A MASSIVELY SCALABLE PERSISTENT CONTENT DISTRIBUTION SYSTEM

Jani Peltotalo, Sami Peltotalo, Alex Jantunen, Lassi Väätämöinen, Jarmo Harju
Tampere University of Technology
Institute of Communications Engineering
P.O.Box 553, 33101 Tampere, Finland
{jani.peltotalo, sami.peltotalo, alex.jantunen, lassi.vaatamoinen, jarmo.harju}@tut.fi

Rami Lehtonen
TeliaSonera Ltd.
Hatanpaan valtatie 18, 33100 Tampere, Finland
rami.lehtonen@teliasonera.com

Rod Walsh
Nokia Research Center
Visiokatu 1, 33720 Tampere, Finland
rod.walsh@nokia.com

ABSTRACT
This paper proposes a novel form of peercasting system as an improved solution for IP-based mass media content delivery. Several approaches are discussed, some of which are already widely deployed, or undergoing such deployment, as homogeneous systems. In particular, IP multicast and Peer-to-Peer (P2P) overlay network techniques are described in isolation - as they form the bedrock of the new peercasting system proposal. This proposal aims to improve existing homogenous and heterogeneous systems for mass media distribution to very large user bases, with needs for timely and reliable delivery, and content persistence. Thus IP multicast has great advantages for delivery with controlled last-mile elements, both for mobile and fixed usage. However, persistence while the number of still-receiving users dwindle, and reliability at reasonable network cost, are better served by P2P techniques. The work primarily considers discrete media delivery, which is useful both alongside with streaming media and in standalone applications. The feasibility of such a combined multicast and P2P system is shown and the working prototype implementation of this proposal, *Delco*, is introduced.

KEY WORDS
Multicast, Peer-to-Peer, Peercasting, Mass media

1 Introduction

The distribution of digital content (music, videos, photos, games etc.) is increasingly becoming the primary task of the IP networks. Coupled with the problems of current networks related to security, capacity demands, heterogeneity of the networks and rights management issues have created a huge demand for advanced solutions in the digital content distribution space.

There are a lot of research activities and studies dealing with different content distribution technologies including multicast, IP Datacast (IPDC), Peer-to-Peer (P2P) and Content Delivery Networks (CDN). A few research groups have also studied how to integrate P2P and CDN technologies. For example Skevik et al. [1] and Xuyz et al. [2] proposed their hybrid solutions for streaming media distribution. However, all the above mentioned content distribution technologies have their own limitations; multicast is currently unutilised, CDN servers are expensive to deploy and maintain, and in current P2P systems there exist severe problems around security issues like Denial of Service (DoS) attacks and viruses.

Multicast was designed globally accessible, but the stateful method for routing packets between operators was considered too resource consuming and therefore global multicast distribution has not been much deployed. Also the need for multicast during 80's and 90's has been quite limited and not until today there exists huge demand for large media delivery over the Internet. The inexistence of global multicast does not however mean that we should not deploy local multicast branches for enhancing the delivery of media if there is a possibility for that. IPTV deployments within operators have been one of the major multicast deployments so far and in those cases the multicast has been enabled only internally and even just for the IPTV service usage. Those multicast enabled isolated networks are coming commonly available and can be also used for other than IPTV service.

P2P alone does not solve the distribution problem, since the network capacity is still a scarce resource and P2P tends to consume at least as much as basic client-server distribution. Of course the bottleneck is not anymore in single links close to media sources, but distributed more evenly in the network. Still there is demand for one-to-many delivery for media files that can be distributed to multiple receivers at the same time. This demand is even extending during the next few years, because of the advances in media consumption over the Internet and increased quality of the media files. Hence, there is a need and possibility for a trusted and controlled mass media delivery system with more optimal use of the network capacity.

In this paper we present a content distribution system, called Delco, which is a new class of peercasting system using both multicast and P2P techniques. Existing peercasting systems (for example PeerCast [3] and TVAnts [4]) have focused on P2P delivery of live video. Recently P2P for video sharing - such as Veoh [5] - may open this up. However, the broadcast medium has provided the thickest pipe to deliver mass media content, and systems such as MBMS and DVB-H promise to deliver mass media also to mobile users in the near future. Both Internet-based and broadcast or ISP walled-garden based datacast have their limitations.

In the Delco system we combine multicast and P2P techniques in a way that has not been previously attempted. System provides the combined benefits of both techniques enabling mass media delivery to massive number of wired and mobile users with a modest service device infrastructure. An overview of the system and further details on the motivation of its use are provided in the following sections. A thorough description of the system architecture and current implementations then precedes a concluding summary and a look into future developments.

2 Motivation

The main motivation for our work is to create better enabling tools for content providers, service providers, network operators and end-users to help the distribution and use of digital content. The advantage of using multicast is to reduce server and the distribution network workload. Similarly the use of P2P helps to reduce server load, which is a weak point when using the traditional client-server model for mass media delivery.

Since multicast commonly uses an unreliable transport protocol (UDP - the User Datagram Protocol), reliability must be achieved by other means. Repeat transmissions and Forward Error Correction (FEC) are two options to achieve the reliability in the main forwarding path. For the return path there are multiple options such as P2P repair and traditional client-server repair. The latter could be for example a HTTP-based file repair scheme. However, if there are a lot of receivers that do not have a complete file after multicast transmission, there becomes a huge load on the HTTP server(s). The former enables receivers to complete the multicast delivery from each other and promises much reduced server load. Figure 1 shows an overall architecture of a combined multicast and P2P content distribution system.

The main requirements for the development of our content distribution system were:

1. The content distribution system should cope with the flash crowd effect and scale to very large audiences.

2. The content distribution and Digital Rights Management (DRM) should be separated, so that the use of various technologies for the delivery as well as for the DRM is allowed.

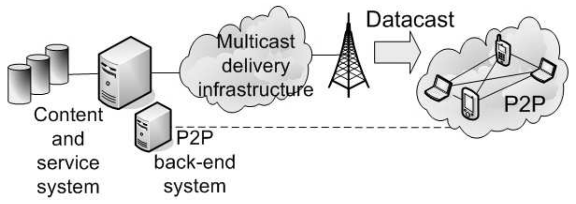

Figure 1. Combined multicast and P2P content distribution system

3. The content distribution network should be resistant to attacks against users (in forms of viruses etc.), content modification and availability of content.

4. The content distribution network should be as independent as possible from the access networks and end-systems.

5. The content distribution mechanisms should be service independent and existing technologies and metadata descriptions should be used as much as possible.

3 System Overview

3.1 Delivery Modes

The Delco system provides three different content delivery modes: multicast-only, combined multicast and P2P, and P2P-only. At the start of a mass media content release there are generally many customers requesting the same content. Hence a service provider can utilise multicast delivery (in multicast-only and combined multicast and P2P modes), which serves very large user groups without overloading server and network resources.

Multicast-only: In this mode reliability may be improved with redundant data (repeated transmissions and FEC data). The amount of redundant data to achieve a certain level of complete reception can be statistically calculated according to predictable characteristics, such as packet loss rate, packet loss distribution, number of receivers, quantity of data, etc. Typical examples of ”a certain level of complete data” would be 99% of receivers achieve complete error-free reception, or 99% of all data is correctly received and decoded by all receivers. Receivers that join a multicast session after its start suffer an initial loss of all data transmitted prior to their joining which complicates this calculation significantly. When we apply the pre-condition that all receivers must be ready to receive multicast data prior to the start of that session, this is significantly simplified and the amount of redundant data required to achieve the same level of complete reception is optimally minimised. However, this pre-condition implies that all receivers must ”join” the session during a time window before its data transmission and that ”late-joiners” are not allowed (or at least no guarantee is offered for their reception quality). Multicast-only delivery also suffers from the limitation that access networks which do not support multicast will not support any multicast-only service at all.

Combined Multicast and P2P: In addition to the multicast delivery the P2P delivery is also enabled and clients whose access networks do not support multicast can join the P2P delivery. Clients whose access networks support multicast can start seeding to non-multicast clients after they have received some data through the multicast delivery (simultaneous multicast and P2P usage). Multicast enabled clients can also use P2P to repair/complete the multicast delivery either during the multicast transmission period (simultaneous multicast and P2P usage) or afterwards (first multicast then P2P usage). In the first case client should not repair data that is to be received through multicast delivery. Using this mode, all receivers can be served, no matter whether they have multicast support or not. The service provider must have P2P seed(s) to backup the delivery, and possibly also initiate it, in a case that there is not a 100% copy of the file among the clients. The server workload for providing small subsets of files will still remain much lower than in the traditional client-server model. In this mode receivers are allowed to join the multicast session during its lifetime, and due to this, the need for a P2P repair after the multicast delivery is higher for any ”late-joiners”. After the popularity of the mass media content has decreased or the amount of multicast clients has dropped below a certain threshold, it is not reasonable to use multicast delivery anymore, and the service provider can use only P2P delivery. For example, the threshold can be calculated from the relative costs of multicast and unicast delivery. If the popularity of the mass media content increases again the service provider can shift back to combined multicast and P2P delivery and schedule a new multicast transmission period.

P2P-only: In this mode a service provider must have one or more P2P seeds to initiate and backup the delivery. The server workload for providing subsets of files decreases when the clients start seeding after they have received some blocks of the files.

3.2 Service Types

The Delco defines two service types: download and channel. In the former the exact content of a service is known before the service transmission starts, and in the latter it is not. The channel service may also be described as a semi-automatic download, where a user accepts the content delivery as for example in a software security update service and the content is automatically downloaded at the receiver. Neither the download service nor the channel service depend on the used delivery mode.

The download service can also be a stream service with two different subtypes. First subtype is a ”progressive download” service, where the content is played during its download, and the whole content is ready for the transmission before the transmission starts (this subtype is a specialization of a download service). The second subtype is a ”live streaming”, where the content is buffered from the source and delivered in pieces.

The channel service may be based on news feed services such as Really Simple Syndication (RSS) and Atom feeds. The Delco client fetches and reads RSS feeds periodically with some time interval and, based on the content information found from the feeds, a client might initiate the downloading process.

An example content of a download service is a movie or a movie package (several movies). A channel service could be, for example, a football channel where news and game clips are delivered after they have been produced.

4 Architecture

4.1 Overview

Figure 2 shows the architecture of the Delco system. The content provider has some content which it would like to deliver to its customers. The operator control interface is used as a content injection Application Program Interface (API) between the content provider and the Delco service provider. In the system some media metadata is needed to describe the content and its delivery method. The web server is used to make this metadata available to the customers, who can also make content searches via the web server. The multicast server is used for the multicast content delivery, which is also called scheduled content delivery. The P2P component is used for the P2P content delivery. In the receiver side there can be several types of machines which act as a receiver, e.g., desktop computer, laptop, PDA, and cellular phone.

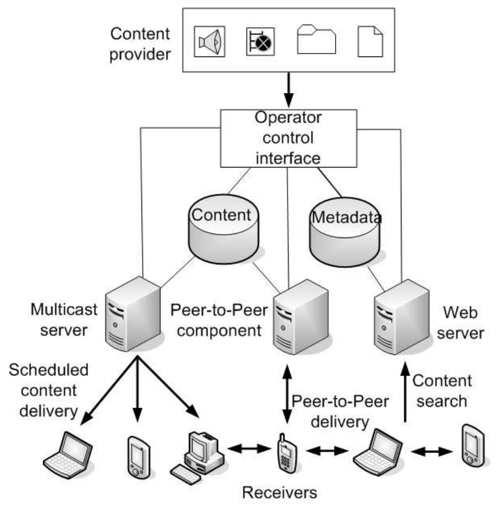

Figure 2. Architecture of the Delco system

The Delco system architecture is specified to support any multicast and P2P technique. For instance application layer multicast and Chord systems could be applied. However, we have implemented the system using File Delivery over Unidirectional Transport (FLUTE) [6] for the multicast delivery, and BitTorrent [7] for the P2P delivery. FLUTE is a protocol for delivering files over the Internet or unidirectional systems from a sender to one or more receivers. It can be used with both multicast and unicast UDP delivery, but it is particularly suited to multicast networks as its lack of uplink signalling enables it to scale massively. Thus, in our implementation, the multicast server is a FLUTE sender. BitTorrent is a protocol for distributing files using P2P techniques. BitTorrent's advantage over plain HTTP is that when multiple downloads of the same file happen concurrently, the downloaders upload to each other, making it possible for the file source to support very large numbers of downloaders with only a modest increase in its load [7]. In our implementation the P2P component consists of a BitTorrent tracker and a BitTorrent seed controlled by the service provider. The receiver includes at least a web browser and the Delco client software, which consists of a FLUTE receiver, a BitTorrent client, an RSS reader, and some glue logic.

Current P2P networks do not guarantee file availability and integrity (fake files can be a problem). In the Delco system, availability is not a problem as the service provider can follow P2P statistics from the BitTorrent tracker and control its own BitTorrent seed. The seed must be online and share the file when a 100% copy of the file is not available among receivers. In other cases, the service provider need not run the seed at all. The statistics cannot be trusted with full certainty because there can be badly behaving clients which indicate that they have the content but do not upload it. However, it does not generate high load to the service provider's seed if it were always on-line because the BitTorrent protocol efficiently distributes the upload task. The fake file problem is also solved in the system, as all content is injected into the network through the operator control interface. This means also that only permitted torrents are shared via the operator's trusted tracker.

4.2 Metadata and Content Pointer Files

A special metadata file (a Delco file) is used to describe the content of a service. The Delco file can describe multiple services, but the default is to describe just one. A service can include multiple content items, each having a unique content ID. A content pointer is used to map a content ID and delivery mechanism of the content. A single content item can have multicast and/or P2P content pointers. If both exists then the combined mode is used, otherwise the specific mode defined by the pointer type is used. The idea is that a Delco client can receive the content according to the instructions given in the Delco file.

In our implementation a Session Description Protocol (SDP) file and a torrent file are used as multicast and P2P content pointers respectively. RSS feeds are used as content pointers in the channel services. It is also possible to define other content pointers, for example ed2k or magnet links for P2P within the limits of the implementation. Our current metadata file has file extension .delco and it uses XML syntax.

By defining the .delco file extension and (experimental) application/x-delco content type, the web browser is able to launch the Delco client after it has received the metadata file. The following example of a Delco file describes a download service with two content items. The content is available via combined multicast and P2P delivery.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<delco>
 <service id="1" name="Movie Package">
  <contentItem id="1" name="Movie 1">
   <contentPointer type="flute"
    value="http://www.example.com/sdp/movie1_flute.sdp"/>
   <contentPointer type="torrent"
    value="http://www.example.com/torrent/movie1.torrent"/>
  </contentItem>
  <contentItem id="2" name="Movie 2">
   <contentPointer type="flute"
    value="http://www.example.com/sdp/movie2_flute.sdp"/>
   <contentPointer type="torrent"
    value="http://www.example.com/torrent/movie2.torrent"/>
  </contentItem>
 </service>
</delco>
```

5 System Considerations

5.1 Advantages of BitTorrent as the P2P Component

The problem with many P2P protocols is that new or casual peers in the network get bad service with long waiting times in queues. The potential of new peers is not fully used and content is spread considerably slower in the face of flash crowds.

BitTorrent provides fast and reliable content sharing in challenging environments. It supports simultaneous downloading from multiple sources and sharing partially downloaded files, so that a peer can upload a file while still downloading it. Download performance is very good even with sudden popularity of a single file (flash crowd) [8].

BitTorrent groups peers according to a torrent file. As they all are interested in the same content, they can effectively help each other completing the download with the help of BitTorrent's algorithms e.g., tit-for-tat incentive mechanism and "local rarest first" policy in piece picking. BitTorrent is also effective against DoS attack where a hostile peer tries to poison a download with corrupted pieces (blocks). A torrent file includes a list of piece hashes (SHA-1), which ensures that received and accepted pieces are always correct.

BitTorrent trackers support content controlling, which means that they can be configured to only share certain torrents via the tracker. The BitTorrent tracker also provides the possibility to see file-downloading statistics. This data is available for each file. BitTorrent does not include any user level content search functionality, which is provided by the web server in the Delco system, but instead BitTorrent is optimised for download performance. This makes it an excellent single purpose plug-in component. BitTorrent is well suited to large content delivery (including gigabyte), and the download block size is adjustable which helps to optimise the download performance and signalling overhead with different file sizes.

5.2 Content Blocking and Integrity Verification

By dividing the content into pieces (blocking) and using hash check for the blocks it is possible to provide reliable simultaneous multicast and P2P delivery. With the help of piece hashes erroneous or missing pieces can be detected and repaired in a way that whole content can be eventually received.

Content blocking can be smoothly used with FLUTE and BitTorrent because block sizes are adjustable in both delivery techniques. However, FLUTE's default block partitioning algorithm generates at most two different sizes for the source blocks, and the sizes are as close to each other as possible.

When the FLUTE's and the BitTorrent's block sizes are equal or the FLUTE's block size is a multiple of the BitTorrent's, all data that has been received by the FLUTE can be utilised. In other cases, a block, which was missed in the FLUTE download, may corrupt data from two or more blocks because all the data cannot be verified (the FLUTE block overlaps two or more BitTorrent blocks).

Block alignment problem with a missing block is described in Figure 3. In 3(a) neither of the FLUTE's block sizes are equal to the BitTorrent's, and one missing FLUTE block corrupts three blocks. In 3(b) smaller FLUTE's block size is equal to BitTorrent's block size, and one missing bigger block corrupts two blocks. In 3(c) FLUTE uses only one block size (Delco enhancement), which is equal to BitTorrent's block size, and one missing FLUTE block affects only one BitTorrent block.

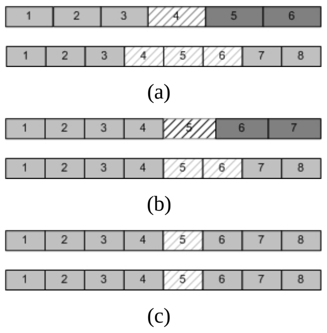

Figure 3. Block alignment problem. FLUTE has the upper block chain and BitTorrent has the lower. The lower block chain represents also Delco receiver's view of the situation.

6 Delco Client Implementation

The architecture of a Delco client implementation is presented in Figure 4. Functional Delco client consists of two parts: User interface and Delco library. Implemented UI uses wxWidgets [9] library, which supports many environments like Win32, Mac OS X, GTK+, X11, Motif, and WinCE. The Delco library is designed to be easy to use, and new applications for different environments can be easily produced as the underlying library is not dependent on the components at the UI at all. Currently Delco library uses Boost [10] for threading and file system operations, Xerces-C++ [11] for parsing XML and Libtorrent [12], FLUTE [13] and cURL [14] for downloading. In the development stage compatibility for Windows and Linux are tested.

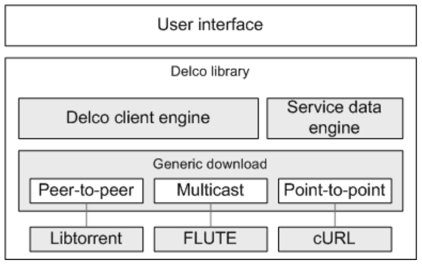

Figure 4. The architecture of the Delco client implementation

Delco library consists of three parts, a Delco client engine, a service data engine and a generic download component. The service data engine is used to construct service data objects from the Delco files. The Delco client engine uses service data objects to create own session for each service. Multiple sessions in a single process is supported by the library. The generic download component defines adapter interfaces for different downloading protocols (peer-to-peer, multicast and point-to-point). With the help of adapter interface it is possible to implement new downloading plug-ins. Plug-ins map the adapter interface to existing library specific functions, which in our case were Libtorrent, FLUTE, and cURL. As we have to retrieve torrents and SDP files through HTTP, we can also use point-to-point (HTTP) in actual download process as an extreme alternative. The Delco client engine uses only adapter interfaces, which gives us later a flexibility to change different protocols and/or libraries and even add more. Quite easily the client can support different P2P libraries simultaneously.

As the same file can be received through many protocols, we must have central point of control for reception. This central component of the Delco client engine is called OutputHandler. Each Delco session has its own OutputHandler and it must be passed to the libraries involved in the session as all file operations (open, write, close) are redirected through it. When an OutputHandler recognizes received data from multicast, it can inform the P2P plug-in (using P2P adapter interface), which further shares the data through P2P network. OutputHandler has the intelligence to control the whole download process. We must also hack a method to a P2P downloading library to make it aware of received data and prevent it from downloading the same data twice. Verifying a received piece is done also through P2P interface, because all modern P2P protocols must verify each piece already independently, so that one block won't ruin the whole download.

Requirements for downloading libraries are that they have C/C++ API and support multiple sessions in a single process. If latter is not met, then Delco library cannot have multiple sessions active within the same Delco process. Also P2P downloading library should support blocking of a file into smaller pieces together with piece hash check, so that we can reliably utilize simultaneous P2P and multicast usage.

Delco library is implemented with C++ using Standard Template Library (STL) for primitive data structures. Also C header is included in OutputHandler to support libraries written in C (like the used FLUTE implementation). As only certain libraries might be available in different environments, generic download component helps to import them into the Delco client as the Delco library uses internally only adapter interfaces. Just a new plug-in must be implemented for a different library support.

7 Delco Server Implementation

The architecture of the Delco server implementation is presented in Figure 5. The architecture is designed with reusability in mind, so that the user interface and data transfer parts (multicast and P2P protocols) could be replaced independently of the current implementation.

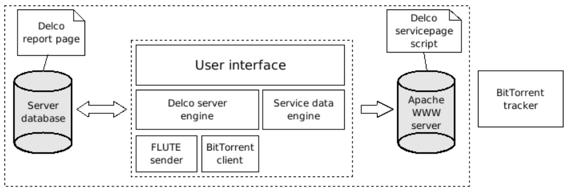

Figure 5. The architecture of the Delco server implementation

The Deldo server system consists of a server database, a Delco specific control interface and data transfer system, a WWW server and a BitTorrent tracker. The database is used for storing the service description data and for client and server reports. The control interface is for administrative use (service creation and content injection), the WWW server is used for service data file distribution and BitTorrent tracker is required for BitTorrent usage. Currently the control interface and the WWW server locate on same host for easier implementation, but it is also possible to distribute the functionality to different host entities.

In addition, there is also no need for the BitTorrent tracker to locate on the same host as the rest of the server, for it is started and setup as a separate process. The only requirement is to define the announcement URL to the Delco server, so that the tracker can be correctly advertised to clients. At the moment BNBT BitTorrent tracker [15] and Apache WWW server [16] are used in the implementation.

7.1 Basic Functionality

To create a Delco service at the server end, a certain procedure scheme must be followed.

First, a multicast session description (an SDP file) must be defined. This can be done by creating the SDP file manually using a text editor or by using the server-offered user interface dialog. Next, a service must be defined. This requires definition of the service type, the content of the service, used delivery methods within the service, and the multicast session description that is used for the service.

If P2P delivery is chosen to be used as the service, the server then creates the torrent files based on the content of the service. Last, a Delco file, which includes the service definitions, must be created. This is done by selecting one or more services from the services available on the server. The server then creates the Delco file and publishes it to the WWW server to make it available for the clients to fetch.

All the functions described above are done by passing the needed input data from the user interface to the service data engine part of the server, which handles the input data and creates both the internal data for server internal use and the service description data files to be used by the clients.

After the service data is created, the server creates a required number of FLUTE sessions and operator back-end BitTorrent seeds for the service content are started by the Delco server engine.

7.2 Implementation Details

The Delco server is implemented using C++, and available software libraries are used wherever applicable. Also, to achieve some degree of coherence in the system, strive is to use same libraries as much as possible on server side that are used in the Delco client application.

Currently used libraries are Xerces-C++ XML parser [11] for writing the Delco file. Libtorrent [12]is used for the BitTorrent client part of the server (for seeding purpose) and for creating the torrent files. Graphical user interface is implemented using gtkmm framework [17], the C++ interfaces for GTK+ libraries. Boost libraries [10] are used for threading and some file system operations. And while these libraries are required by Libtorrent library, using them in other parts of the software minimizes the variety of different libraries used, thus making implementation more sensible.

Although current implementation supports GNU/Linux-platform only (Debian testing/unstable), the Delco server should be portable to other operating systems, at least Win32 environment should be supported with only slight modifications, because all the libraries used are available for Windows-system also.

The Delco service page is generated using a PHP script. The script is accessed via the WWW server and it creates a listing of the available Delco files (using the database) that are currently distributed and accessible by clients.

8 Conclusions and Future Work

This paper presents a mass media content distribution system based on IP multicast and P2P delivery technique. The strengths of both IP multicast and P2P overlay have been leveraged to simultaneously scale server and distribution network capacities to serving greater number of receiving hosts for mass media content. The paper demonstrated the feasibility of such a system, even using existing software components which may be found among the open source community.

Several issues still remain unsolved or only partly solved. Large scale laboratory and field testing of simultaneous multicast and P2P download, with free transitions between P2P and multicast delivery, would highlight system bottlenecks and usability issues. Some security aspects, such as known DoS attacks against P2P networks - like poisoning peer list database of a BitTorrent tracker - are also worthy of further investigation. A detailed performance analysis of the Delco system would yield a better understanding of how to optimise peercasting systems and also the scale of the benefit of implementing custom-designed multicast and P2P components as an alternative to reusing existing generic code.

It is already evident that peercasting is an exciting and potentially fundamental step in the deployment of massively scalable mass media content distribution systems. It includes the promise of improved broadband, mobile datacast and triple/quad-play systems through better optimised underlying technology usage. Future work is expected to reveal the full extent of that improvement and its impact.

References

[1] Karl-Andre Skevik, Vera Goebel, and Thomas Plagemann. Analysis of BitTorrent and its use for the Design of a P2P based Streaming Protocol for a Hybrid CDN. Technical report, Delft University of Technology Parallel and Distributed Systems, June 2004.

[2] Dongyan Xu, Sunil Suresh Kulkarni, Catherine Rosenberg, and Heung-Keung Chai. Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems Journal*, 11(4):383–399, April 2006.

[3] PeerCast. PeerCast P2P broadcasting [online]. 2006. Available from: http://www.peercast.org/ [cited 21 November 2006].

[4] TVAnts. TVAnts streaming software [online]. 2006. Available from: http://www.tvants.com/ [cited 21 November 2006].

[5] Veoh. Veoh Video Network [online]. 2006. Available from: http://www.veoh.com/ [cited 21 November 2006].

[6] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh. FLUTE - File Delivery over Unidirectional Transport. RFC 3926 (Experimental), October 2004.

[7] BitTorrent. BitTorrent protocol specification [online]. 2003. Available from: http://www.bittorrent.org/protocol.html [cited 21 November 2006].

[8] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Felber, Al A. Hamra, and L. Garcs-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *Proceedings of PAM2004*, April 2004.

[9] wxWidgets. wxWidgets cross-platform GUI library [online]. 2006. Available from: http://www.wxwidgets.org/ [cited 21 November 2006].

[10] Boost. Boost C++ libraries [online]. 2005. Available from: http://www.boost.org/ [cited 21 November 2006].

[11] Xerces. Xerces C++ parser [online]. 2005. Available from: http://xml.apache.org/xerces-c/ [cited 21 November 2006].

[12] libtorrent. libtorrent C++ library [online]. 2006. Available from: http://www.rasterbar.com/products/libtorrent/index.html [cited 21 November 2006].

[13] MAD-FCL. MAD Project's Home Page [online]. 2006. Available from: http://www.atm.tut.fi/mad/ [cited 21 November 2006].

[14] cURL. cURL and libcurl [online]. 2006. Available from: http://curl.haxx.se/ [cited 21 November 2006].

[15] BNBT. BNBT EasyTracker [online]. 2004. Available from: http://bnbteasytracker.sourceforge.net/ [cited 21 November 2006].

[16] Apache. The Apache HTTP Server Project [online]. 2006. Available from: http://httpd.apache.org/ [cited 21 November 2006].

[17] gtkmm. gtkmm - C++ Interfaces for GTK+ and GNOME [online]. 2005. Available from: http://www.gtkmm.org/ [cited 21 November 2006].