

The suitability of P2P for live video streaming

Jeff Hinds

Telkom SA Ltd (Technical Product Development)
Telephone: (012) 311-4743 Email: hindsja@telkom.co.za

Abstract—This paper discusses peer-to-peer technologies with the focus on video streaming. The primary objective is to determine the suitability of peer-to-peer networking as a platform for live video streaming by assessing available alternatives and developing a reference implementation. This implementation provides a platform for analysis and simulation as a means to determine potential gains derived by implementing video streaming using a peer-to-peer network. Peer-to-peer technology brings the capability to stream live video to any regular PC thus allowing almost any Internet connected computer to become a broadcast station with an almost unlimited number of viewers or clients. An analysis of available codecs was carried out and showed that MPEG-4 was the most optimal codec. The reference implementation showed that a relatively simple peer-to-peer streaming system can be developed without the need for an extensive protocol or overlay network.

Index Terms—Peer-to-peer, real-time, live, streaming, video, multimedia, peercasting, video-on-demand

I. INTRODUCTION

PEER-to-peer is a concept which is now well known and embedded in Internet technology and the minds of Internet designers. Peer-to-peer based technologies include DNS¹ and IRC² as well as a plethora of file sharing protocols and applications such as KaZaa, eMule and Bittorrent [1]. The large-scale use of peer-to-peer for multimedia streaming has been explored for VoIP (with the most notable implementation being Skype) and recently for video streaming and broadcasting in Joost³. Several challenges need to be dealt with when considering a peer-to-peer approach for video streaming and overcoming these difficulties is a component of this work. The target output is a simple yet functional peer-to-peer video streamer capable of providing a fluid video stream (with acceptable quality) to a large number of clients using basic PC hardware for servers and starter peers. This streaming platform is used to experimentally determine if a small and simple application that offers significant network scale is feasible.

II. BACKGROUND

A. Video compression

Streaming of a live video source would be impossible without video compression. Video compression is an important aspect that must be carefully considered and implemented properly.

¹Domain Name Service

²Internet Relay Chat

³Joost is an interactive software for distributing TV shows and other forms of video over the Web using peer-to-peer tv technology

1) *Handling video sources*: A number of static and dynamic sources can be used for video streaming. Static video sources include video-on-demand, DVD or playable video files in a variety of container formats (such as AVI⁴, MKV⁵, MP4/MPEG-4[2]). Dynamic video is captured live and is created at the same time that it is to be streamed. These two types of video have completely different packaging schemes. Essentially, real-time video streaming only focuses on streaming live sources but should be able to supply a video stream from a file in a broadcasting environment (if required). Dynamic streams can be constructed using various streaming formats (and are often based on variations of the stored video containers). Pure streaming formats are less common but do exist (examples include SWF⁶, RealMedia⁷ and FLV⁸).

2) *Platform compatibility*: A good streaming system should exhibit platform compatibility so as to not restrict users to any particular platform. In this case, open source tools (such as ffmpeg⁹) are used in order to remove any dependencies on a particular platform. For optimal performance, servers should be developed for a Unix-based platform so that the server can run on a variety of systems including BSD¹⁰. Clients should be able to connect to each other irrespective of application version or platform. Peers exchange data and control information on a IP level and thus, client side applications can be developed for any platform. Platform portability is easier to achieve when making use of open source development.

3) *Real-time compression*: Due to the nature of the streaming application, compression/encoding must be carried out in real-time. Specific methods and techniques have been designed purely for real-time compression such as that proposed by [4] where a real-time digital video compression system (DVCS) is developed using 3-dimensional sub-band coding.

4) *Implementation concerns*: Implementation issues for video compression include the following:

- Intensive processing requirements
- Scalability of implemented video codec
- Attainable level of visual quality for chosen video codec
- Generation of video data that can be segmented and reassembled after packet loss has occurred

⁴Audio Video Interleave

⁵Matroska open source multimedia container

⁶Macromedia Shockwave Flash format

⁷Multimedia streaming format developed by Real Networks

⁸Flash Video format

⁹A building block for reading/writing encoded media files without requiring additional codecs - forms the base of MPlayer [3]

¹⁰Berkeley System Distribution

- Indexing of received video frames for seeking/pausing and resuming

The above-mentioned concerns are pivotal when designing and adjusting the implemented video compression system. A codec meeting these requirements will provide a video stream with minimal frame loss which is essential for streamed video.

B. Video streaming

The process of video streaming differs from that of traditional video compression and playback as a video stream needs to be composed of video frames that are encoded in such a way that playback does not stop (and suffers only slightly) when video frames (or chunks) are lost. The reasoning for this is that video streams are typically served over low bandwidth networks which generally exhibit packet loss under congestion (the most prominent example would be the Internet).

The second main concern for video streaming is encoding performance. Optimal performance is essential to prevent any delays in delivery of video frames. A fluid video stream is reliant on an encoding process that encodes in real-time (i.e. at the same or faster frames/second rate than playback).

1) *Use of compression:* As with stored video, compression plays a major role in video streaming. Implementation of a codec specifically designed for streaming is essential. Examples of codecs designed specifically to cater for streaming video include the following: Windows Media Video, RealVideo and Macromedia Flash Video.

2) *Challenges:* The challenges faced by video streaming include design issues such as the design of the compression scheme (usually of high complexity) as well as the real-time processing requirement. This can place extreme load on an average PC and this should be compensated for in the design of the media preparation component. The media encoder should encode source video and send it to the peer-to-peer network in real time with minimal or no delay.

3) *Stream segmentation and re-assembly:* The choice of compression scheme has a direct impact on the video streaming capability since the compressed video stream must lend itself to segmentation and correct reassembly while recovering or compensating for packet losses. Any video stream should be able to handle packet losses and recover in a graceful way. Enduring blank interruptions in a stream, re-requesting/sending of packets or buffering are all ways that the streaming system can overcome loss of data in a video stream with varying levels of success and acceptability.

4) *Indexing and marking:* Indexing is also important since it may or may not be a requirement that viewers can seek (backwards) in a live video stream. An index can also be used to maintain and reconstruct a video stream if required. The trade off is that packet losses could result in the entire video stream becoming unplayable due to loss of the index and any synchronisation markers which would typically be used for synchronising video with audio and recovering from errors or lost data.

C. Peer-to-peer

Peer-to-peer networks were popularised as a means of distribution of copyrighted multimedia on the Internet. Initially, peer-to-peer networks were only built for file sharing but have become commonplace in other applications. The use of peer-to-peer for video streaming has been very limited to date with only a handful of implementations.

1) *Types of network:* Within the realm of peer-to-peer, different types of network exist. The two primary network types are:

- **Pure peer-to-peer**

Peers act as client and server with no central server managing the network and no single point of failure.

- **Hybrid peer-to-peer**

A server is used to interface peers, manage requests and route data packets to the correct endpoints. Peers host and transfer all the information in the network and maintain direct connections between each other.

The hybrid model is typically preferred for streaming since the management of the stream and of peers in the streaming swarm is simplified.

2) *Applications:* The overwhelming majority of development in peer-to-peer networking focuses on advancing file and information sharing while offering redundancy, security (by obscurity and encryption) and privacy. Other applications include VoIP, formation of large online communities (for gaming, personal communication, online trading etc) as well as video and multimedia streaming (radio and TV).

3) *Challenges:* The challenges faced by peer-to-peer networks are the main reason that they continue to be avoided as a platform for modern applications. These challenges include:

- Implementation complexity due to the management aspect in client-server (hybrid peer-to-peer) and client-client communications (for pure peer-to-peer networks)
- Non-homogenous peers make it challenging to predict network performance and to make allowances for differing resource availability
- Control of the network becomes problematic (especially for pure peer-to-peer) whether it be management of or controlling what information is distributed (for example, copyrighted materials, pay-to-use multimedia etc)

4) *Advantages:* There are several beneficial aspects of peer-to-peer networks that make them ideal for supporting the applications mentioned in the previous section. These include the following:

- All clients in the network provide resources such as processing power, bandwidth and storage space
- Addition of peers increases the total available capacity and thus performance of the entire network if implemented correctly (unlike the traditional client-server model)
- The distributed architecture increases the possible number of failure points with the result being a very robust network

- In a true peer-to-peer network, the level of user privacy is increased due to the obscurity offered in a complex peer-to-peer network since it is not a requirement for each user to maintain a connection with a central server

D. Peer-to-peer streaming (peercasting)

The primary purpose of peer-to-peer streaming is to deliver high volumes of data at minimal cost in a short space of time. Streaming over peer-to-peer is typically based on one of three main architectures: tree-based, split-stream or meshed-overlay streaming. In addition, various techniques are applied to peercasting designs in order to improve performance and to solve specific problems encountered by peer-to-peer networks. These include MDC¹¹ (or layered encoding) [5], specific techniques for low bitrate encoding [6] and scalable video compression methods for application in variable bandwidth environments (such as for real time video streaming over the Internet) [7].

E. Current implementations of peer-to-peer streaming

P2PCast is a decentralized, scalable, fault-tolerant self-organizing peer-to-peer implementation developed to facilitate streaming of content to thousands of nodes from behind a relatively low-bandwidth network [8]. Users contribute their available bandwidth to this splitstreamed system which uses a novel approach to manage the sub-streams as a forest of multicast trees.

PeerCast is an open source streaming media multicast tool which is generally used for streaming audio and makes use of a bandwidth-distributing IP multicast approach where users can set how many relays to connect in the downstream [9].

Alluvium is a peer-to-peer TV broadcasting initiative by the ACTLab¹². Alluvium uses a technology called swarmcasting whereby distributed download acceleration is used to provide peer-to-peer multimedia streaming [10].

Octoshape is a closed source peer-to-peer multicast audio/video streaming system that uses grid distributed bandwidth to minimize the load on the broadcaster's bandwidth. Each user relays a part of the stream to other users in the grid. This implementation has shown to be very stable and fault tolerant due to its grid design [11].

GnuStream is a prototype receiver-driven peer-to-peer media streaming system [12] built on top of *Gnutella*. GnuStream integrates dynamic peer location and streaming capacity aggregation and thus is highly efficient at load distribution and recovering from changes in the structure of the source network.

GhostShare is a peer-to-peer network built on the Pastry substrate [13] and has anonymity and load balancing as design goals. The main application of GhostShare is the provision of an instantly viewable pre-paid video stream that is streamed or downloaded directly from participating peers.

Joost is a commercial peer-to-peer TV application currently in beta testing. The operational architecture is not disclosed which makes it difficult to determine what type of peer-to-peer

network is being implemented. Links with the developers of KaZaa and Skype imply a network with dedicated hosts (i.e. hybrid peer-to-peer) and supernodes.

F. Peer-to-peer multimedia streaming methods used in the literature

[14] propose a TCP-friendly network adaptive video coding algorithm which exploits path diversity using MDC. This is highly effective in the peer-to-peer environment due to the variation in streaming sources. A peer-to-peer model for on-demand streaming is proposed in [15]. This model is functionally similar to Bittorrent in that seeders are used to serve data but it differs in the complex indexing system used.

III. IMPLEMENTATION DESIGN

The design for the implemented peercasting network uses a combination of the tracking server and client model as well as the true serverless peer-to-peer network architecture and is thus classified as a hybrid peer-to-peer network. Apart from connection initialisation and setup, the majority of communication (and most importantly, the exchange of video data chunks) happens directly between interconnected peers. This functionality means that this streaming system is based on a meshed-overlay model. The implementation is designed to be simple and to show that a basic peer-to-peer streaming system can be relatively simple to develop. Extension or improvement of a current system is not the aim of this work.

A. Network structure

The network is composed of a static server (known as the tracking server) which is responsible for establishing communications between peers and also controlling the exchange of video chunks. Each and every node within the network maintains a connection with the tracking server as well as a short list of peers supplied either by the tracking server or by the connected peers. A redundancy list is maintained and used for fast recovery from node failures.

The tracking server stores a list of super-peers (able to provide more resources than regular peers) which can and will be considered the starter peers in the distribution model. These starter peers will normally receive all the original data from the video server and will be responsible for transporting received data chunks to a larger number of peers. The typical minimum distribution would be $n + 1$ or greater downstream copies for each n starter nodes where $n > 3$.

1) *Structure and function of the Tracking server:* The tracking server is designed to be modular. The functions of the tracking server are:

- initiate communications with starter peers
- connect starter peers to the media server
- provide new peers with the address of a semi-random upstream peer
- handle orphaned peer nodes
- construct and maintain the peer-to-peer network

The main function of the tracking server is to handle new peers that join the network. The tracking server maintains a database

¹¹Multiple Description Coding

¹²at the University of Texas in Austin, TX

and dynamic layout of the network (similar to Figure 2). The purpose of tracking the network layout is to continually have an ideal placement point for new nodes joining the stream. The tracking server sets up peer-to-upstream-peer communications. After initial setup, peers only communicate with their upstream peer (and downstream peers eventually) but keep a connection to the tracker open for status updates and peer connection requests.

2) *Peer architecture*: Each peer node in the network has a relatively complex architecture. This architecture is designed to offer as much *pure* peer-to-peer functionality as possible and to reduce the load requirement on the tracking server and media server. Peers become dependent on their immediate upstream peers but also maintain a *backup* list of connectable peers in case of stream failure. Regular peers are able to connect directly to the tracking server but will always receive stream packets from their immediate upstream peers. These regular peers will never connect to the media server or any other peers unless their upstream peer node(s) go down. Immediately upon connection to the video stream, peer nodes will receive the IP address of their 3 upstream peers (as well as 3 backups) to which they will connect and request their *copy* of the stream (in the form of data *parts*).

B. Operational flow

It is assumed that the tracking server and media server are functionally different though it may be possible to perform both functions using a single, powerful server. Initially, the tracking server will maintain a static list of n starter peers that will connect directly to the media server (responsible for encoding and preparing video chunks for distribution).

Starter peers will maintain a persistent connection to the tracking server such as to maintain an open channel to receive potential downstream peers. A node joining the peer-to-peer streaming network will immediately connect to the tracking server (the only server that the new node will *see*). The tracking server will supply connection details for n starter peers to which the new node will connect.

These starter nodes will have an upper limit on the number of active connections. Once this limit has been reached, these nodes will pass all connection requests to downstream nodes in a linear way. For example, the first such request goes to the first node to be connected (numbered as A1 for starter node A etc) while the subsequent request would be passed to the second and so on. These parameters of operation may be adjusted for optimization based on node characteristics.

C. Application model

The peercasting application encompasses three main modules:

- 1) **Media server** which handles the media source, encodes video and streams video chunks (all in real time)
- 2) **Tracking server** which manages all communications in the streaming client network
- 3) The **peer module** is the client/viewer and mini-server for receiving and re-distributing video data

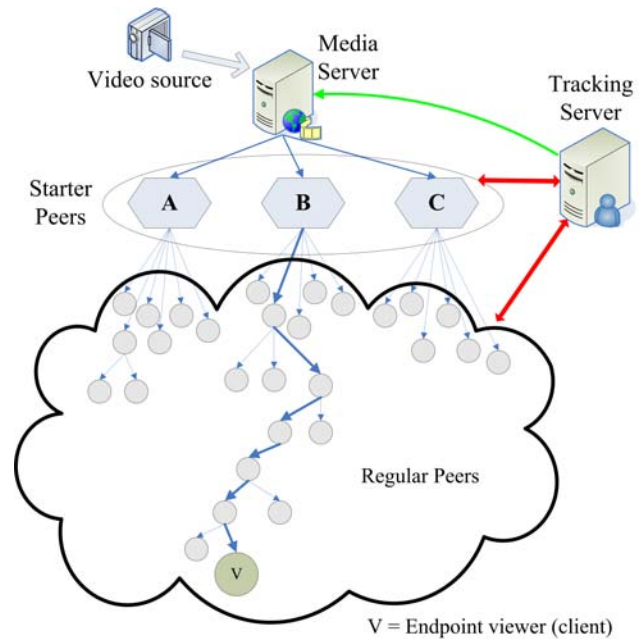


Fig. 1. Basic network layout and operational model of the designed peer-to-peer media streaming network

The application model is presented in Figure 1 (there are two *types* of peers). Node labels are assigned in joining order and are assigned for optimization and for tracking the data path back to the source. This enables faster re-connection after node failure and allows for accurate reporting on the performance of distant nodes within the network.

D. Design specifications

The design for the peer-to-peer system borrows from aspects of other similar systems but implements unique methods for controlling and managing peer-to-peer traffic flow. The tracking and media servers are designed to be scalable, modular and portable to any platform with the main development focused on implementation in a Linux environment.

The design is expected to perform well in a network with up to 2% average packet loss. This figure is very high resulting in a sufficiently fault tolerant and error resilient video codec being implemented. A chunk-based video packaging and transmission approach (within the peer-to-peer system) allows for a higher redundancy by shrinking the affected portion of the video when packets are dropped. The length of a dropped sequence is no longer than the chunk from which the dropped packet was originally created.

E. Peer-to-peer concepts

This section briefly describes how the implementation addresses typical peer-to-peer streaming concepts.

- **Packet scheduling** Packets are scheduled and sent based on incoming requests. These requests are timed to build up and maintain a small playback buffer.
- **Rate allocation algorithm** The encoding rate is static and is set or allocated based on the average profile (in

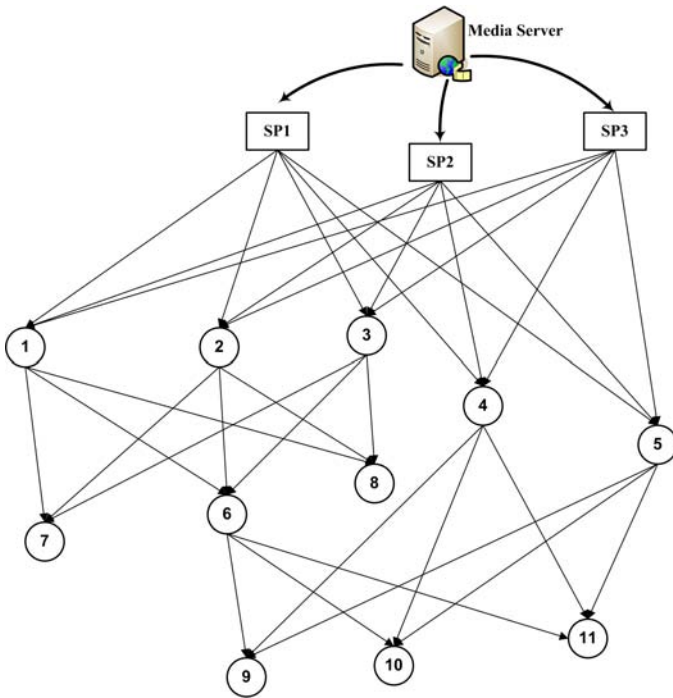


Fig. 2. Example network configuration and layout for analysis and simulation

terms of minimum bandwidth) of the target end user.

- **Peer classification** Peers are classified into levels based on available resources and/or number of hops from the source.
- **Peer selection** The peer selection algorithm is implemented by the tracking server and depends on peer classification as well as upload slot capacity and utilisation.
- **Admission control** Admission is controlled by the tracking server. Peers can be authorised and can only connect by initialising with the tracking server.
- **Incentive mechanism** No incentive mechanism is applied. However, peers contributing more upload bandwidth will be placed higher up in the hierarchy. The result will be a more stable stream.

IV. ANALYTICAL AND SIMULATION RESULTS

A. Experimental setup

The environment used for analysis and simulation is modelled to resemble a typical deployment scenario as shown in Figure 2. The media server is assumed to have full 100Mbps symmetric bandwidth whereas starter peers are limited to 4096/2048 up/down Kbps such as to model a fast Internet connection. Regular peers are configured as 1024/512 so that the performance using a typical broadband connection can be determined.

B. Mathematical analysis

The analysis of the implemented system is based on the methodology and results of [16]. Mathematical analysis is based on the system configuration described above. Analysis is used to determine expected overheads as well as the

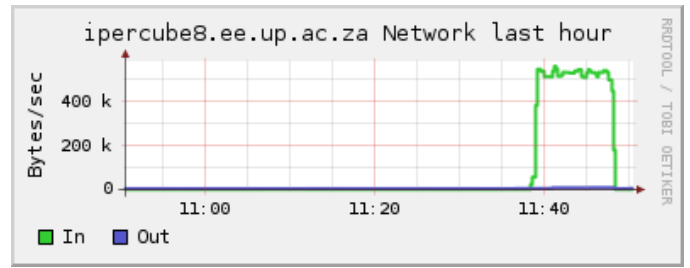


Fig. 3. Download graph for a typical starter peer

conditions under which universal streaming (where each and every peer successfully receives the stream) is possible. It was determined by analysis that based on the configuration for the implementation scenario, $0.37p_1$ super peers are required for p_1 regular peers (over the entire network independent of network architecture and topology) in order to support universal streaming. This result implies that over and above the number of starter peers (i.e. super peers), a number of super peers may be required in order to guarantee universal streaming. Equations were developed to model the performance of the communication and control protocol. An expression for determining the maximum encoding rate for the stream was constructed based on the media server upload rate and the minimum download rate for a regular peer. This was used to select the encoding rate for simulation.

C. Simulation

The implemented peer-to-peer streaming solution was simulated by encoding and streaming a video using the developed components. A small number of starter peers were used to stream video to a set of regular peers. The aim of the simulation was to ascertain whether or not streaming using peer-to-peer was advantageous for a media server. Figure 3 shows the download graph for a typical starter peer. An upload bandwidth consumption graph (Figure 4) shows the typical media server outgoing bandwidth for 4 connected starter peers. Outgoing bandwidth was found to be constant during a typical peer-to-peer streaming session with 3 starter peers and up to 12 regular peers. A comparison of available video codecs highlighted that MPEG-4 was the optimal codec based on a quality ratio determined as $\frac{EncodingBitRate}{AverageFramePSNR}^{13}$. This comparison is shown in Figure 5.

V. FUTURE WORK

This work lends itself to a number of enhancements. The peer-to-peer component in the implementation is generic and can be adapted for basic sharing of files and other multimedia. The study of peer-to-peer as a platform for multimedia streaming can be extended to include incorporating a QoS model, multi-layer encoding, the ability to share multiple channels (while viewing only one) and in a similar vein other PVR¹⁴

¹³Peak Signal to Noise Ratio - a measure for overall inter-frame quality measurement

¹⁴Personal Video Recorder

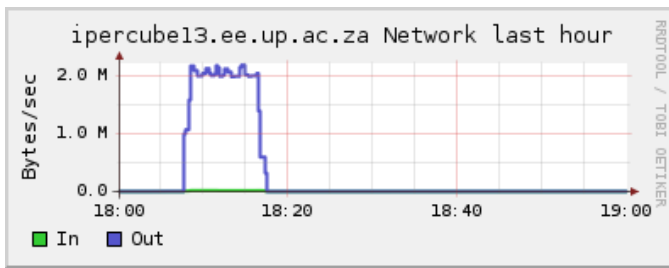


Fig. 4. Upload graph for a media server with 4 connected starter peers

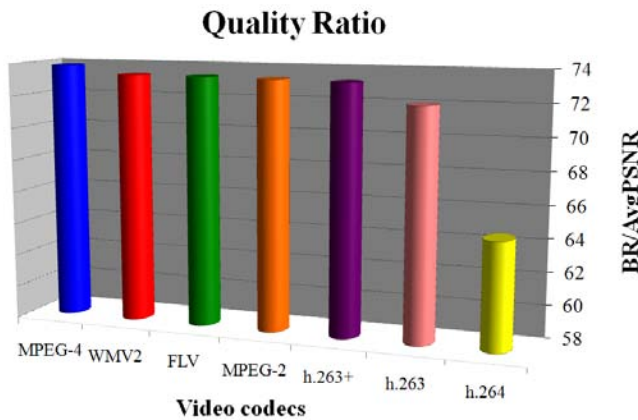


Fig. 5. Quality ratio comparisons for codec compression testing performed

functionality. Future development of integration interfaces (for example, with broadcast TV or with other streaming platforms) would add value to any peer-to-peer streaming platform.

VI. CONCLUSION

Providing a real-time peer-to-peer driven video stream was challenging, yet possible. Streaming servers are typically dedicated with high bandwidth and stream video using a one-to-many approach. A peer-to-peer distribution system allows any Internet-connected PC to broadcast video and enables a streaming network to support a higher number of viewers compared to the current unicast method. The advantages of a many-to-many network are offset by design complexity and other challenges including dealing with a high churn rate and consequently limited performance predictability. Simulation and analysis of a custom-designed implementation indicated that a live video stream can be maintained using a server with relatively low bandwidth as the network scales in size. The peercasting system can also be used to scale existing streaming networks (as a repeater). The implementation showed that a user can stream video data to a large network using no more bandwidth than 3-4 times the video encoding bit rate and that the network scale is not dependent on the total resources made available by this user.

REFERENCES

- [1] "Bittorrent Protocol Specification v1.0," Theory.org, April 2006. [Online]. Available: <http://wiki.theory.org/BitTorrentSpecification>
- [2] "MPEG-4 Part-2 standard specifications - Visual: A compression codec for visual data (video, still textures, synthetic images, etc.)," ISO/IEC Standard 14496-2, ISO/IEC, May 2004. [Online]. Available: <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=39259>
- [3] (2006, April) MPlayer: THE Open source media player. Made available under GNU General Public License v2. [Online]. Available: <http://www.mplayerhq.hu>
- [4] P. Chilamakuri and F. Guediri, "An affordable solution to real-time video compression," in *Southcon '95 Conference Record*, March 1995, pp. 261–265.
- [5] S. P. Panwar *et al.*, "Streaming layered encoded video using peers," in *IEEE International Conference on Multimedia and Expo (ICME)*. Center for Advanced Technology in Telecommunications and Distributed Information Systems, Polytechnic University, July 2005.
- [6] E. K. Y. Dong and D. Zhenhai, "Exploiting Limited Upstream Bandwidth in Peer-to-Peer Streaming," in *IEEE International Conference on Multimedia and Expo (ICME)*, July 2005, pp. 1230–1233.
- [7] M. Johanson, "A Scalable Video Compression Algorithm for Real-time Internet Applications," in *4th EURASIP Conference focused on Video / Image Processing and Multimedia Communications*, July 2003, pp. 329–334.
- [8] A. Nicolosi and S. Annappureddy, "P2pcast: A peer-to-peer multicast scheme for streaming data," in *First IRIS Student Workshop*, August 2003. [Online]. Available: <http://www.cs.nyu.edu/~nicolosi/P2PCast/P2PCast-PR.pdf>
- [9] (2006, March) Peercast: P2P broadcasting for everyone. Made available under GNU General Public License. [Online]. Available: <http://www.peercast.org>
- [10] (2006, February) Alluvium: P2P TV streaming. University of Texas, Austin, TX. [Online]. Available: <http://actlab.tv>
- [11] (2005, October) The Octoshape Live Streaming solution. Octoshape APS. [Online]. Available: <http://www.octoshape.com>
- [12] Y. Dong *et al.*, "GnuStream: a P2P Media Streaming Prototype," in *IEEE International Conference on Multimedia and Expo (ICME)*, vol. 2, July 2003, pp. 325–328.
- [13] G. P. A. Nandan and P. Salomoni, "Ghostshare - Reliable and Anonymous P2P Video Distribution," in *IEEE Communications Society Globecom 2004 Workshops*, 2004, pp. 200–210.
- [14] J. K. Y. Altunbasak and R. M. Mersereau, "Network-adaptive video streaming using multiple description coding and path diversity," in *IEEE International Conference on Multimedia and Expo (ICME)*, vol. 2, July 2003, pp. 653–656.
- [15] B. K. Bhargava and M. M. Hefeeda, "On-Demand Media Streaming over the Internet," in *The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 03)*, 2003.
- [16] R. Kumar, Y. Liu, and K. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proceedings of IEEE Conference on Computer and Communications (INFOCOM) 2007*, 2007, unpublished. (Last accessed: March 2007). [Online]. Available: <http://eeweb.poly.edu/faculty/yongliu/docs/streaming.pdf>

Jeff Hinds completed his B.Eng(Computer) and B.Eng(Hons)(cum laude) degrees at the University of Pretoria in 2004 and 2005 respectively. He has completed an M.Eng degree at the Department of Electrical, Electronic and Computer Engineering at the University of Pretoria and is currently employed full time at Telkom SA in the Technical Product Development service unit. His research interests include video technologies, caching and localisation, peer-to-peer protocols, wireless networking as well as general IP based services and technologies.