different changes in these conditions. During the process of streaming, whether it is a Client-Server scenario or simple Peer-to-Peer scenario, bandwidth is always utilized by the transmission [6]. According to [5], the quality of service control between the receiver and sender is poor due to the Internet’s inability to guarantee the availability of bandwidth between the two ends. Ho & Lee [5] tackled this phenomenon by employing a “predictive buffering” algorithm, which is not utilized from a single sender but from many senders. They concluded that the overall difference between their buffering prediction algorithm and the actual ‘efficient buffer time’ is within a satisfactory range. However, the particular investigation utilised more than one sender, so for the purposes of this paper, the buffering prediction algorithm shall not be incorporated.

#### B. Multicasting

According to [2], streaming media is delivered in three basic transmission methods:

- 1) “Unicasting involves an individual stream between a server and the user” and is most commonly adapted for On-demand delivery of media streaming [2].
- 2) “Broadcasting” is the scenario where a “single stream is transmitted to many users concurrently” with each user having a connection to the server directly [2].
- 3) “Multicasting” is a “single stream that is sent by a special multicast IP-address” onto a multicast network and is viewable by any of clients on the network by joining the broadcast [2].

The last transmission method (multicasting) is the focus of this research and is in essence a more efficient form of broadcasting. Another important factor (in multicasting) is that only a single stream is sent by the server to the router/routers, whilst in the

broadcast scenario each client has a direct connection to the server [7]. However, some streaming environments can have a multicasting distribution directly from the server.

#### C. Distribution and Diversity

Whether there are single or multiple servers providing a media stream, managing both the distribution and the diversity of the stream is part of the foundation of any streaming environment. Conklin et al. [8] discuss the concept of video coding for the delivery of streaming media specifically over the Internet. The primary focus consists of three sections:

- 1) The delivery mechanisms; which can be unicasted or alternatively multicasted.
- 2) The optimization criteria; which consists of minimizing the distortion for the connection rate or minimising the traffic.
- 3) The distribution model of the stream; which can be on-demand or alternatively streamed live.

Although not mentioned in this discussion, an effective optimization strategy that both minimises the distortion and decreases traffic is the use “Multiple Descriptions”. These are “separate bit-streams containing complementary descriptions of a signal stream” [9]. Another “Multiple Description” paper [10], investigates the creation of these descriptions, particularly the compression representations needed for the content delivery of the media streams. “Multiple Descriptions” are a suitable optimisation solution for this problem. However, for this paper we are merely investigating the general implications of a simple caching system, not a complete optimisation algorithm.

#### D. Caching

This performance strategy is a fundamental concept for maintaining quality streaming to users. Caching is essentially storing copies of the streaming data on the local network for future use by other users on the local network [2]. A cache mixture strategy was employed specifically for streaming media files by [11], which uses two simplified streaming caching styles to produce a new hybrid strategy in order to increase the quality of streaming efficiently. It combines “Resource Based Caching (RBC)” which looks at the file size and required bandwidth, with the “Least Frequency Used (LFU)” strategy which caches videos with the highest request frequencies. These two are blended to propose an algorithm based on “Pooled RBC” and a “RBC/LFU Hybrid cache”. A similar strategy was employed by [12] who provided a combination of both “Batched Patch” caching and “Prefix/interval” caching strategies, operating over multicasting environment.

These concepts and strategies represent only a fraction of video streaming, but what this paper aims to do is provide a simplified caching system example. Such a simplified caching system would enable us to effectively examine the cache size in relation to the bandwidth of a simulated transmission. What separates this paper’s investigation from previous work done is the direct examination of simple cache sizes in relation to different bandwidths and at the same time, comparing their performance with regards to video quality. Network simulation will be discussed in the next section.

### III. NETWORK SIMULATION

User Streaming environments are unique to both customer and client needs. In order to effectively examine how media is transmitted we needed to deploy networking environments that represent various streaming scenarios. Since networks can span almost any sort of design, setting up a physical network can be time consuming and very cost ineffective if the desired outcome is not up to the client’s specification. In order to overcome this, we propose using a network simulator as part of our cached streaming distribution analysis tool.

Various simulation tools were available but Ns-2 provides an open-source and cost effective means for representing various networks. Ns-2 is a networking research discrete event simulator [13], written in C++ and interfaced with the “OTcl interpreter” as a front-end [14], [15]. It is currently developed for UNIX systems but can be ported to Windows systems through Cygwin [16]. Simulations are objects which are created through the interpreter in which the interpreter can reproduce these objects using Ns-2’s class hierarchy. Ns-2 uses C++ for simulations to quickly implement the manipulation of bytes, algorithms and packets that span various protocols, whilst OTcl is very quick to make changes to the simulation configuration.

Simulations are comprised of network objects (nodes), the network topology for the layout of the network and the events that govern the flow of traffic. C++ provides the back-end to the simulations, whereas the interface for these events in Ns-2 is prepared by TCL-Script. Upon successful completion of the script, various files are produced that each provide different forms of functionality [14]. The traditional TCL script produced from a given simulation can produce a tracefile. A tracefile records each packet at a link or queue with its send time, arrival time and simple packet information [14]. This file is unprocessed packet data similar to that of a ‘traffic-dump’ from a packet sniffer. For the purposes of this paper, a detailed explanation of tracefiles is explored in section “V” to provide a better understanding of this research as a whole.

### IV. ENVIRONMENT

Ns-2 is a very extensible tool. Various modules and expansions have been made to either improve the internal operations of the environment or enhance the output of the simulation results. As mentioned earlier, tracefiles are the core outcome from Ns-2 simulations. “Nam” is an animation tool that provides a visual representation of an Ns-2 simulation. Nam is an extension of the Ns-2 environment [14] which reads its own tracefile in a similar manner to that of the Ns-2 tracefiles, with the exception being that this tracefile provides detailed information with regards to the topology, the construction of the network and all the events for the simulation. As most of the processing takes place after the simulation has been completed Nam can be referred to a post-processor [14].

Ns-2 is a discrete event network simulator which produces packet information, whilst Nam produces its own animation script which illustrates the network simulation. Ns-2 alone does not provide any network video analysis but an existing

framework is available that can assist in the evaluation of video transmissions through a network.

“Evalvid” is a framework that allows for the evaluation of video quality in transmissions over networks [17]. Its usefulness lies in its ability to cater for both real and simulated networking environments. Evalvid has been used extensively in simulated research, with specific regards to Ns-2. Two publications [18], [17], provide examples of utilising this framework within Ns-2. The second publication provides an amendment to the existing Evalvid set of tools and provides new code (Evalvid-RA) to help community members provide more optimized codecs that can better sustain the various networking conditions of the modern Internet [19]. The following resources [18], [20] provide an in-depth investigation into the integration of video and Ns-2. The latter provides more detailed documentation of integrating Evalvid into Ns-2 for video quality research. This facilitates the design of the simulation environment presented in the next section.

### V. DESIGN

Ns-2 is an open-source network simulation tool which comprises of various components in a class hierarchy to construct simulation objects. To provide quality video analysis simulations, integrating Evalvid into the Ns-2 environment yielded some functions and files that provided the tools for this paper. They are provided in Table. I:

Table I  
EVALVID FILES AND FUNCTIONS

Cygwin	Operating System - Windows Based Linux environment
Myevalvid2	C++ extensions to core simulator to accept new nodes for video support
*.yuv	Uncompressed Raw video files used for testing
*.tcl	Simulation script file: represents an Ns-2 simulation
*.par	Parameter file to compress raw video files into mpeg data format
mpeg4encoder	Compresses the Parameter file from Un-encoded raw yuv format to a file
MP4.exe	Creates a reference tracefile of the video being sent and packetizes the video
sd_trace	The sender trace file generated through the simulation
rd_trace	The receiver trace file generated through the simulation at a node
et.exe	Generates sent compressed video after simulation using all the trace files (sd_trace and rd_trace and reference tracefile) and original compressed video. It then provides information on lost packets
mpeg4decoder	Decompresses the Parameter back to Uncompressed Raw yuv format

This paper investigates a simulated implementation of broadcasting and multicasting within a streaming network environment. The test setup involves using two distribution scenarios. The following represents the common testing specifications for each scenario:

- 1) The server plays a single video stream
- 2) All nodes start to receive the stream at the same time
- 3) All connections between the server and users are equal
- 4) No other traffic is sent during the video transmission
- 5) The video stream is sent from start to finish with no pauses or breaks in between.

The first scenario consists of a single server that provides a stream broadcast to clients that are all connected to it. This demonstration is equivalent to that of a traditional radio broadcast [2] as shown in Fig. 1.

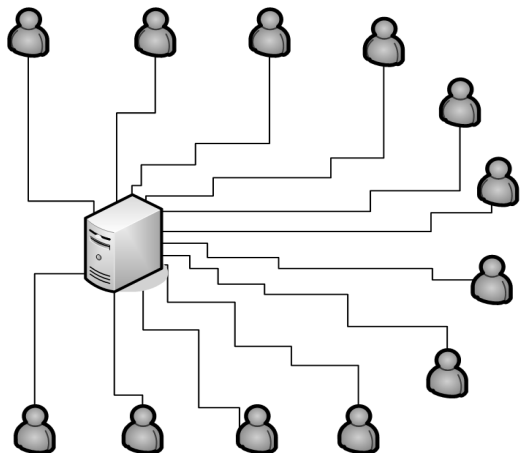


Figure 1. Broadcast simulation topology

The second scenario represents a stream which is being multicasted to various nodes on the network of which any node can effectively join the group [2]. This example is equivalent to that of a conference call [2] as shown in Fig. 2.

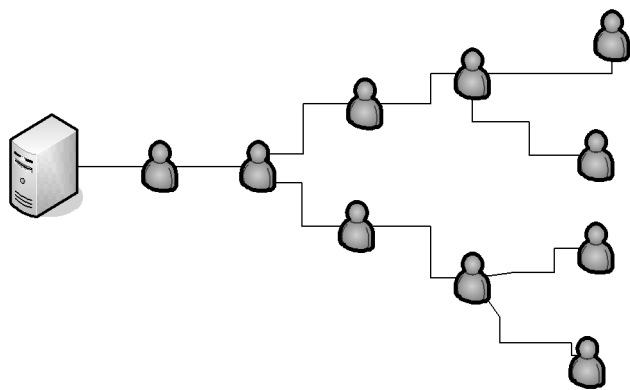


Figure 2. Multicast simulation topology

Broadcasting does not necessarily have to fit the traditional star topology. Broadcasting can be implemented across a normal tree topology, but the resultant communication is still effectively the same. The advantage of multicasting is that clients can opt into the “multicast address” as they choose.

These scenarios consist of a single source node and destination nodes (of which each destination can forward the stream to another node as is the case with multicast scenario). Since this paper seeks to investigate a simplified cache architecture, the cache placement for each scenario consists of a cache size variable for each receiving node. This cache size variable represents the minimum number of cached packets needed to maintain the same streaming video quality as the original stream. This shall be further explained in the Section VII. With

every experiment, there are various conditions and limitations that govern the boundaries of the simulation. This paper seeks to define them in the next section.

## VI. LIMITATIONS

Various limitations are imposed on the both scenarios to help provide a simple analysis into the investigation. The following conditions are enforced upon the whole experiment:

- 1) The number of nodes in each scenario is a representation of an arbitrary network.
- 2) All videos are obtained from an online controlled library of research [21]
- 3) All High quality videos conform to a 352x288 CIF Standard in a YUV 4:2:0 format
- 4) All Low quality videos conform to a 176x144 QCIF Standard in a YUV 4:2:0 format
- 5) All videos are uniform in frame length with 300 frames
- 6) All videos last ten seconds long with an initial frame rate of 30 Frames per second, which stays the same
- 7) The simulation assumes that nodes are connected with physical links and that wireless connections exhibit the same characteristics than that of a physical network
- 8) Full wireless implementation is available in Ns-2, but this paper assumes the pre-stated conditions
- 9) This paper only investigates compressed MPEG video, but other codecs are available from Evalvid

## VII. TESTING

The testing consists of six videos. These videos are grouped into three categories of which each category represents the degree of movement in the video. Within each of these categories is the same video encoded at a high quality standard and at a low quality standard. The quality of the video is in direct relation to the video resolution. The first video (named “Akiyo”) represents a news cast where the movement within the video is relatively low. This view of the Akiyo video as being relatively invariant can be justified by the fact that changes from frame to frame are not too dramatic. The Foreman video represents a construction worker supervising a construction site, where the movement in this video is significantly greater than that of the first video. Here, there is a higher degree of change between frames. The Hall video represents small movement within an office hall. Here there is little change to the overall movement of the scene. The videos are hence referred to as Akiyo HQ/LQ, Foreman HQ/LQ and Hall HQ/LQ.

This paper also seeks to investigate the impact of various networking bandwidths on these videos and cache values. The bandwidths used in the investigation consist of the most common available bandwidth values which are as follows:

- 1) 9.6 Kb/s, 14.4Kb/s, 28.8Kb/s 33.6Kb/s/ and 56Kb/s represent traditional dial-up methods
- 2) 64 Kb/s and 128 Kb/s represent the ISDN Variants
- 3) 256Kb/s, 384 Kb/s, 512K/s and 1000Kb/s represents the ADSL standards
- 4) 10000Kb/s represents old LAN configurations.

From these network speeds, we hope to determine the outcome of a video stream simulation with their respective

cache ratio's in comparison to the different qualities of video and different network bandwidths.

For the broadcast and multicast scenarios, this paper investigates the impact of the different network bandwidths on each of the stipulated videos and the cache size needed to compensate for the loss of quality within each of those bandwidths. In order to simulate the various conditions, the following Algorithm is simulated on each of the scenarios

---

**Algorithm 1** Testing Algorithm

---

```

for each scenario do {
  for each video do {
    for each bandwidth do {
      Stream video from server to clients
      log the lost packets
      Add suitable cache for the loss
      Store the scenario, video
      Store the bandwidth, cache size
    }
  }
}

```

---

The above algorithm is implemented into the system by utilising the various tools and functions obtained from the integration of Evalvid into Ns-2. All the videos are compressed from yuv format to a compressed file using the mpeg4encoder.exe program. A stream video file reference is generated on the compressed file using the MP4.exe. For each of the two scenarios, OTcl simulation scripts are created. Within each simulation script, the respective topology is implemented using its various methods and classes. A simple Algorithm for the simulation scripts follows below:

---

**Algorithm 2** Broadcast.tcl / Multicast.tcl

---

```

Constants: Bndwidth, Delay, Packet_sze, Cache_sze
Enable multicast option {Multicast}
Create nodes
Create links from server to all nodes {broadcast}
Create links from server to first node {Multicast}
Link sender trace file {sd_trace} to server
  Setup source traffic from the server node
  Setup node traffic ends
Link receiver trace file {rd_trace} to each node
Obtain video info from compressed file
  Apply video info to traffic stream
Set start actions for the simulation
Run the simulation

```

---

Both of these simulation scripts are then run for each of the six videos and at the same time they are simulated using each of the different network bandwidths. Separate scripts are generated to iterate over various steps. By incorporating the et.exe function, we could collect the lost packets yielded as result of the lower bandwidths.

As mentioned in section "V", a necessary packet cache size is needed to compensate for the streaming video quality loss incurred. This value is calculated by systematically repeating the simulation and adjusting this cache size so that the number of lost packets decreases as close to zero as possible. This cache size is then implemented on the appropriate nodes and the simulation is verified with the appropriate Evalvid quality

test. The information yielded us the following graphs in Fig.3 and Fig.4:

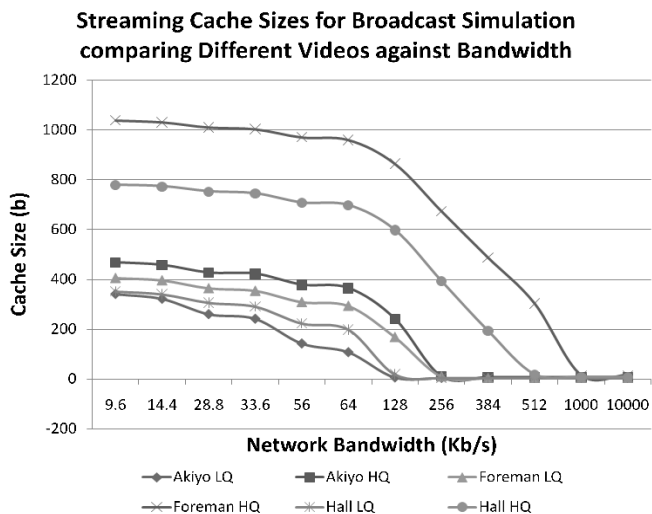


Figure 3. Broadcast test results

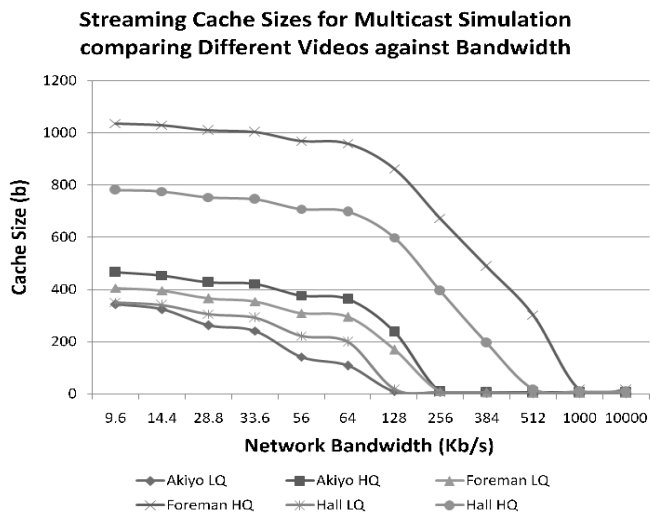


Figure 4. Multicast test results

From these graphs, it is evident that there is a correlation between the cache size, the bandwidth and the video quality. An analysis of these graphs yielded us the following results identified in the following conclusion:

VIII. CONCLUSION

This paper set out to analyse streaming media distribution scenarios for broadcasting and multicasting within a simulated network environment. A simple caching system was then applied to the scenarios in order to determine if there is a significant difference between the cache size in relation to the network bandwidths of these two optimization techniques (multicasting and broadcasting). This was conducted within the given simulated streaming network environment. At the same time, this caching system was used to compare the impact of

videos (characterised by different resolutions and degrees of picture motion) on the cache performance.

A caching system was successfully implemented into a simulated video streaming environment in which it demonstrates that a suitable deployment of a caching system can provide a successful video stream between two ends within the stipulated network bandwidth range. The implementation of the two different scenarios within the Ns-2 environment demonstrates that a multicast network can provide an efficient means of broadcasting with an insignificant drop in performance. From the observations noted, a correlation between the network bandwidth ranges and the size of the cache was identified. Finally, both the degree of motion present within the video and its resolution play a significant part in the production of the distribution stream.

As streaming media provides an alternative form of media distribution to traditional downloading, we note a need for the improvement of this method. This paper shows that the application of an appropriate caching system to a simulated video streaming environment can greatly enhance the quality in an efficient manner. This may also be applicable to a real world example and could provide a means to improve the quality of videos streamed on the Internet, and thereby advancement to media streaming as we know it today.

#### A. Possible Extensions

This paper details a simulated video streaming network implementation. An interesting adaptation is to compare a simulated network to a physical network and draw a comparison between the simulated environment and a real environment. A further extension can be the implementation of various other complex caching algorithms. A comparison between these algorithms in correlation to the bandwidths can again also be investigated. More investigation into the use of different codecs can yield more complex scenarios where different codecs might be better suited for different bandwidth conditions.

#### REFERENCES

- [1] Adobe, "A streaming media primer," Adobe Dynamic Media Group, 2000. [Online]. Available: <http://www.adobe.com/products/aftereffects/pdfs/AdobeStr.pdf>
- [2] S. Mack, *Streaming Media Bible*. Hungry Minds, Inc., 2002. [Online]. Available: <http://www.streamingmediabible.com/html/toc.html>
- [3] E. Shumacher-Rasmussen, "Make your content count," *Streaming Media Inc: Innovation Series*, vol. Best Practices: Media, Broadcast & Entertainment, no. 5, pp. 2–3, July 2006. [Online]. Available: <http://www.streamingmedia.com/whitepapers/>
- [4] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik, "Scalable distributed stream processing," in *CIDR 2003 - First Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, January 2003.
- [5] P. Y. Ho and J. Y. B. Lee, "Predictive buffering for multi-source video streaming over the internet," in *Global Telecommunications Conference*, San Francisco, CA, USA: The Chinese University of Hong Kong, November 2006, pp. 1–6.
- [6] C. Krasic, "A framework for quality-adaptive media streaming: Encode once – stream anywhere," Ph.D. dissertation, OGI School of Science & Engineering, Oregon Health & Science University, February 2004.
- [7] M. Savic, "Enterprise video streaming applications and products," *Streaming Media inc: Innovation Series*, vol. Solutions For Enterprise Streaming & Digital Media, no. 3, pp. 18–19, July 2004. [Online]. Available: <http://www.streamingmedia.com/whitepapers/>

- [8] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video coding for streaming media delivery on the internet," in *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 11, no. 3, 281, March 2001, p. 269.
- [9] J. G. Apostolopoulos and M. D. Trott, "Path diversity for enhanced media streaming," HP - Mobile and Media Systems Laboratory, HP Laboratories Palo Alto, Tech. Rep., July 2004.
- [10] V. K. Goyal, "Multiple description coding: Compression meets the network," *IEEE SIGNAL PROCESSING MAGAZINE*, pp. 74–93, September 2001.
- [11] J. M. Almeida, D. L. Eager, M. Ferris, and M. K. Vernon, "Provisioning content distribution networks for streaming media," University of Wisconsin and University of Saskatchewan, December 2002.
- [12] P. Frossard and O. Verscheure, "Batched patch caching for streaming media," in *IEEE COMMUNICATIONS LETTERS*, vol. 6, no. 4, April 2002, pp. 159–161.
- [13] P. K. Choudhary and K. S. Trivedi, "Discrete event simulation: Tools and applications," Center for Advanced Computing and Communication Department of Electrical and Computer Engineering Duke University, October 2004.
- [14] K. Fall and K. Varadhan, *The ns Manual (formerly ns Notes and Documentation)*, The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC., October 2005.
- [15] P. Haldar and X. Chen, "Ns tutorial 2002," November 2002.
- [16] S. Chamberlain, "Gnu + cygnus + windows = cygwin," Cygnus Solutions and Red Hat, October 2007. [Online]. Available: <http://cygwin.com/>
- [17] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid - a framework for video transmission and quality evaluation," in *13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Urbana, Illinois, USA: Technical University of Berlin, Telecommunication Networks Group (TKN), September 2003.
- [18] K.-L. Kao, C.-H. Ke, and C.-K. Shieh, "An advanced simulation tool-set for video transmission performance evaluation," in *ACM International Conference Proceeding Series*, P. from the 2006 workshop on ns-2: the IP network simulator, Ed., vol. 202, no. 10. ACM New York, USA, 2006. [Online]. Available: <http://portal.acm.org/citation.cfm?id=1190464>
- [19] A. Lie and J. Klaue, "Evalvid-ra: trace driven simulation of rate adaptive mpeg-4 vbr video," *Multimedia Systems*, vol. 14, no. 1, pp. 33–50, June 2008. [Online]. Available: <http://www.springerlink.com/content/e5557223016234kv/fulltext.pdf>
- [20] K. Chih-Heng, "How to evaluate mpeg video transmission using the ns2 simulator," Kinmen Institute of Technology Information Management, Taiwan, May 2006. [Online]. Available: [http://hpds.ee.ncku.edu.tw/~smallko/ns2/Evalvid\\_in\\_NS2.htm](http://hpds.ee.ncku.edu.tw/~smallko/ns2/Evalvid_in_NS2.htm)
- [21] U. Arizona State, "Video traces research group," Arizona State University, 12 2007. [Online]. Available: <http://trace.eas.asu.edu/index.html>

**P. Mulumba** is with Rhodes University currently doing an MSc research in Distributed Multimedia, This author completed both his undergraduate and Honours years at Rhodes University, and his schooling at St Andrews in Grahamstown. The author's research interests are distributed media, networking, and Media compression techniques, (e-mail: [g02m2469@campus.ru.ac.za](mailto:g02m2469@campus.ru.ac.za)).

**P.G Clayton** is the Deputy Vice Chancellor: Research and Development at Rhodes University. He was formerly the head of the Telkom CoE (Centre of Excellence) in Distributed Multimedia at Rhodes University. He recently received a National Science & Technology Forum award for Outstanding Contributions to Science, Engineering and Technology, (e-mail: [p.clayton@ru.ac.za](mailto:p.clayton@ru.ac.za)).

**G.C Wells** is an associate professor with Rhodes University, Grahamstown. He is the Head of Department of Computer Science at Rhodes University, and teaches programming language-related courses in the Department. He completed his PHD degree in Parallel and Distributed Computing at the university of Bristol. He is also a Sun-certified Java Programmer, (e-mail: [g.wells@ru.ac.za](mailto:g.wells@ru.ac.za)).