Managing Peer-to-Peer Live Streaming Applications

Raymond Cunningham, Bartosz Biskupski, and René Meier

Distributed Systems Group,
Department of Computer Science,
Trinity College Dublin

Abstract. A number of p2p live streaming systems [1], [2], [3], [4], [5], [6] and [7] have been proposed in recent years. Typically, the description of these systems focuses on how the live stream is transmitted from its source to a number of viewers within the particular p2p network and how these systems deal with the failure of one or more viewers during transmission of the stream. An important aspect of each of these systems that is typically overlooked is how individual stream transmitters and viewers of these streams are managed in terms of registration, configuration and maintenance. In this paper, a set of management related abstractions common to many p2p live streaming systems are identified. This paper describes the MeshTV architecture, capturing these abstractions, to simplify the management of p2p live streaming applications. The architecture has been evaluated through a number of experiments and has been assessed against existing related work.

1 Introduction

A number of Peer-to-Peer (p2p) live streaming systems [1], [2], [3], [4], [5], [6] and [7] have been proposed in recent years. Typically the description of these systems focuses on how the particular live stream is transmitted, for example, whether it uses an underlying tree based or mesh based topology and/or the assumptions made about the underlying network infrastructure (such as updates to intermediate network level routers, etc).

In contrast, this paper focuses on an architecture and the corresponding abstractions needed to ease the deployment and ongoing management of an application-level p2p live streaming system. Our architecture does not make any assumptions about the underlying network layer routing infrastructure. In general, the abstractions common to p2p live streaming systems can be broken into a number of categories:

- Coordination
- Management
- Communication

The coordination category comprises a number of use cases such as the registration of participating peers (both a transmitter and one or more viewers)

which is typically not described in existing p2p live streaming systems. In addition, how the ongoing configuration of these participating peers is achieved is typically not described. An underlying assumption of most live streaming p2p overlay based systems is that the total system upload (i.e., the sum of the upload capacities at all peers) can be utilised correctly to enable all peers to download the stream at the stream rate at which the transmitter transmits. Thus, it is important for a running system to be able to verify this latter assumption.

The management category includes a number of different aspects of a p2p live streaming application such as the gathering of relevant statistics related to the live stream and the enforcement of a revenue model for a particular stream. These aspects are typically overlooked in favour of ensuring that the p2p live streaming system maximises a global system parameter such as the total upload utilisation. However, the particular system is typically unaware of what the total system upload capacity actually is.

The final abstraction covers the lower level management related issues of communicating the live stream from a transmitter to a number of interested viewers. Typically, how bootstrapping is achieved in most systems is not covered in detail as it is assumed to be a solved problem. However, the solution to this problem depends on the number of current peers in the system and the percentage of the peers that are joining and/or departing the system.

These abstractions and how they relate to the MeshTV architecture will be further elaborated in section 2. Section 2 is followed by a description in section 3 of a realisation of the lowest layer of the architecture and an evaluation of this realisation in section 4. These sections are followed by an assessment of related work and how this work fits into the MeshTV architecture in section 5. Finally, section 6 concludes the paper and discusses future work.

2 MeshTV Architecture

As illustrated in Figure 1, the MeshTV Architecture is broken into a number of layers corresponding to the categories identified in section 1. The upper most layer of the architecture is called the Stream Coordination layer and enables potential viewers of a particular stream to discover the stream (typically by browsing a list of streams) that a transmitter is transmitting or intends to transmit and for the particular peers that are providing the stream to be easily configured throughout the lifetime of the stream.

2.1 Coordination

Before a transmitter can begin transmitting a live stream, it is required to register its details (such as type of content, point of contact for the stream (i.e., underlying mesh/tree component endpoint), start time of transmission, etc) so that potential viewers of the stream can become aware of this stream. The registration of a particular stream occurs at a logically centralised stream manager that can record relevant details about the stream for later querying by interested

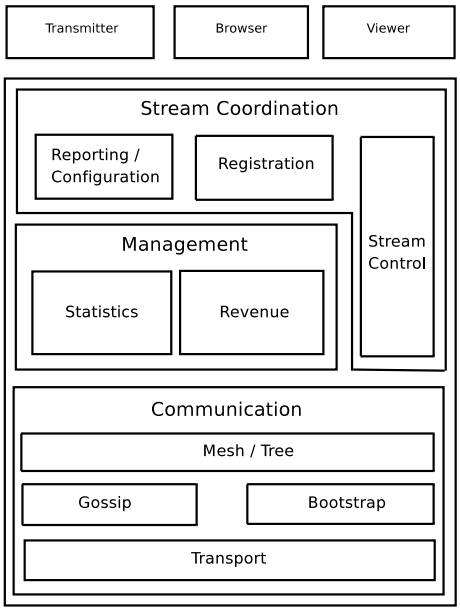


Fig. 1. MeshTV Architecture

potential viewing peers. Note that how to secure the registration of a transmitter (and a potential viewer) will not be addressed in this paper though a number of possible approaches exist such as [8] and [9].

As will be seen in the following sections, this logically centralised stream manager plays an important role in a number of the abstractions identified in this paper.

When the transmitter registers its information with the stream manager, the stream manager records information related to the transmission with the statistics management component and then generates a billing profile for the future transmission of the content by using the lower layer revenue management component. This profile may allow advertisements to be tailored to the content of a particular transmission or the transmitter may require a potential viewer to pay to watch its particular transmission.

In general, the configuration of the different components that constitute a live streaming application should be configured in one of two ways, depending on the number of peers in the topology. Firstly, when the number of nodes is below a known threshold configuration may be carried out directly by a particular peer such as the transmitter contacting each of the viewing peers directly to adapt their operation. For example, if the transmitter wishes to switch to a newer codec (to reduce bandwidth consumption) while transmitting using an older codec, the transmitter may contact each individual viewing peer aware of this change (assuming that the number of viewing peers is relatively small).

Secondly, if the number of viewing peers is above the threshold, configuration could be carried out using a gossip protocol ([10], [11], [12]) to spread the burden of configuration across all the peers in the system.

This dual mode of operation for the configuration of different aspects of a peers behaviour is important as existing live streaming systems do not highlight how such configuration is achieved and focus on the operation of these when a certain number of participating peers are in the system. The approach to configure/tune a live streaming system with a total of 100 peers can be very different from a system with 1,000 peers which in turn can be different from a system with 10,000 peers.

An important part of the configuration/reporting component of the MeshTV architecture is how bandwidth related information/statistics (such as total upload bandwidth utilisation) is reported. For example, the total upload bandwidth utilisation is an important global system property that existing mesh based systems and multi-tree systems attempt to maximise. A human manager or a realisation of the MeshTV architecture may use this bandwidth related information to change relevant protocol parameters to better adapt to the current dynamic environment within which the particular stream is being transmitted.

In a similar way to how system wide configuration is achieved, the collection of bandwidth related statistics would also have a dual mode of operation with direct point to point communication being used for the live streaming systems with a small number of peers and aggregation [13], [14], [15] (with an underlying gossip protocol) being using for larger systems. This second (and more complex) approach using a number of aggregation rounds may be initiated by any peer in the system gossipping an initial aggregation message to its set of neighbours that then gossip this message onto their neighbours. As the aggregation message containing the aggregated statistics propagates throughout the system, each peer on receiving the aggregation message updates these statistics with local information related to its bandwidth usage. These local bandwidth related statistics are maintained by the statistics component of the management layer.

2.2 Management

There are two main parts in the management layer of the MeshTV architecture. As briefly discussed in the previous sub-section, the statistics management component is responsible for aggregating important statistics related to the transmission of the stream such as, for example, the total number of peers, the total amount of bandwidth downloaded at all peers and the average neighbour degree of each peer throughout the system. This statistical information could be used locally by the peer or be used to inform other peers (such as joining peers) about the state of the system.

An additional capability of the statistics component is to build a model of how the lower communication layer uses the peers upload bandwidth as the stream is transmitted. This upload bandwidth model would depend on the characteristics of a particular peer's set of neighbours and may require the peer to learn

how much upload bandwidth it can currently offer to the overall live streaming system.

The second part of the management layer is the revenue component whose role is to ensure that a particular revenue model for the stream is enforced. There are a large number of possible revenue models for a stream that are possible such as one based on advertisements, a single advance up front payment or a split revenue model with the first number of minutes free followed by micro-payments for each subsequent minute of viewing.

Note that some of these revenue models require registration of the viewing peer while others may not. A transmitter may wish to change the revenue model of its stream based on the current demand with one revenue model being used at the beginning of a stream's transmission before being switched to another revenue model as demand increases.

The commencement of a stream is recorded with a local and/or remote statistics component before retrieving optional adverts from a local and/or remote revenue component. In the remote case, this information is recorded with the stream manager. The stream manager uses this information to provide a list of possible streams that can be viewed. Before returning this list of streams, the stream manager retrieves relevant statistics of the currently available streams (possibly using a profile of the viewing peer). Relevant statistics for a stream may include the current bandwidth consumed most recently by the stream, the current stream rate and number of viewers that are currently watching the stream.

In addition to the above statistics, the stream manager also retrieves details related to the cost of viewing each stream in the list of streams. This should allow different types of pricing to be achieved based on the type of viewer that is requesting the stream.

2.3 Communication

In this section, a number of lower level abstractions are presented that are related to the transmission and reception of the live stream.

After registering its details with the stream manager, the transmitter begins transmitting its content by sending it to the communication layer in the MeshTV architecture as illustrated in Figure 1. The mesh/tree component is considered as a decentralised component (which we consider as a single logical component for the purposes of this document) that manages the distribution of the stream content from the transmitter to the other viewing peers that are participating in the mesh/tree. Note that this distribution can be done over either a tree-based or mesh-based topology such as those covered in section 5.

Sending the content to the underlying topology requires that the mesh/tree component has a list of other peers to which it can transmit. When a transmitter begins to transmit the stream, the mesh/tree component of the transmitter contacts the bootstrap component of the communication layer to initialise a new mesh or tree for the transmission that is about to start. This in turn results in a mesh or tree peer being created that represents the transmitter on the underlying

topology. The mesh/tree component (on the transmitter) then initialises its list of neighbouring nodes to be empty.

It is possible for the above steps to happen when a peer indicates its intention to transmit a stream during registration. This allows the underlying topology to be setup with an initial set of viewing peers before the stream begins transmission and possibly reduces the latency at the beginning of the transmission. This bootstrap component is considered as a logically centralised component that could be hosted or maintained by the transmitter or could be distributed across a number of peers.

When a MeshTV viewer chooses a particular stream from the stream manager, the stream manager records the addition of a new viewer with the statistics manager and then verifies with the revenue component that the viewer has the requisite credentials to view the stream. Thus depending on the type of stream that is of interest, the viewer may or may not need one or more components (such as the registration or revenue components). In a similar way to the transmitter initiating the mesh/tree component for the transmission of its content, the mesh/tree component (on behalf of a viewer) must also join the mesh/tree by utilising the services of the local bootstrap component. Firstly, the mesh/tree component sends a request to join the topology (previously created by the transmitter) to the bootstrap component for that topology which in turn results in the creation of a new peer on the topology that represents the joining viewer. As a result of this request, the mesh/tree component receives a number of potential neighbours that it then uses to initialise its neighbourhood.

A common abstraction that a number of live streaming systems use is that of a neighbourhood of peers that an individual peer uses to transmit the stream to and/or to receive the stream from. In the MeshTV architecture, the neighbourhood component provides functionality to ease the burden on a particular peer of maintaining a set of neighbouring peers such as providing one or more techniques for the detection of a failed neighbouring peer. In addition, the neighbourhood component can easily maintain a profile of the communication (bandwidth) capacity of each of its neighbouring peers which can then be used to inform the decision making of the mesh/tree component.

Finally, the neighbourhood component can be used in conjunction with the gossip component to provide the capability to communicate with all the peers in a large-scale system. This communication could be achieved over a specialised random mesh network or using the existing topology that is already in use by the live stream.

3 MeshTV Peer-to-Peer Protocol

In this section we show how the communication component of the MeshTV architecture can be realised using an existing p2p protocol. Complete details of this p2p protocol are available in [16]. In the following subsections we present each communication subcomponent in our system.

3.1 Mesh

The mesh overlay is formed by joining peers connecting to randomly selected neighbours and then periodically refining them using an exploration algorithm. Each peer maintains two sets of neighbours - receivers, which are the neighbours to which it uploads data and senders, which are the neighbours from which data is downloaded. The transmitter splits the data stream into small data chunks, which are exchanged between neighbouring peers in an epidemic fashion. Peers maintain local knowledge about data chunks possessed by their senders and inform receivers whenever they receive (or generate, in case of the transmitter) a new data chunk. Whenever a sender of a peer notifies it about a newly received chunk, the peer requests this new chunk if it has not requested this chunk from another peer already.

The exploration algorithm is used to adapt the overlay to optimise the video streaming throughput by maximising the utilisation of available upload bandwidth of peers, which we consider the most scarce resource in the system. The algorithm is executed by each peer independently and its goal is to adapt the peer's set of senders to improve the download rate. The algorithm is executed by a peer periodically in a series of rounds and ensures that:

- A peer has a constant (configurable) number of senders.
- A peer replaces the sender from which it receives the worst download rate with a new sender provided by the bootstrap component that selects it randomly from all peers in the overlay.

The exploration algorithm adapts the mesh overlay so that *(i)* the upload bandwidth of all peers is efficiently utilised, *(ii)* download rates of nodes are improved and *(iii)* network latency between interacting peers is reduced. Upload bandwidth is utilised by matching a peer's number of receivers with its available upload bandwidth. The reason for this is that a peer continues to gain new receivers when it is underloaded and loses some receivers (i.e., receivers replace it with less loaded senders) when it is overloaded.

A peer joining the network initially acquires a random set of senders from its bootstrap component. The exploration algorithm will then continuously attempt to improve the peer's download rate by replacing the slowest senders with senders that can provide higher transfer rates, thereby effectively optimising its set of senders. This approach also decreases the network latency between neighbouring peers as a consequence of using TCP to transmit data chunks, and TCP's built-in congestion control. The reason for this is that when multiple connections share an overloaded link, TCP allocates more bandwidth to connections with lower network round-trip times (RTT) [17]. When a bottleneck occurs at the sender's uplink, more upload bandwidth is allocated to receivers with low latency. Similarly, when a bottleneck occurs at the receiver's downlink, more download bandwidth is allocated to senders with low latency. This causes receivers to replace distant senders (for which TCP allocates less bandwidth) with senders that are potentially closer.

3.2 Bootstrap

The MeshTV bootstrap component is used by the mesh component to provide a random sample of peers (potential neighbours) when a peer joins the system and whenever the exploration algorithm is executed. Bootstrap uses the gossip component, described below, to periodically obtain a new sample of peers.

3.3 Gossip

The gossip component in MeshTV is based on a peer sampling service [18] in which peers randomly exchange membership information between themselves. This results in each peer periodically obtaining a random subset of all peers in the system, which are then provided to the bootstrap component and are used to create and refine the mesh overlay.

3.4 Transport

The MeshTV p2p protocol uses TCP for transferring data chunks between peers. The use of TCP as a transport protocol enables the exploration algorithm to improve proximity between neighbouring peers as described in section 3.1.

4 MeshTV Evaluation

In this section, we present an initial evaluation of the communication layer of the MeshTV architecture encompassing the previously highlighted p2p protocol [16]. This evaluation was carried out in ns-2 [19]. In particular, we show that the p2p protocol optimises upload bandwidth utilisation, improves throughput and network proximity between neighbouring peers by using the combination of the mesh, bootstrap, gossip and transport components.

Ns-2 [19] provides a realistic model of the physical network and the TCP/IP stack (the MeshTV p2p protocol uses TCP New Reno) at the cost of reduced scalability limiting the number of nodes that have been simulated to 500. Our previous experience in evaluating larger overlays in less accurate flow-level simulators lead us to believe that the findings presented here are also valid for larger overlays [7]. The physical network topology created for simulations is a full mesh with bandwidth being limited on the access links (uplinks and downlinks). The node bandwidth distribution has been derived from Gnutella p2p system measurements [20] and nodes have been categorised into 4 groups: A, B, C and D (see Table 1). Network latencies between nodes are selected uniformly at random between 2ms and 300ms. MeshTV parameters used in the experiments are presented in Table 2. The mesh overlay in the experiments is initially random, formed by nodes selecting random senders.

Category	Downlink	Uplink	Ratio
A	10 Mbps	5 Mbps	15%
B	3 Mbps	1 Mbps	25%
C	1.5 Mbps	384 Kbps	40%
D	784 Kbps	128 Kbps	20%

Table 1. Node bandwidth distribution

Parameter	Value
stream rate	1500 Kbps
number of senders	5
exploration round length	5 sec
chunk size	4 KB
pipelined requests	8
sliding window size	30 sec

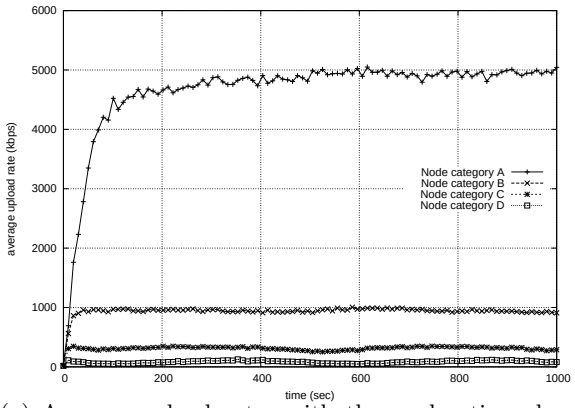
Table 2. Protocol parameters

4.1 Upload Utilisation

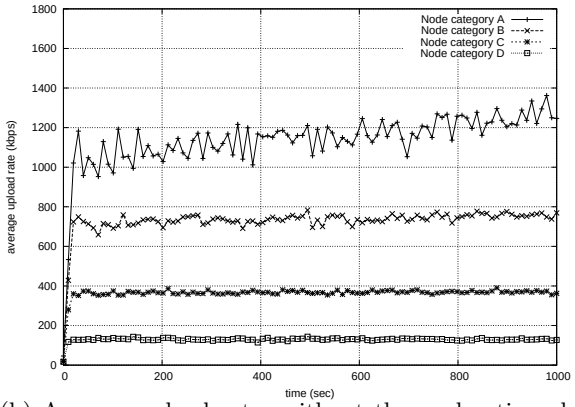
Figure 2 compares the average upload utilisation of the overlay with and without the adaptation (note the different scales on the y-axes). It shows that when the exploration algorithm is used, a node's upload reaches its maximum upload capacity as shown in Table 1. This means that nodes in all categories fully utilise their upload bandwidth. In contrast, when the exploration is not used, the upload bandwidth of nodes in the highest categories A and B is greatly underutilised. It can be observed from these figures and the given node bandwidth distribution that the total aggregated upload for adapted and not adapted overlays is about 550 Mbps and 260 Mbps respectively. This means that the upload bandwidth utilisation is improved by over 100% when the overlay is adapted by the exploration algorithm.

4.2 Throughput

The improved utilisation of the upload bandwidth results in nodes increasing their data throughput. Figure 3 compares the data rates with and without the exploration algorithm (note again the different scales on the y-axes). It shows that the data rates received in the adapted overlay are all much higher than in the case of a random mesh overlay. However, not only the upload bandwidth of senders, but also a node's own download capacity limits the received data rate. So, for instance, the download rate of nodes in category D is limited by their download bandwidth of 784 Kbps. Other node categories have higher download capacity and thus achieve higher download rates. MeshTV accommodates limited download bandwidth of some nodes and different data rates received by different node categories through the use of the Multiple Description Coding (MDC) technique and specifically MDC-FEC [21]. The MDC technique enables



(a) Average upload rates with the exploration algorithm



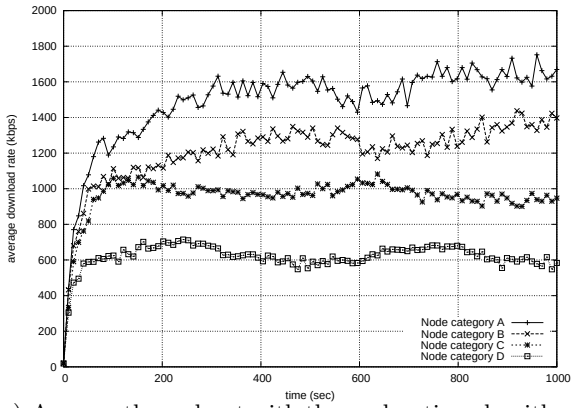
(b) Average upload rates without the exploration algorithm

Fig. 2. Optimising average upload rates.

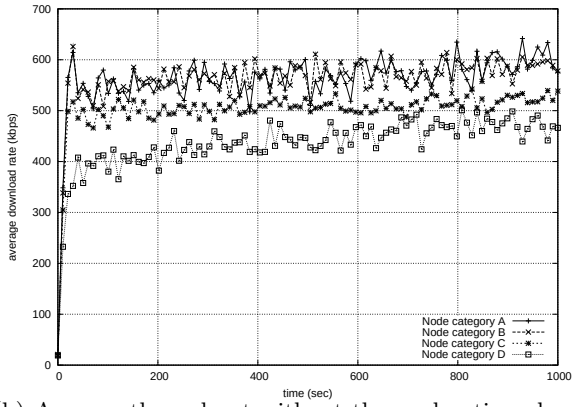
the original video stream to be split into a number of descriptions. A node can download any subset of all descriptions to recreate the video stream.

4.3 Node Proximity

Figure 4 shows how the exploration algorithm reduces the network latency between interacting nodes. Initially, the random mesh overlay has an average latency between neighbouring nodes roughly equal to 151ms as the latencies are assigned randomly between 2ms and 300ms. Since senders allocate more upload bandwidth to closer receivers, the overlay adapts, resulting in a reduction of the average latency to about 75ms, which is a 50% improvement. The exploration algorithm does not further reduce the distances between neighbouring nodes as this might degrade the data throughput. Connecting exclusively to the nearest



(a) Average throughput with the exploration algorithm



(b) Average throughput without the exploration algorithm

Fig. 3. Improving the average throughput.

senders implies two undesired effects. Nodes that share low-latency links with many other nodes might be overloaded and the overlay might be divided into disconnected clusters of nearby nodes. The exploration algorithm prevents these unwanted effects as it improves proximity only when this does not degrade the data throughput. This is because a high-latency underloaded node will provide higher data throughput than a low-latency overloaded node and thus will be preferred as a sender.

5 Related Work

As would be expected from a number of research project solutions in the domain of p2p live streaming (such as Bullet [2], Splitstream [1], Chainsaw [3],

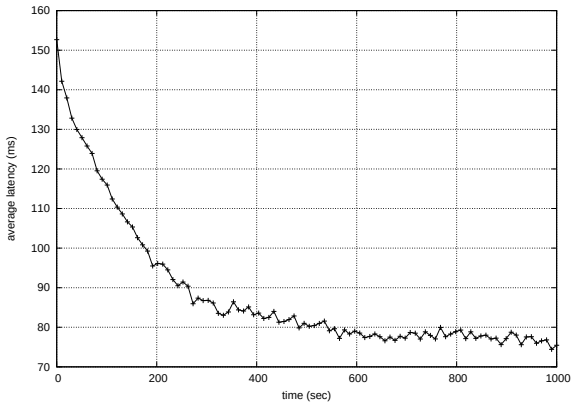


Fig. 4. Improving proximity of neighbours.

Coolstreaming [6] and MeshCast [7]) these solutions has focussed on the challenging distributed system problems of achieving low latency and robustness in such a dynamic environment and not been on the management of streams or the generation of revenue.

Most of these systems are mesh based with the exception of Splitstream which uses a tree and Bullet which uses a hybrid approach of a tree for dissemination of the stream and a mesh for retransmission of lost or dropped packets.

A number of the systems use a form of gossip in the construction of their underlying topologies. For example, Bullet executes the RanSub algorithm [22] to deliver a uniform random subset of peers to each peer in the system. Coolstreaming also uses a gossip protocol to achieve membership management. MeshCast on the other hand uses a variant of the Newscast algorithm [12] to distribute a sampling of the peers in the system. The use of bootstrapping is not stressed in any of the descriptions of these systems though it is typically needed to enable the correct functioning of each system in the presence of new peers joining and existing peers departing.

Finally, a number of these systems attempt to model the communication capability of a neighbouring peer so that the overall system can better utilise the bandwidth (typically upload bandwidth) at all nodes. Bullet builds on top of TCP Friendly Rate Control [23] (with each peer periodically evaluating its senders and receivers and dropping or replacing them if they do not provide sufficient bandwidth to that peer. In coolstreaming, a peer estimates the performance of its neighbours to guide the scheduling of requests for parts of the stream. MeshCast also attempts to estimate its neighbouring peers upload capacity using its sender/receiver balancing algorithm.

6 Conclusions/Future Work

This paper described the MeshTV architecture which incorporates a set of abstractions that are common to a number of existing live streaming systems. This layered architecture divides these abstractions into three categories which have been labelled coordination, management and communication. A realisation of this architecture incorporating an existing p2p protocol was implemented in ns2.

Work is currently ongoing to provide an implementation of the MeshTV architecture (outside of the ns2 simulator) that incorporates one or more existing live streaming systems. On completion, this will enable a further evaluation of our architecture including the performance advantages and impact of using different higher layer components such as the statistics managment component. It will also be possible to investigate the performance impact on a set of peers that are executing two or more live streaming systems (e.g., as part of a wireless access point based home entertainment system hub).

7 Acknowledgements

This work was partly funded by the "Information Society Technology" Programme of the Commission of the European Union under research contract IST-507953 (DBE) and by Enterprise Ireland under the Commercialisation Proof of Concept Programme (MeshTV).

References

1. Castro, M., Drushel, P., Kermarrec, A., Nandi, A., Rowstron, A., Singh, A.: "Splitstream: High-Bandwidth Multicast in Cooperative Environments.". In: SOSP. (2003)
2. Kostic, D., Rodriguez, A., Albrecht, J., Vahdat, A.: "Bullet: high bandwidth data dissemination using an overlay mesh.". In: "Symposium on Operating System Principles.". (2003)
3. Pai, V.S., Kumar, K., Tamilmani, K., Sambamurthy, V.: "Chainsaw: Eliminating trees from overlay multicast.". In: IPTPS. (2005) 127–140
4. Jannotti, J., Gifford, D.K., Johnson, K.L., Kaashoek, M.F., O'Toole, J.: "Overcast: Reliable multicasting with an overlay network.". In: OSDI. (2000)
5. Castro, M., Drushel, P., Kermarrec, A., Rowstron, A.: "SCRIBE: A large-scale decentralised application level multicast infrastructure.". IEEE JSAC (2002) 1–9
6. Zhang, X., Liu, J., Li, B., Yum, T.S.P.: Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming (2005)
7. Biskupski, B., Cunningham, R., Dowling, J., Meier, R.: High-bandwidth mesh-based overlay multicast in heterogeneous environments. In: AAA-IDEA '06: Proceedings of the 2nd international workshop on Advanced architectures and algorithms for internet delivery and applications, New York, NY, USA, ACM (2006) 4
8. Hiclanan, K., Elgamal, T.: The SSL protocol. Internet draft, Netscape Communications Corp. Technical report (1995)

9. Kohl, J.T., Neuman, B.C.: The Kerberos network authentication service (V5). Technical Report 1510 (1993)
10. Jelasity, M., Babaoglu, O.: "T-man: Gossip-based overlay topology management.". In: 3rd International Workshop on Engineering Self-Organising Applications. (2005)
11. Rao, A., Lakshminarayanan, K., Surana, S., Karp, R., Stoica, I.: "load balancing in structured p2p systems". In: 2nd International Workshop on Peer-to-Peer Systems. (2003)
12. Jelasity, M., van Steen, M.: Large-scale newscast computing on the Internet. Technical Report IR-503, Department of Computer Science Vrije Universiteit, Amsterdam, The Netherlands (2002)
13. Jelasity, M., Montresor, A.: "Epidemic-style proactive aggregation in large overlay networks.". In: Proceedings of the 24th International Conference on Distributed Computing Systems. (2004) 102–109
14. Jelasity, M., Montresor, A., Babaoglu, O.: "robust aggregation protocols for large-scale overlay networks.". In: International Conference on Dependable Systems and Networks. (2004) 19–28
15. Kempe, D., Dobra, A., Gehrke, J.: "Gossip-based computation of aggregate information". In: 44th IEEE Symposium on Foundations of Computer Science. (2003) 482–491
16. Biskupski, B., Cunningham, R., Meier, R.: Improving throughput and node proximity of p2p live video streaming through overlay adaptation. In: Proceedings of the 9th IEEE International Symposium on Multimedia (ISM 2007), Los Alamitos, CA, USA, IEEE Computer Society (2007) 245–252
17. Lakshman, T.V., Madhow, U.: The performance of TCP/IP for networks with high bandwidth-delay products and random loss. IEEE/ACM Trans. Netw. **5**(3) (1997) 336–350
18. Jelasity, M., Guerraoui, R., Kermarrec, A.M., van Steen, M.: The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In: Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, New York, NY, USA, Springer-Verlag New York, Inc. (2004) 79–98
19. McCanne, S., Floyd, S.: ns—Network Simulator. http://www.isi.edu/nsnam/ns
20. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. In: Proceedings of Multimedia Computing and Networking. (2002)
21. Goyal, V.K.: Multiple description coding: Compression meets the network. IEEE Signal Processing Magazine **18**(5) (September 2001) 74–93
22. Kostic, D., Rodriguez, A., Albrecht, J., Bhirud, A., Vahdat, A.: Using random subsets to build scalable network services. In: Proceedings of 4th USENIX Symposium on Internet Technologies and Systems (USITS). (2003)
23. Floyd, S., Handley, M., Padhye, J., Widmer, J.: Equation-based congestion control for unicast applications. In: SIGCOMM 2000, Stockholm, Sweden (August 2000) 43–56