Real time video streaming using a peer-to-peer data distribution approach

Jeffrey Hinds
Department of Electrical, Electronic and Computer Engineering, University of Pretoria
Telephone: (012) 420-2177 Email: jeff@tuks.co.za

Abstract— **This paper discusses peer-to-peer technologies as well as aspects related to video streaming with the eventual purpose of combining these two fields to produce a peer-to-peer based video streaming service. An implementation is developed and described in the paper. Video streaming applications are becoming more commonplace with the adoption of video calling, mobile TV, DVB-H[1] as well as Internet radio and TV stations. Peer-to-peer technology brings the capability to stream live video to any regular PC thus allowing any Internet connected computer to become a broadcast station with an almost unlimited number of viewers or clients. Recent developments in peer-to-peer file sharing technologies can be exploited and utilized to build effective swarm-based peercasting systems. The proposed implementation strives to take advantage of recent advancements and currently successful technologies while encompassing a design that solves a number of common problems faced by peer-to-peer multimedia streaming applications.**

Index Terms— **Peer-to-peer, real-time, streaming, video, multimedia, peercasting**

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 DNS[2], IRC[3], VoIP[4] as well as a plethora of file sharing protocols and applications such as KaZaa[5], eMule[6] and Bittorrent [2]. The use of peer-to-peer for audio transfer has been explored for VoIP (most notably Skype[7]) but as yet, no major video streaming services have been developed based on a pure peer-to-peer model. Several challenges are faced by a peer-to-peer approach for video streaming, overcoming these difficulties is the focus of this work. The target output is a functional peer-to-peer video streamer capable of providing a fluid video stream (with acceptable quality) to a large number of clients (or peers) with a basic PC as the server or starter-peer.

II. Background

A. Video compression

Streaming of a live video source would be impossible without video compression due to the intensive nature of video data. Video compression is an important aspect that must be carefully considered and implemented properly for maximum functionality.

1) Handling video sources: A number of static and dynamic sources can be used for video streaming. Essentially, the difference is that static video is typically pre-recorded or previously prepared and can be considered stored video. Examples of static video include video-on-demand, DVD or playable video files. 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 only focuses on streaming live sources but should be able to supply a video stream from a file in a broadcasting environment (if required).

Static video sources are generally stored in files (or similar media such as DVDs, tapes etc) and thus have a variety of container formats (such as AVI[8], MKV[9], MP4[10]) but are essentially the same. Dynamic streams can be constructed using various streaming formats. These can be variations of the stored video containers. Pure streaming formats are less common but do exist (examples include SWF[11] and RealMedia[12]).

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 ffmpeg[13]) are used in order to remove any dependencies on a particular platform. For optimal performance, the tracking server will be developed for the Linux platform. Clients will be able to connect to each other irrespective of application version or platform. This is one of the cornerstones of a peer-to-peer system. Peers exchange data and control information on a IP level and thus, client side applications can be developed for any platform. Platform portability is the primary advantage of 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

[1] Digital Video Broadcast-Handheld [1]
[2] Domain Name Service
[3] Internet Relay Chat
[4] Voice over Internet Protocol
[5] http://www.kazaa.com
[6] http://www.emule-project.net
[7] http://www.skype.com
[8] Audio Video Interleave
[9] Matroska open source multimedia container
[10] MPEG-4 [3]
[11] Macromedia Shockwave Flash format
[12] Multimedia streaming format developed by Real Networks
[13] A building block for reading/writing encoded media files without requiring additional codecs - forms the base of MPlayer [4]

purely for real-time compression such as that proposed by [5] 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
- Indexing of received video frames for seeking/pausing and resuming

The above-mentioned concerns are pivotal in selecting and adjusting the implemented video codec or compression and preparation system. A codec that overcomes these concerns will provide a video stream with a high average QoS[14] which is essential for streamed video.

B. Video streaming

The process of video streaming differs from that of traditional video compression and playback or decompression. A video stream needs to be composed of video frames that are encoded in such a way that the stream playback does not stop (and if possible, suffers only slightly) when video frames are lost. This should include the case where numerous frames (or chunks) of video are lost. The reasoning for this is that video streams are typically served over low bandwidth networks which generally exhibit substantial packet loss (the most prominent example would be the Internet).
The second main concern for video streaming is performance (during encoding). A real time stream should be encoded (or compressed) in a relatively short space of time. This is essential to prevent any delays in delivery of video frames due to processing. A fluid video stream is reliant on an encoding process that can encode video data in real-time (or faster).

1) Use of compression: As with stored video, compression plays a major role in video streaming. Compressed video data uses much less storage space than uncompressed video and thus uses available bandwidth much more efficiently in a streaming scenario. The choice of compression 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.
Implementation of a codec 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.
Another issue faced by video streaming is client compatibility. The streaming source should reach the widest possible audience and should not make use of an obscure codec that very few viewers will have installed out-of-the-box.

3) Stream segmentation and re-assembly: A video stream should be encoded to support segmentation and near-perfect reassembly. Any video stream should consider packet loss and be able to recover in a graceful way. Enduring blank interruptions in a stream, re-requesting/sending of packets or buffering are all ways that can be used to overcome loss of data in a video stream with varying levels of success and acceptability.
Indexing is also important since it may or may not be a requirement that viewers can seek (backwards) in real-time 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.

C. Peer-to-peer

Peer-to-peer networks were popularised by the rise of file sharing and 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 (such as VoIP). The use of peer-to-peer for video streaming has been very limited to date with only a handful of implementations (discussed below).

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 central router.
- Hybrid peer-to-peer
  A central 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.

2) Applications: As mentioned earlier, the primary application of peer-to-peer networks is file sharing. The overwhelming majority of development in peer-to-peer focuses on advancing file and information sharing while offering redundancy, security (by obscurity) 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 and client-client communications (for hybrid peer-to-peer)

[14]Quality of Service

- In a true meshed pure peer-to-peer network, maintaining balanced information sharing and control can be very difficult
- 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 network as a whole (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 the central server

D. Peer-to-peer streaming

The primary purpose of peer-to-peer streaming is to deliver high volumes of data at minimal cost. Different architectures for streaming over peer-to-peer exist of which two examples are the tree based multicast system and the split stream multicast system. Other streaming applications are discussed later in this section. In addition, various techniques are applied to peer-to-peer streaming designs in order to improve performance and to solve specific problems encountered by peer-to-peer networks such as multiple description coding (or layered encoding) [6], specific techniques for low bitrate encoding [7] and scalable video compression methods for application in variable bandwidth environments (such as for real time video streaming over the Internet) [8].

1) Tree-based multicast system: This streaming network is based on a single tree architecture with a top level node, interior nodes and leaf nodes as shown in Figure 1. This is the general peer-to-peer streaming model. The tree-based approach has two major drawbacks: as it scales, the tree becomes heavily unbalanced with the majority of the load being placed on nodes that are *higher* up and secondly, interior nodes become bandwidth bottlenecks as the overall tree grows larger (assuming these nodes are regular nodes).

2) Split stream multicast system: An improvement to the tree-based system is proposed in [9] where the stream is split into multiple stripes which individually use separate multicast trees to distribute each stripe of the stream to downstream peers (see Figure 2. The ideal in this model is that most peers are interior nodes in only one tree and leaf nodes in all other trees. This system distributes the forwarding workload among all peers and provides maximum benefit when a large number of co-operative peers request a single stream [10]. The advantage is that peers can choose to join a subset of the provided stripes and thus control their available bandwidth. Robustness is improved due to the fast recovery from a stripe failure. The advantages are offset by the difficulty in finding an optimal *forest*.

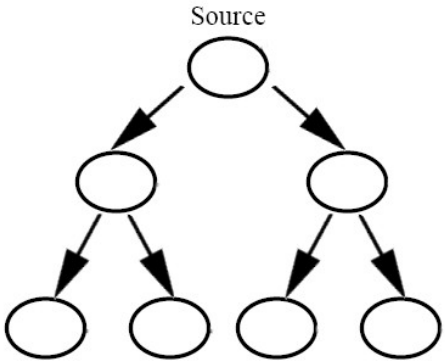

Fig. 1. A tree-based multicast system

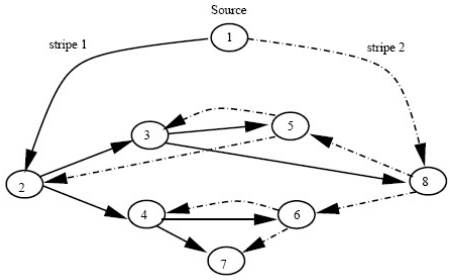

Fig. 2. The splitstream approach where the original source is split into two stripes with individual multicast trees

3) Other examples of peer-to-peer streaming

P2PCast

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 [11]. Users contribute their available bandwidth to this splitstreamed system which uses a novel approach to manage the stripes as a forest of multicast trees.

PeerCast

PeerCast is an open source streaming media multicast tool which is generally used for streaming audio and makes use of a bandwidth distributing approach where users can choose how many relays to connect in the downstream [12].

Another peer-to-peer media distribution system has also been named PeerCast [13]. This particular implementation employs a complex IP multicast approach focused on disseminating data throughout a peer-to-peer file sharing network.

FreeCast
FreeCast is a Java based audio broadcasting application that utilizes the upstream bandwidth of listeners to provide the audio stream to other listeners thus alleviating the bandwidth strain for large audio broadcasting networks [14].
Alluvium
Alluvium is a peer-to-peer TV broadcasting initiative by the ACTLab[15]. Alluvium uses a technology called swarmcasting whereby distributed download acceleration is used to provide peer-to-peer multimedia streaming [15].
Octoshape
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 [16].
GnuStream
GnuStream is a prototype receiver-driven peer-to-peer media streaming system [17] 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
GhostShare is a peer-to-peer network built on the Pastry substrate [18] and has anonymity and load balancing as the primary 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.

4) Peer-to-peer multimedia streaming methods used in the literature
[19] propose a TCP-friendly network adaptive video coding algorithm which exploits path diversity using multiple description coding. 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 media streaming is proposed in [20]. 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 peer-to-peer network uses a combination of the tracking server and client model as well as the true serverless peer-to-peer network infrastructure. This is achieved by using a tracking server to initiate and control connections. However, the majority of communications (and most importantly, the exchange of video data chunks) will happen directly between interconnected peers. These peers will use the tracking server as a discovery mechanism. The basic network layout is shown in Figure 3.

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 will maintain a connection with the tracking server as well as a short list of peers supplied either by the tracking server (mostly) or by the connected peers. A redundancy list will be maintained and used for fast recovery from node failures.
The tracking server will contain a list of super-peers (able to provide more resources than regular peers) which will be considered the starter peers in the distribution model. These peers will 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 (scaling exponentially) for each *n* starter nodes.

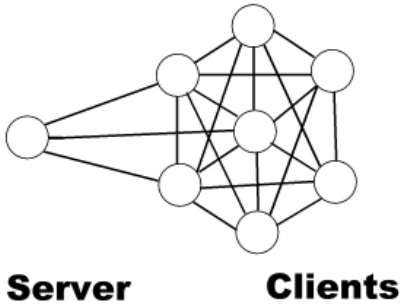

Fig. 3. The basic peer-to-peer model with the tracking server and clients. In this case, the media server and tracking server are one and the same

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 source
- 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 have recently joined the swarm. The tracking server maintains a complex database and rough layout of the network. This layout is truly dynamic and ever changing. The purpose of the network layout is to continually have an ideal placement point for new nodes joining the swarm for optimized peer placement. The tracking server sets up peer-to-upstream-peer communications. After the initial setup, peers only communicate with their upstream peer (and their downstream peers eventually) but maintain an open connection to the tracker for status updates, downstream connection requests etc.
The tracking server keeps track of the starter peers and maintains contact with the top level peers at all times.

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

[15]at the University of Texas in Austin, TX

to reduce the load requirement on the tracking server. Peers become dependent on their immediate upstream peer but also maintain a *backup* list of connectable peers in case of stream failure(indicated by the dashed lines between peers in figure 4. Regular peers are able to connect directly to the tracking server (shown as dotted lines in figure 4 but will always receive stream packets from their immediate upstream peer. 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 eventual upstream peer (as well as 3 backups) to which they will connect and request their *copy* of the stream.

B. Operational flow

It is assumed that the tracking server and video 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 (the hexagonal blocks in Figure 4 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 (the thick lines in figure 4 such as to maintain an open channel to receive potential downstream peers. A node joining the peer-to-peer swarm will immediately connect to the tracking server (the only server that the new node will s̈ee). 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 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** the client/viewer and mini-server for receiving and re-distributing video data

The application model is presented in figure 4 where the different identities can be identified (there are two *types* of peers). The peer naming convention is as follows: Xj-k-l-m (backup node(s)) where X represents the starter node and j-k-l-m the intermediary nodes from the starter node down to the named node. Node labels are assigned in joining order. Node IDs are labelled this way 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.

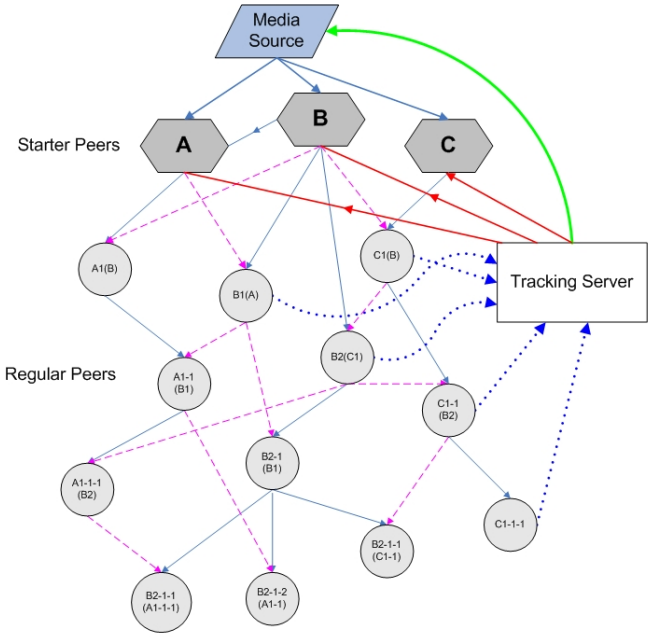

Fig. 4. Layout, communications and operational model of the designed peer-to-peer media streaming network

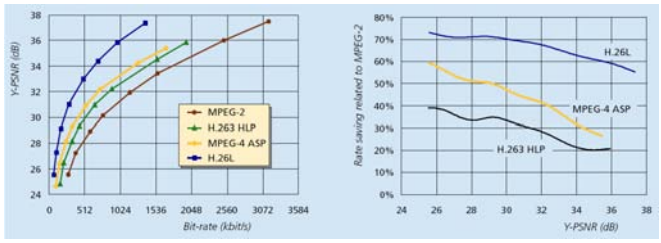

Fig. 5. Y-PSNR comparison for various codecs at various bitrates and the bitrate savings achieved compared to MPEG-2 and H.263 [22]

D. Design specifications

The video codec implemented is an H.264 MPEG-4 codec [21] which has been adapted and optimised for streaming. This codec offers exceptional visual quality with a manageable and adjustable implementation complexity. The choice of H.264 is driven by the higher quality achievable for a lower bit rate as shown in Figure 5.

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 server and video server are designed to be completely scalable, modular and portable to any platform with the main line of development focusing on implementing the servers in a Linux environment.
The design is expected to perform well in a network with up to 5% average packet loss. This figure is very high resulting in a sufficiently fault tolerant and error redundant video codec being implemented. A chunk-based video packaging and transmission approach (within the peer-to-peer system) allows for a slightly 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 in which the dropped packet was originally packaged.

IV. Future work

This work lends itself to a number of enhancements. The most prominent development will be the modularisation of the various components. This would allow complete reuse of the designed peer-to-peer model for any form of multimedia streaming, VoIP or file sharing. Standardisation of the interface between the peer-to-peer controller and the video streaming processor will allow for drop-in replacement of any multimedia codec (including improved codec revisions as well as future high quality codecs). This would allow integration of a data module or an enhanced video codec (with embedded content or DRM[16] functionality as an example). As with all platform specific software, porting of the user software and server software across platforms (Win32, Win64, Linux, MacOS, FreeBSD etc) is always a driver for future development.

V. Conclusion

Providing a real-time peer-to-peer driven video stream has proven to be challenging, yet possible. Typically, video stream sources are dedicated high bandwidth servers which stream video data using a one-to-many approach. Implementation of a peer-to-peer distribution system allows any regular Internet-connected PC to become a video source. In addition, this approach increases the maximum size of the client swarm. Application of the peer-to-peer approach can allow a typical streaming network to take on a greater number of additional clients when compared to the existing method. The advantages of a many-to-many network are offset by difficulty in design as well as other challenges such as a highly dynamic network and provision of a high level of QoS. Implementation of the H.264 video coding algorithm on a custom design peer-to-peer network framework provides high quality streaming with a simple yet robust data distribution structure.

References

[1] *Transmission System for Handheld Terminals (DVB-H)*, ETSA Std. 302 304.

[2] "Bittorrent Protocol Specification v1.0," Theory.org, April 2006. [Online]. Available: http://wiki.theory.org/BitTorrentSpecification

[3] "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

[4] (2006, April) MPlayer: THE Open source media player. Made available under GNU General Public License v2. [Online]. Available: http://www.mplayerhq.hu

[5] P. Chilamakuri and F. Guediri, "An affordable solution to real-time video compression," in *Southcon '95 Conference Record*, March 1995, pp. 261–265.

[6] 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.

[7] 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.

[8] 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.

[9] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," in *19th ACM Symposium on Operating Systems Principles, 2003*, 2003. [Online]. Available: http://project-iris.net/irisbib/papers/splitstream:iptps03/paper.pdf

[10] A. Habib and J. Chuang, "Incentive mechanism for peer-to-peer media streaming." [Online]. Available: http://p2pecon.berkeley.edu/pub/HC-IWQOS04.pdf

[11] A. Nicolosi and S. Annapureddy, "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

[12] (2006, March) Peercast: P2P broadcasting for everyone. Made available under GNU General Public License. [Online]. Available: http://www.peercast.org

[13] H. D. Karatza and K. G. Zerfiridis, "Large Scale Dissemination Using a Peer-to-Peer Network," in *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 03)*, 2003.

[14] (2005, October) Freecast: Peer-to-peer streaming. Made available under GNU General Public License. [Online]. Available: http://www.freecast.org

[15] (2006, February) Alluvium: P2P TV streaming. University of Texas, Austin, TX. [Online]. Available: http://actlab.tv

[16] (2005, October) The Octoshape Live Streaming solution. Octoshape APS. [Online]. Available: http://www.octoshape.com

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] "MPEG-4 Part-10 standard specifications - Advanced Video Coding: A codec for video signals," ISO/IEC Standard 14496-10, ISO/IEC, December 2005, identical to the ITU-T H.264 standard. [Online]. Available: http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=43058

[22] H. S. R. Schfer and T. Wiegand, "The emerging H.264/AVC standard," Heinrich Hertz Institute, Berlin, Germany, Tech. Rep. 293, January 2003. [Online]. Available: http://www.ebu.ch/trev_293-schaefer.pdf

[16] Digital Rights Management

Jeffrey Hinds completed his B.Eng(Computer) and B.Eng(Hons)(cum laude) degrees at the University of Pretoria in 2004 and 2005 respectively. He is currently working towards an M.Eng degree at the Department of Electrical, Electronic and Computer Engineering at the University of Pretoria as part of the Telkom CeTEIS research group. His research interests include video technologies, peer-to-peer protocols and wireless networking.