# DAg-stream: Distributed Video Adaptation for Overlay Streaming to Heterogeneous Devices

Razib Iqbal, Shervin Shirmohammadi

*School of Information Technology and Engineering (SITE), University of Ottawa*

*800 King Edward Avenue, Ottawa, Ontario, K1N 6N5, Canada*

[riqbal |shervin]@discover.uottawa.ca

## Abstract

Combining the advantages of Peer-to-Peer (P2P) content distribution concept and metadata driven adaptation of videos in compressed domain, in this paper, we propose a simple but scalable design of distributed adaptation and overlay streaming using MPEG-21 gBSD, called DAg-stream. The objective is not only to shift the bandwidth burden to end participating peers, but also to move the computation load for adapting video contents away from dedicated media-streaming/adaptation servers. It is an initiative to merge the adaptation operations and the P2P streaming basics to support the expansion of context-aware mobile P2P systems. DAg-stream organizes mobile and heterogeneous peers into overlays. For each video, a separate overlay is formed. No control message is exchanged among peers for overlay maintenance. We present a combination of infrastructure-centric and application end-point architecture. The infrastructure-centric architecture refers to a tree controller, named DAg-master, which is responsible for tree/overlay administering and maintenance. The application end-point architecture refers to video sharing, streaming and adaptation by the participating resourceful peers. The motivation for this work is based on the experiences and lessons learned so far about developing a video adaptation system for heterogeneous devices. In this article, we present our architecture and some experimental evaluations supporting the design concept for overlay video streaming and online adaptation.

*Keywords: H.264, MPEG-21 gBSD, Overlay streaming, Video adaptation*

# 1. Introduction

Modern wireless networks support media rich applications such as video messaging, video telephony, video portals, video group communication, and real time video streaming. In recent years, there has been significant interest in the use of Peer-to-Peer (P2P) technologies for Internet video broadcast/streaming. However, the sparse use of IP multicasting and the high cost of bandwidth and maintenance required for server-based solutions are the two main factors that only allow limited resource providers to broadcast media rich contents to the end-users over the internet.

The number of handheld devices capable of playing multimedia contents has also increased significantly. In addition, entertainment devices with built-in Wireless Fidelity (Wi-Fi), for example, iPOD, PSP etc. are making it easier to access online resources and video streams. However, functional limitation of these devices restricts them to become an autonomous node in a P2P system. This limitation also makes it difficult to render and display regular videos on these devices like that of desktops or laptops. This drives the recent research enthusiasm for efficient application side P2P video streaming over the internet addressing the Universal Multimedia Access (UMA).

In this paper, we emphasize on structured metadata based adaptation of live (non-interactive) video streams and streaming. Our scheme facilitates the cooperation of participating clients for improving the utilization of the spare resources in an overlay. With peers contributing CPU cycles and bandwidth to adapt and stream videos respectively, we aim to relieve the need for streaming or adaptation servers.

## 1.1. Motivation

Heterogeneity in multimedia enabled devices grows in terms of screen resolution, processing power, and available bandwidth. For example, mobile phones typically have a smaller screen and lower bandwidth, e.g., through Global Packet Radio Services (GPRS), while PDAs have higher resolution screens and can have access to broadband connections such as Wi-Fi. Inevitably, each of these devices has their own set of distinct combination of features which restrain their access to different video contents in an interoperable manner. In a simple scenario, temporal adaptation is required to meet the diversity of network types that requires flexible media contents to fit network pipes of different bandwidth. Now, the actual motivation behind this work is that with the modern computing capabilities, a single peer can carry out sufficient adaptation operations and stream media contents to other peers. A media streaming server based video broadcast system typically has a single dedicated source which may be assumed failsafe and is present throughout a broadcast session. However, high requirement of dedicated bandwidth to serve all heterogeneous client requests is an expensive approach. Therefore, the focus of this paper is on simultaneous video adaptation and streaming in a P2P overlay. By distributing the workload to low-cost,

off-the-shelve computing hosts such as personal computers (PC) and workstations, one can eliminate the need for costly centralized adaptation and streaming servers and at the same time, improve the system's scalability and availability. Our transcoding solution [1] using MPEG-21 generic Bitstream Syntax Description (gBSD) does not require a dedicated media adaptation server; rather, any intermediary node[1] can adapt the video on the fly. Moreover, raw video frames can be encoded to H.264 video along with the gBSD generation in real time.

## 1.2. Contributions

The main contribution of this paper is the incorporation of a video adaptation scheme into a classical P2P streaming system. We propose to utilize the available computing power of the participating peers to serve those peers requiring adapted videos, by adapting the video on the fly. It is a lightweight design where core operations are simple to implement and easy to deploy as a client-server application. Part of our contribution is to avoid the definite presence of adaptation server(s), used in many of the current systems to adapt the media for specific heterogeneous devices, since such servers imply additional resources as well as causing bottlenecks. Adaptation is therefore performed by the peers themselves, and the overlay tree is constructed with the adaptation requirements in mind. The benefit of this approach is that it will help to avoid or at least significantly reduce the deployment of dedicated adaptation server(s) in the streaming architecture. Equally important, this will allow heterogeneous devices including small handhelds the opportunity to participate in the P2P system. Another part of our contribution is that, in this design, we free small hand-held devices from any adaptation operation and tree maintenance (a realistic and practical choice considering their limited resources) and allow them to join the overlay tree just to receive the adapted stream. Since these nodes are not able to perform either adaptation or serve another client, therefore a special care should be taken in order not to saturate the system by blocking outgoing links. These nodes are denoted as '*free riders*'. Such similar architectures have not been used in the literature before except [2]. We use a simple yet scalable overlay design, and then organize video streaming sessions among the participating peers following a master-sender-driven approach with the help of cooperative peers. Our proposed scheme rely exclusively on shared-bandwidth and CPU resources at application endpoints i.e. peers. We have considered H.264 Advanced Video Codec (AVC) and MPEG-21 multimedia framework (Part -7: Digital Item Adaptation) [3] for compressed domain adaptation. While constructing the overlay, instead of using fixed out-degree for each node, we have computed out-degree dynamically based on the computing power and the sending throughput. It gives a better indication of resource availability and also makes our architecture realistic.

---

[1] The term "node" is used interchangeably with "peer" and "client" throughout this manuscript

### 1.3. Organization of the article

The rest of this article is organized as follows: Section 2 describes the different works that have been carried out on designing first generation P2P systems. A brief overview of MPEG-21 gBSD and gBSD-based adaptation is mentioned in Section 3. Section 4 provides the design of the proposed system where we cover how the gBSD-based adaptation can be applied to P2P streaming, and then focus on the overlay formation and different components of DAg-stream. Section 5 presents the performance metrics, simulation setup and performance evaluation of the design. Section 6 is devoted to a discussion of some issues relevant to the proposed system. Finally, we draw the conclusion in Section 7.

## 2. Related Work

Although in theory IP multicast is the most efficient multicasting vehicle, its deployment remains limited due to the unsolvable technical, management, and business issues which have prevented service providers to give multicast support to Internet users at homes. As remedies to unavailability of IP multicasting, research pioneers have been focusing on application layer multicasting for at least the past ten years [4]. In literature, excellent research entailing ALM-based solutions include NICE [5], ZIGZAG [6] to name a few. Most of these works however focus on protocol design and do not consider adaptation to meet peer heterogeneity.

The most common approach for P2P streaming is to construct one or several multicast trees to distribute the stream between the source and different users, e.g. [7][8]. Another approach lets the peers self-organize in a mesh and request different portions of the video from their neighbors, with no particular emphasize on the structure of the distribution path, e.g. [9]. Error resilient transport for P2P video streaming is covered in [10]. In [10], authors focused on broadcast categories of robust video streaming schemes like forward error correction, prioritized automatic repeat request etc. To meet the demand of P2P video streaming, many first generation applications have appeared on the internet, such as PPLive (*www.pplive.com*), PPStream (*www.ppstream.com*) etc. However, their protocol design and encoding structures have not been targeted at peer heterogeneity. Moreover, their unstable video quality does not offer a high-quality viewing experience. Tan et al's work [11] focuses on network delays between source and receiver while building multicast trees for multimedia streaming. However, for different scenarios mentioned in this paper, if we consider the latency added during video capture, encoding and adaptation, the network delay is very small and less important than adaptation requirements or resource limitations.

In a traditional, decentralized design, peers communicate directly with each other for the existence, sharing and exchange of data as well as other resources such as bandwidth. Problem with the decentralized solution is that a node may overwhelm other neighboring nodes (e.g. AMMO [12]).

Layered encoded video streaming has also received a significant research interest in the multimedia community in the last couple of years. PALS [13], for example, is a receiver-driven approach for quality adaptive playback of layer encoded streaming media from a group of congestion controlled sender peers to a single receiver peer. Since, in layered encoding, only a limited number of layers can be stored for each video, end nodes might get less quality than what they can handle. The work by Xiaofeng et al. [14] is one of the few video streaming systems that uses ordinary computers (peers) as servers rather than using a single or a few dedicated servers. In [14], each video is first encoded into multiple descriptions and then each description is placed on a different server. The solution is good if there is some collaboration between peers before the streaming starts but it is inconvenient for an isolated peer to initiate the streaming session and to place the video descriptions in different participating nodes. Finally, G. Exarchakos and N. Antonopoulos in [15] have proposed an unstructured P2P overlay to share the network resources. Their design places dedicated servers and utilizes peers to reduce server load, whereas our goal is to avoid dedicated servers (for adaptation and streaming) and utilize the idle resources of the participating peers.

From our own experience [1] and from [10], we can draw the conclusion that significant gains can be obtained when systems are capable of adapting the content based on the encoding structure but in an independent way. More details on the necessity and benefits of adaptation in mobile environments can be found in [16]. In the literature, there are prominent architectures for P2P streaming solutions but none of them completely overcomes the challenges from the dynamic P2P environment. Some of these challenges arise due to peer heterogeneity and peer mobility. To solve these challenges, simultaneous adaptation and streaming of video contents in a distributed manner is therefore a new concept. In the literature, there are not many promising works in this topic except the PAT system [2], proposed by Liu et al.

In the PAT system, authors suggest saving the intermediary transcoding results as meta-data and re-use the metadata for a later transcoding request. The incentive behind PAT and ours are similar, but their proposed scheme is substantially different from ours in a number of aspects: 1) Our system does not need meta-data overlay, because transformation effort of the gBSD to adapted gBSD is very small compared to that of bitstream. 2) Without a standard metadata support, variety of adaptation approaches in a distributed multimedia environment makes adaptation procedures incompatible and unacceptable. Our system provides a complete solution to adaptation and streaming utilizing MPEG-21 gBSD for H.264 video. gBSD provides the flexibility to adapt video contents even outside the streaming system. 3) The presence of backup parents is not needed in our design because the tree controller re-assigns a parent when the current parent leaves the system. In PAT, authors report that meta-data takes up to 6% additional bandwidth and metadata size is 20-25% of the original video file size for frame rate

transcoding. In contrast, gBSD takes up to only 2-5% of the compressed bitstream and gives details of the bitstream.

Utilizing the MPEG-21 gBSD for distributed adaptation in an overlay for video streaming is a new concept, and no previous study has been reported till date except [17] and [18]. In [18] an analytical approach based on 0-1 Integer Linear Programming method is used to model the system and to calculate the optimum overlay to serve peers by a single parent. In this paper, we elaborate our initial design [17] of adaptive video streaming for mobile/heterogeneous devices. We introduce a dual parent approach combined with a practical way to measure peer CPU power and performance analysis incorporating batching.

# 3. MPEG-21 gBSD and gBSD-based Adaptation

Lack of interoperability among advanced multimedia packaging and distribution applications motivated the initiatives of the Motion Picture Expert Group (MPEG) to start working on the definition of enabling normative technology for the multimedia contents tagged as MPEG-21. Part 7, Digital Item Adaptation (DIA), of the MPEG-21 framework [3] specifies the syntax and semantics of tools that may be used to assist adaptation of a Digital Item. In this section, we provide a brief overview of MPEG-21 gBSD and gBSD based adaptation technique. However, we refer interested readers to [1] for details on the gBSD-based adaptation.

To perform adaptation in an intermediary node, it is unquestionably advantageous to have an adaptation engine that is not aware of any specific video codec. For this reason, in MPEG-21 framework, an XML based generic approach is referred for manipulating bitstreams. XML is used to describe the high level structure of the bitstream, and the resulting XML document is called a Bitstream Syntax Description (BS Description, BSD). This BS Description is not to replace the original binary format of the resource but rather acts as a metadata. With such a description, it is then possible for a resource adaptation engine to generate an adapted bitstream. Now, if the adaptation takes place on some intermediary nodes like gateways and proxies, then a codec independent schema is more appropriate. Therefore, a generic Bitstream Syntax Schema (gBS Schema) is specified in the framework. This schema ensures codec independence, semantically meaningful marking of syntactical elements, and hierarchical descriptions of the bitstream. A description conforming to gBS Schema is called generic Bitstream Syntax Description (gBSD). The video bitstream along with its gBSD is denoted as Digital Item (DI). DI is the basic content on the delivery path. gBSD, in the form of XML, is used to identify the segments in the compressed bitstream to perform adaptation operations in intermediary nodes. In this regard, at first, gBSD is

transformed to adapted gBSD by means of XSLT [19]. Finally, adapted bitstream is generated by discarding the gBSD portions corresponding to specific frames, and updating necessary information.

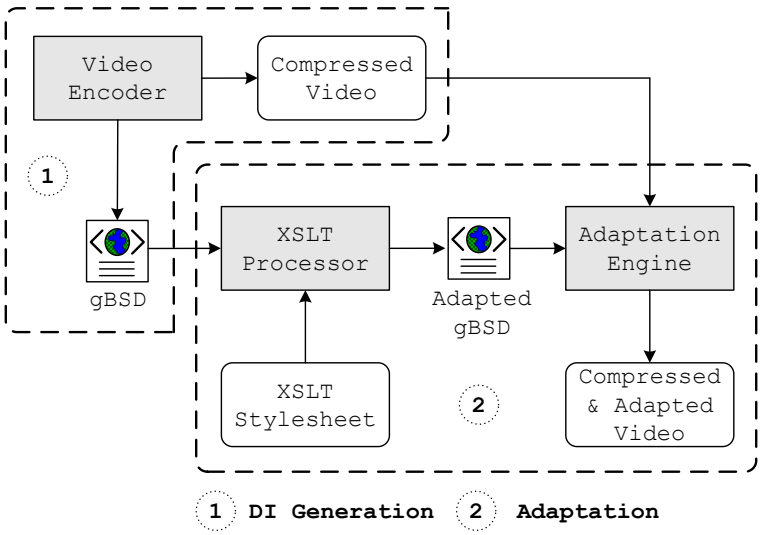


Figure 1. gBSD-based adaptation architecture

Figure 1 portrait the compressed domain adaptation approach. Part-1, DI Generation, is done during the encoding of the video stream. Part-2, Adaptation, is performed in some intermediary node, such as a peer in our case. In this method, small handheld devices are free of any adaptation operations, so they will not require the MPEG-21 engine.

**gBSD-based adaptation of continuous streams:** For XSLT transformation of gBSD, complete XML description of a video file need to be loaded before being adapted. Now, if we generate the gBSD of a long video file then it is inconvenient to transmit the gBSD to an intermediary node for future adaptation. Therefore, we recommend video streams to be processed as small segments, and gBSD for each of these segments will be generated accordingly. Benefits of this approach are as follows:

- For variable bit rate in the transmission channel, modified bit rate requirements can be applied to the next available video segments.
- The intermediary nodes do not need to save the video segments for a long time and may discard right after re-transmitting and/or adapting to other user(s).
- If the network characteristics or user preference changes during a streaming session, the master can assign a new parent who can serve this peer accordingly.

# 4. Overall Design

Given global knowledge of the participating nodes and plentiful of bandwidth and computing power, designing the overlays would be relatively straightforward. However, in real scenario, designs are constrained by the limited bandwidth and computing power available with the peers.

DAg-stream organizes receivers into multiple sub-trees based on a hierarchy of adaptation and streaming requirements. To build the overlay atop this hierarchy, a set of rules are defined. The design enables a separate overlay for each video. DAg-overlays have a central rendezvous point - the tree controller, called DAg-master. DAg-master is responsible for node joining and maintaining a multicast tree for each video stream. In this system, either a media server or a participating peer, denoted as DAg-starter, initiates the streaming and forms an overlay with the help of the DAg-master. In addition to the desired content, the DAg-starter should be able to adapt the content and provide overall streaming service to the immediate receiver peers in the tree. As described in Section 3, the streamed content also has a corresponding metadata component in the form of MPEG-21 gBSD. Now, to make a successful P2P streaming overlay, efficiency of the overlay is directly dependent on users' contributing bandwidth and CPU power. In our design, we have made the following assumptions:

- We assume that a participating node will share its bandwidth as well as CPU power in the presence of some incentives, e.g. peers with higher share-ratio[2] will get priority to join an overlay.
- Each end host has limited interface bandwidth and CPU power, which eventually constrains the overall out-degree of that peer.
- There is a tree controller whose address is known in advance. This acts as the rendezvous point and assumed not to fail.
- The address of the video source, i.e. DAg-starter is known to the tree controller only, and the controller itself does not initiate or provide any video stream.

## 4.1. Overview of the adaptation approach in the overlay

We consider the adaptation system/approach described in [1] as a utility tool. In this section, we shall briefly explain using this tool how video streams can be adapted, transmitted and buffered to match the varying network condition in a P2P overlay. For ease of understanding, in Figure 2, a coded video sequence is shown. As it can be seen, the video is divided into clips (Ref. gBSD-based adaptation of

[2] Share-ratio is calculated from the total amount of data uploaded over total amount of data downloaded

continuous streams), and framesets[3]. The chain of temporal dependencies between the frames[4] is also shown.

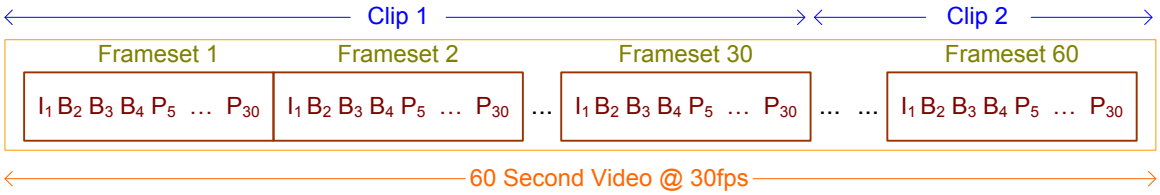


Figure 2. Preparing video for temporal adaptation and streaming

When a peer joins the overlay, it sends its available bandwidth and CPU power to the controller. For implementation convenience, we have mapped CPU power to adaptation time (to be described later). Based on this information, controller assigns this peer 'adaptation and streaming peer' role or 'streaming peer' role. We denote the peers capable of both adaptation and streaming, or only streaming, as regular peers. Devices with limited resources are free of any streaming or adaptation operations and may join as a free rider. Mobile peers are those peers who are not physically static. They can be a regular peer or a free rider.

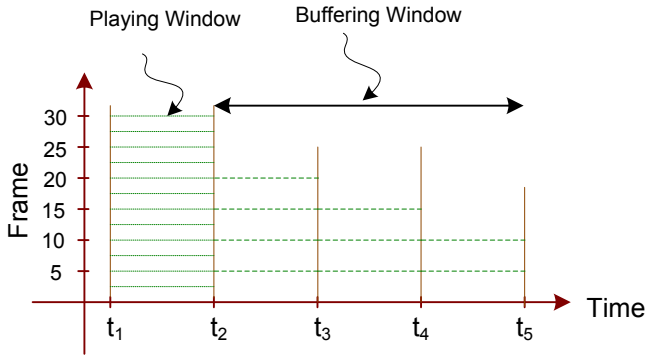


Figure 3. Buffering on the receiver's side to smoothen live streaming

**Buffer size:** To smoothen live streaming, the receiver needs a buffer (usually in the order of seconds) to store the clips for later use or immediate playback. As we proceed with the framesets, in each second the decoder and player deals with the number of frames defined in each frameset (i.e. 1 – 30 frames). Since the live video is processed in small clips, a loosely coupled synchronization between peers is required. More specifically, an initial buffering is needed which is filled up with the received video segments for immediate playback and retransmissions to another node (see Figure 3).

---

[3] 'Frameset' refers to the number of frames equal to the frame rate at which the raw video is being encoded (usually 30fps). After adaptation for 15frames per second (fps), there will be 15frames in each frameset of that clip/video.

[4] I = Intra Frame, B = Bi-directional Frame, P = Inter Frame

For obvious reasons, stream startup time shall vary mostly due to the adaptation time variance at the intermediate nodes and the inconsistent communication delay between peers. For live video streaming, there will be an additional delay for initial stream capturing and processing. Now, if all the segment lengths for a video stream are fixed, and the difference in processing time is small, then an initial buffer size (IBS) for each times units (in seconds) can be computed as: $IBS = clipLength \times frameRate \times averageFrameSize$. However, user's buffering capabilities will vary due to the presence of heterogeneous devices in the mobile environment and their respective memory sizes. If we consider small handheld devices which have limited memory space, the clip length can be shortened, e.g. 1 second long SQCIF (frame resolution 128×96, size 3Kb) video at 5 frames per second (fps). Since the gBSD is transmitted first, the *IBS* can be derived directly from the gBSD because gBSD consists of the necessary information like number of frames, frame size etc.

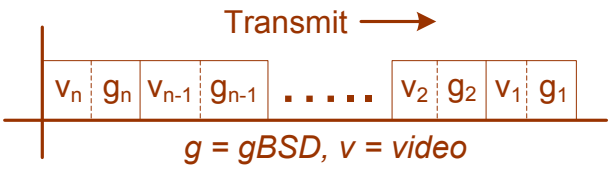


Figure 4. gBSD transmission with video

What we have seen so far is that the source split video into parts. Then these parts are transmitted into succession. It enables the receiver to decode and playback the content, while the video is still being delivered and buffered at the receiver. Now, if a receiver node requires adapted video then the sender/parent node adapts the video parts (i.e. clips/framesets) based on gBSD before transmitting those parts. Therefore, in addition to streaming the video content, we also require the transmission of the metadata i.e. gBSD. Transmission of gBSD basically consists of the following steps (Figure 4): 1. Generate the gBSD of the compressed video, 2. Tag the gBSD with a sequence number analogous to the video segment, and 3. Transmit the gBSD along with the video segment.

Now, one question that may arise is that how does the controller learn about an incoming peer's computing power or bandwidth. To address this issue, MPEG-21 framework also provides Usage Environment Description (UED) tools which include the description of terminal capabilities (e.g. codec capabilities), network descriptions (e.g. maximum or minimum bandwidth), user characteristics (e.g. personalized requirements) etc. In our case, such descriptions are used to identify user requirements and device capability.

**Mapping CPU power to Adaptation time:** To compute CPU power of a peer, controller needs specific information like processor speed, L2 cache, memory, clock rate etc. Nevertheless, absence of a benchmark to classify these computation factors make it complicated and unreliable if we take into

consideration the diverse hardware/systems available these days. To overcome this issue, we map the CPU power into adaptation time and create an adaptation timing profile for each peer. Towards this, a peer will adapt a small video clip for different temporal adaptations (30fps to 1fps) while installing the program. Based on this adaptation time, controller will be able to know how many adaptations this peer can perform in a certain time frame; hence controller will assign adaptation tasks to different nodes.

In the previous section, we have mentioned that framesets are used for our adaptation approach and each frameset refers to one-second video consisting of 30frames. So, for real time adaptation, considering the initial startup delay, a capable peer has to adapt each frameset in less than one second – hence, the total number of adaptation tasks it can perform in one second will symbolize how many peers it can serve in terms of adaptation (provided it also has enough bandwidth). For example, if a peer requires 100 millisecond to adapt a frameset from 30fps to 20fps then in one second it can perform around 10 similar adaptation operations.

## 4.2. Overview of the tree construction

A crucial decision in tree construction is choosing the number of parents to serve a peer. Complexity of the tree construction depends on this number because we consider both adaptation and streaming. To serve a peer, finding more than one parent having enough CPU power and bandwidth is sometimes inconvenient especially when the heterogeneity of peers is taken into consideration. In this paper, we present scheme which supports maximum 2 simultaneous parents to support a peer requiring adaptation. Multiple parents (more than 2 parents) option is under consideration and is left as a future work.

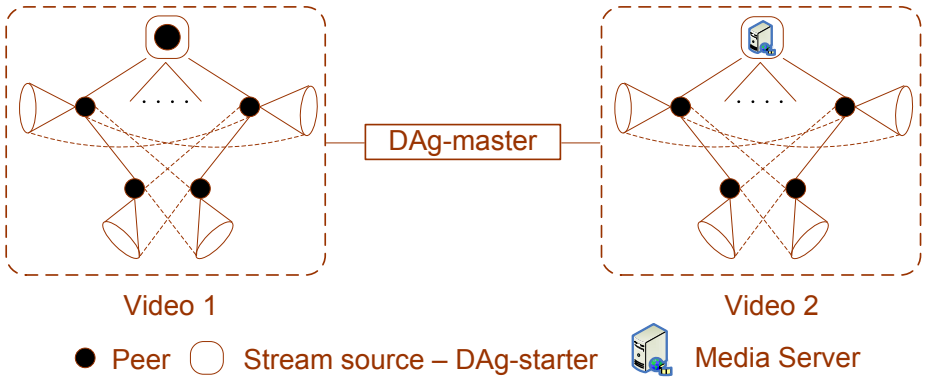


Figure 5. DAg-stream overlay design: A media server or an ordinary peer can be the stream source - Solid lines and Dotted lines refers to Primary sender and Secondary sender, respectively

Figure 5 shows the conceptual design of the system. In this design, either a media server or a participating peer may initiate the streaming and forms an overlay with the help of the DAg-master. This means that

for each video a separate unidirectional single source overlay is formed, and all these overlays are managed by the DAg-master. Each video stream originates from the stream starter which is the root node for that particular overlay. It is possible that stream starter may not have enough CPU power to do adaptation but the starter should be able to provide streaming service to some immediate receiver peers in the tree. Moreover, to perform adaptations in intermediary nodes, stream starter must be able to produce and forward the corresponding metadata component in the form of MPEG-21 gBSD for the streamed video content. Please note that the tree controller is not responsible for providing any video stream itself. Small hand-held devices are out of scope of any adaptation operation and tree maintenance. They can join the overlay just to receive the (adapted) stream. In addition, it is optional to transmit gBSD to these devices since they are not able to adapt video.

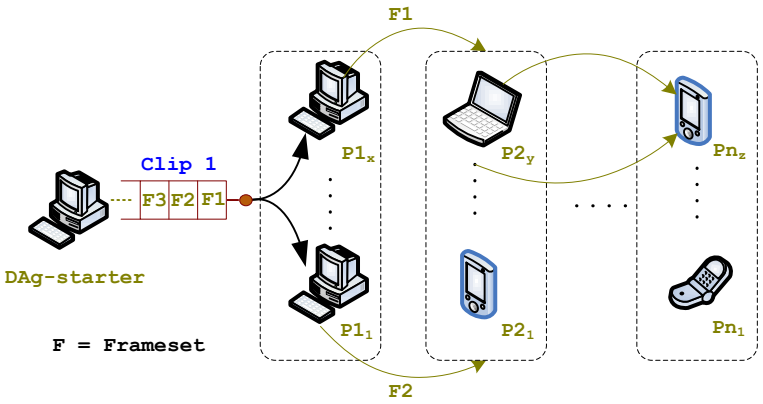


Figure 6. Video streaming in the multicast tree

In our design, the nodes at the top of the overlay are probably the wealthier nodes, but as we can see from Figure 6 that for obvious reason first layer peers ($P1_1$, …, $P1_x$) are served by single parent (i.e. DAg-starter). However, starting from the second layer, the controller may attempt to assign two parents for each incoming peer. Fact is that serving a peer with more than one peer is conceptually easy but practically the system becomes complicated in terms of implementation and tree maintenance. From Figure 6, we can also notice that video starter sends live video to its immediate peer as small clips consisting of framesets. Afterwards, first layer peers send odd or even framesets to its immediate peers as instructed by the controller and so on. On the receiver's side, these odd and even framesets are coordinated in the buffer. Nevertheless, if streaming and adaptation is done by a single peer, then both odd and even framesets are sent by that peer. Now, the benefit of dual parent approach and splitting video streams into framesets is that a receiver peer will not starve unless both of its parents have been disconnected. We shall learn additional details on these issues shortly.

To serve a node by one parent, the selection process is quite straight forward where the master needs to select a suitable parent from the existing set of capable peers. However, to serve a node by two parents,

there is a necessity of coordination among senders. This is one of the key challenges especially when adaptation of the bitstreams is a priority. We thus propose to send the clips as a coordinated odd-even framesets to the receiver peer from the sender peers as instructed by the master.

***DAg-master - The tree controller*:** DAg-master acts as a centralized index of the available streams and stream sources. It authorizes, classifies, and clusters peers according to their bandwidth and CPU power when they attempt to join an overlay. DAg-master monitors the status of peers and also maintains dynamics and diversity of the overlays as reflected in the following cases: 1. a sender may stop contributing by hanging, 2. out bound bandwidth and computing power may change over time. The tree controller keeps track of all the connected peers to the overlay(s). A set of nodes, $Set_p$, served by another peer, P, is determined by the controller based on the selection procedure presented in Section 4.2.2. P receives the updated list and required information from the controller. If the stream starter leaves or fails, the tree is dissolved immediately by the controller because stream starter was the provider of the video content for that tree.

From the UED description, the tree controller computes the adaptation requirements. For example, if there is no preference from the recipient, then depending on the download bandwidth, the temporal adaptation requirement can be computed in terms of target frame rate. Otherwise, the controller attempts to serve the node according to its preference. However, a node's video requirement cannot be higher than the original streamed video.

One important thing is that presence of backup parents is not needed in our case. DAg-master usually keeps the information sorted. So, the tree controller can (re) assign a parent when the current parent leaves the system. However, to serve a disconnected peer, the DAg-master maintains a node priority. For example, existing peers with higher resources have higher precedence over new incoming peer and free riders, respectively.

### *4.2.1. Node registration*

Every incoming node passes basic information entailing device requirements, network characteristics, adaptation capability and personal preferences in the form of UED descriptions to the controller. From implementation view point it can be seen like this - every peer wishing to join this system will have a program running in its end. The program will register the peer to a multicast streaming session. The idea is that the program will know the address of the DAg-master beforehand like that of Yahoo or MSN messenger. However, if the system is deployed within a neighborhood then the address of the DAg-master can be set manually. To enable an automated peer discovery, we suggest applying the Peer Discovery Protocol (PDP) within JXTA services [21]. Once connected, the peer will receive a list of

available streams. Then the user will choose a stream to view, based on which, DAg-master will assign parent(s) to it.

### *4.2.2. Parent selection*

Based on the incoming peer's requirements, DAg-master selects two candidate parents that can directly serve this peer possibly with smallest height in the (sub) tree. If the DAg-master is unable to find two parents then it attempts to assign one parent. DAg-master executes the following steps to select the candidate parent(s)–

1. A set of potential parents, $Set_1$, is selected as stream source that has the requested video ($vid_r$) along with the gBSD, i.e., $Set_1 = \{p \mid p \in Overlay, available\_video(p) = vid_r\}$.
2. Find a subset of $Set_1$ based on available bandwidth, i.e., $Set_2 = \{p \mid p \in Set_1 \text{ and } available\_bandwidth(p) = true\}$.
3. *Check adaptation requirement:* For each p, $p \in Set_2$, if there are peer(s) having the same video quality, then no adaptation is required.
4. If adaptation is required, select a subset of $Set_2$ based on available computing power i.e., $Set_3 = \{p \mid p \in Set_2, available\_CPU(p) = true\}$.
5. Finally, the cumulative distance of each peer, $p \in Set_3$, to the controller (in terms of delay) is measured to select the parent(s) that has minimum distance to the controller.

Remarks:

- The required bandwidth varies for different videos. When a peer wants to share a video, it sends the stream related information to the controller.
- Availability of bandwidth has higher priority than available computing power because if there is not enough bandwidth then adapted video streams cannot be streamed to another peer.
- It may seem that when a new peer joins, the algorithm has to be run again. Hence, $Set_1$ is temporarily saved by the controller to avoid re-computation every time a new peer joins the overlay. Controller updates this list over time when a peer leaves the overlay. However, the list is recomputed after tree refinement operations (Section 4.2.5.).
- In dual parent approach, for obvious reasons both the parents may not have similar computational and network properties. Therefore, adaptation and streaming requests served by two parents will be based on the minimum of one of the parents can support.
- Due to mobility, if the upload bandwidth of a parent node decreases and affects the bandwidth to serve its immediate peers, then it will attempt to serve peers according to their peer class and joining order.

### *4.2.3. Node joining constraint*

Even though we plan to deal with heterogeneity but it is also true that a peer having no sending throughput and computing power does not let the overlay scale beyond that point. In such a case, the overlay will be saturated quickly when too many receiver-only peers join the overlay at the beginning. This fact cannot be ignored and is taken into account by placing the following constraint:

**Constraint 01**: If the incoming node is a 'receiver-only' peer then the selected parents must be able to serve at least one more regular peer.

### *4.2.4. Node Departure*

In a live streaming session, it is very likely that a node may depart gracefully by informing the controller or ungracefully by hanging. For the latter case, in our design, if there are multiple reconnection requests from the peers of a (sub) tree, then controller verifies the existence of the parent of that (sub) tree. Otherwise, if there is no reconnection request, a parent reports the departure of such node to the controller.

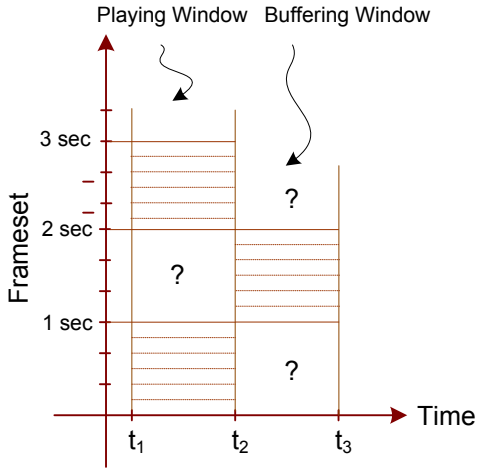


Figure 7. Frameset skip due to sudden departure of a parent

Please note, in case of a parent disconnection (i.e. 1 out of 2 parents), we prefer to keep its children connected to the overlay served by single parent only. The reason behind this design decision is to keep the peer disconnection rate low and to give service priority to the existing node(s). Giving service priority to existing node means that as soon as a capable peer is discovered to serve the single-parent child, the controller will assign the new peer to it as a second parent. Till then, the controller instructs the existing parent to serve the affected node with extra resource, provided that this parent has extra bandwidth and computing power. If the existing parent can not extend its service towards that affected peer then until a second parent can be assigned, the affected peer will experience temporal jitter (due to frameset skip as

shown in Figure 7) in the video. Controller applies the following steps to find a replacement parent for the affected node(s):

1. If the departing node is a leaf node without any responsibility, controller updates the $Set_1$ and informs the parent.

2. If a peer gets disconnected and requests a new link, controller checks the existence of its parent.

- If the parent is active, then controller instructs them to reconnect.
- If the parent has departed, and it was the stream source (i.e. DAg-starter), then the tree is dissolved immediately.
- If the parent was the root of a (sub) tree, then controller chooses a capable peer from the same level or the ancestors of the disconnected peer and instructs to serve the affected peers.
- If the above steps fail, then controller starts distributing the children of disconnected parent in the overlay.

### *4.2.5. Tree refinement*

It is understandable that due to the dynamics of both node joining and node departure, the overlay structure can easily become sub-optimal. Now, if assignments of peers do not yield the full quality vis-à-vis delay requirement of a particular session then a peer switching (i.e. tree refinement) is attempted. Additionally, overlay reconfiguration facilitates expanding the service to new nodes. Currently, we use a simple refinement technique by shifting the wealthier nodes in the higher layer of a tree.

DAg-master decides switching of active senders and new incoming peers so that the overlay performance remains satisfactory. It checks the aptitude of the available nodes, i.e. available bandwidth, CPU power, delay from the controller, and share-ratio of a peer. Afterwards it refines the tree by placing the nodes that are more capable in the upper layer of the (sub) trees. For example, consider the case in which ‘p’ and ‘q’ are two existing peers and ‘c’ is the stream starter, where, ‘p’ is connected to ‘c’, and ‘q’ is connected to ‘p’. Suppose that the delay of a new incoming peer ‘s’ will exceed its threshold if it is served by the first available serving peer ‘p’. Therefore, the tree controller will replace ‘p’ with ‘s’, satisfying the following conditions (otherwise ‘p’ will serve ‘s’):

*1. out-degree(c) > 1*

*2. bandwidth(s) ≥ bandwidth(p)*

*3. CPU-power(s) ≥ CPU-power(p)*

*4. delay(c,s) < delay(c,p)*

*5. share-ratio(s) > share-ratio(p)*

Condition ‘1’ above ensures that ‘c’ can serve ‘s’ without starving ‘p’. Once the node ‘c’ and ‘s’ is connected, tree controller instructs ‘p’ to connect to ‘s’.

### 4.3. DAg-stream as a client-server application

From application viewpoint, DAg-stream can be seen as a client-server application. Its components are mainly divided into receiver side functions, sender side functions and adaptation engine. However, for small handhelds, the application can have only the receiver side functions. Now, while implementing, there are three software components: 1. *Tree Controller* (DAg-master), 2. *DAg-client* (for regular peers), 3. *DAg-client-light* (for mobile peers). *Tree Controller* application functionalities are quite straight forward. It creates and manages separate overlays for each video stream. *DAg-client* is capable of initiating a stream, adaptation and regular streaming. Additionally, it is possible to append tree controller functionalities in the *DAg-client*. *DAg-client-light* is capable of receiving streamed data only. Figure 8 gives a snapshot of our design from a client-server application perspective. In Table 1, we further mention different components, their description and presence in DAg-stream application.

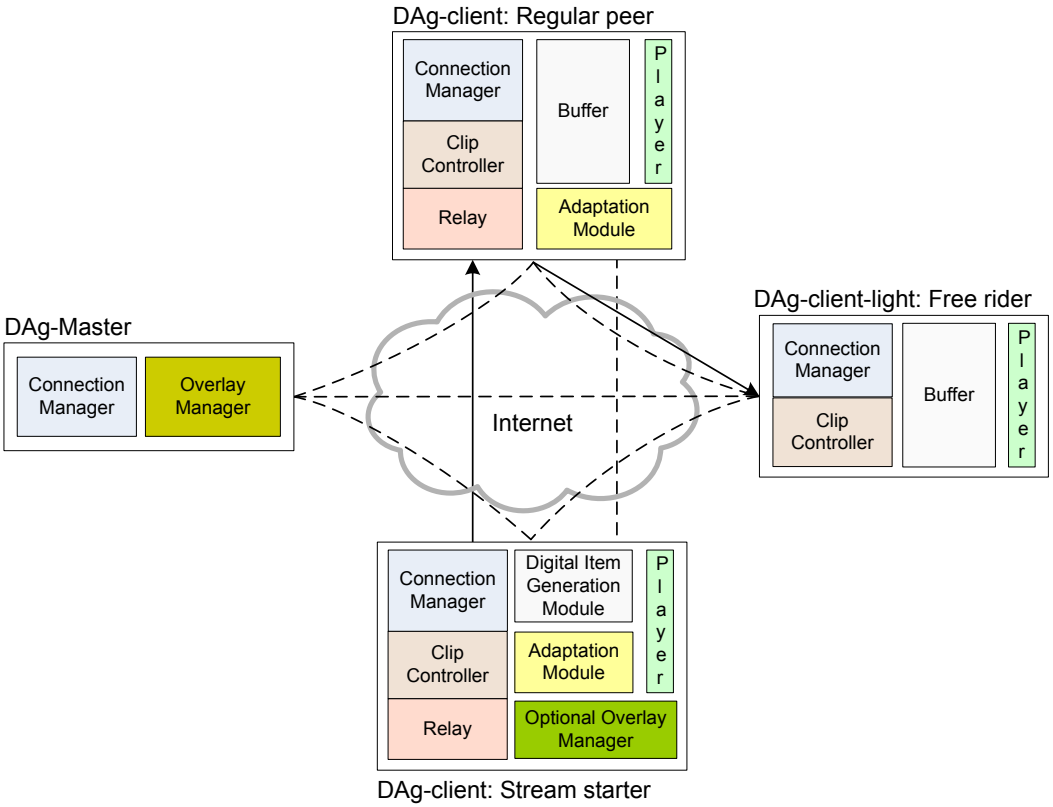


Figure 8. DAg-stream as an application - Solid lines and Dotted lines, refers to Video data and Control data, respectively

Table 1. Different System Components of DAg-stream application

| Component Name | Presence | Description |
|---|---|---|
| Connection Manager | DAg-master, Regular Peer, Source Peer, Free-Rider | Responsible for handling individual TCP/IP and UDP/IP connections among nodes |
| Overlay Manager | DAg-master, Stream starters | Keeps track of overlay related data, e.g. peers, stream source. Also assigns parents, handles node joining, peer failure etc. |
| Clip Controller | All peers | It coordinates clips and corresponding gBSD. For senders, it also tags the clips and gBSDs. It also maintains the share-ratio. |
| Relay | Stream starter, Regular Peer | It forwards the received video segments (after adaptation if required) to other peers connected to this peer. |
| Digital Item Generation Module | Stream starter | Generates H.264 encoded video and corresponding gBSD |
| Adaptation Module | Stream starter, Regular Peer | Responsible for adapting video segments according to receiver's requirement |

**Detecting peer failure:** A peer failure is detected by monitoring the TCP control channel established between the receiver and each of the sending peers. If a connection reset is detected, then either the receiver peer(s) or the parent of the departed node informs the master.

**Handling Network Fluctuations:** Typically, adaptation requirements and/or bandwidth requirements are computed and set at the beginning of the streaming sessions. For the single parent case, the streaming rate assigned by the master is always lower than the available bandwidth. However, network fluctuations can happen due to network congestion in the network. In such a case, DAg-master will attempt to enable dual parent approach. If not, then the adaptation and/or bandwidth requirements will be recomputed.

# 5. Evaluation

In this section we present the performance of our proposed scheme. The experimental results include both the aspects of our scheme covering temporal/frame rate adaptation applying MPEG-21 and building the multicast tree. For ALM, our experimental results are collected over a simulated network.

## 5.1. Performance metrics

If we consider the delay in the overlay, then the network delay between peers is negligible with respect to the adaptation time. Therefore, we provide an adaptation time profiling showing the feasibility of compressed domain adaptation online. Since, we are interested in live video streaming, so the number of hops in the ALM tree is important to measure the quality of experience. Higher number of hops will introduce larger delay from source node to the end node. In order to assess the overhead of overlay streaming, we use the metrics peer stretch, where *Peer stretch = (Delay along the ALM path) / (Unicast delay)*. The major focus of this design was to increase the service quantity. Therefore, node joining success rate represents the effectiveness of the overall design. Here, node joining includes both new incoming peer request and disconnected peer rejoin request. Bandwidth and Adaptation slot utilization gives an idea of the portion of the total overlay resources utilized. For each peer, adaptation slot refers to the average number of adaptation operations a peer can perform in 1-second based on the adaptation timing profile of the peer. Moreover, Quality of experience in terms of viewing a video stream decreases due to frequent node joining and node departure especially when these are service nodes. To evaluate the quality of experience by a viewer, we have introduced a matrix named Service Disruption Index (SDI) which is the ratio of the total streaming disruption time to the total viewing time since the playback begins.

## 5.2. Experimental setup

To demonstrate the adaptation performance, we have used two test sequences – 'Bridge-far' and 'Highway'. Each of the video sequences consists of 300 frames (QCIF: 176×144). To generate the gBSD, we have enhanced the ITU-T H.264 video reference software implementation [20] with gBSD generation functionality. The gBSD for each video is generated during the encoding process of the raw bitstream. Since gBSD is provided by the DAg-starter, so it has to have the capability to generate gBSD for each streaming video.

To investigate the overlay generation performance we have designed our own simulator. For this simulation, we consider one video-starter and around 150 nodes. The video starter is capable of both streaming and adapting to its immediate peers. For all 151 nodes, we set the following properties – a node ID, peer-class, upload bandwidth, download bandwidth, and alive-time. For the video starter we also set the corresponding video properties, like – video resolution, frame rate, and average frame size. The nodes are initially kept in a pool of *not-connected-nodes*. Once the simulation starts, the simulator uses a random number generator to pick a node from the pool in every 2-time units. For each node upon successful connection, alive-time varies from 5-time units to 1000-time units. The end-to-end delay

threshold is set to 300ms. We have tested our design with two approaches – 1. Single parent approach - incoming nodes are served by only one parent, 2. Dual parent approach - incoming nodes are preferably served by two parents otherwise by single parent. In both approaches, the participating peers are heterogeneous (in terms of available bandwidth) and demonstrate random behavior (in terms of arrival, departure and streaming/adaptation requirements). Now, in some experiments we have introduced batching. Batching incurs service latency but increases the possibility of larger group formation. This is because when the batch duration is introduced, it is very likely that more similar requests will be accumulated within this short interval of time. However, it is also true that it increases initial service latency that may cause some peers to stop. We set the batching duration to 10 time-units.

To make the simulation realistic, we have considered the asymmetric nature of nodes which means that nodes behind DSL and Cable can receive several hundreds of kilobits per second but can fundamentally only donate less. The upload and download bandwidth for a peer varies from 1Kbps to 1Mbps and 8Kbps to 7Mbps respectively, which is based on the experimental conditions used in [22]. Moreover, the upload and download bandwidth changes during the alive time of a peer in the overlay. The temporal video quality is computed automatically based on the download bandwidth of an incoming peer. We also assume that on average it requires 6Kbps of bandwidth per video frame (about 180 Kbps for a video at 30 fps) regardless of the video motion and frame type. While joining, two connections are established with each incoming peer – a UDP connection for sending the video stream and a TCP connection for sending the control data. In an area of $10^8$ sq. kilometers, all peers are placed randomly. Peer-to-Peer delay is measured computing the Euclidean distance between peers and assigning a delay per unit distance. For ease of implementation, we assume throughout this simulation that delay between any two peers is symmetric.

## 5.3. Experimental Results

Table 2 presents the average time to adapt one frameset for the test video sequences. Since not all peers are capable of adaptation, we have classified peers into 3-tiers which can do adaptation – others are assumed as not capable of performing any adaptation and considered as Tier-4. We have found that, on average, for each adaptation operation CPU usage varies from 50% - 63%. From the table data, we can claim that the adaptation time is quite acceptable considering the fact that the processing is being done in compressed domain and in a live fashion. Here, adaptation time includes adaptation of both the gBSD and the video.

Table 2. gBSD-based temporal adaptation performance (1 frameset; 30 frames; Frame size: QCIF)

| Peer Class | Target Frame Rate | Time (Millisecond) |
|---|---|---|
| Tier-1<br>Specification: Pentium IV, 3.0 Ghz, 1GB RAM or equivalent | 1 – 5fps | 46 |
| | 6 – 10 fps | 62 |
| | 11 – 15 fps | 78 |
| | 16 – 20fps | 93 |
| | 21 – 25 fps | 109 |
| | 26 – 29 fps | 125 |
| Tier-2<br>Specification: AMD Athlon XP, 1.4 Ghz, 512MB RAM or equivalent | 1 – 5fps | 100 |
| | 6 – 10 fps | 125 |
| | 11 – 15 fps | 150 |
| | 16 – 20fps | 210 |
| | 21 – 25 fps | 230 |
| | 26 – 29 fps | 260 |
| Tier-3<br>Specification: Pentium III, 700 Mhz, 256 MB RAM or equivalent | 1 – 5fps | 190 |
| | 6 – 10 fps | 230 |
| | 11 – 15 fps | 270 |
| | 16 – 20fps | 310 |
| | 21 – 25 fps | 340 |
| | 26 – 29 fps | 380 |

From the simulation results, we have found that overall performance of the design is quite dependent on the participating nodes. In Table 3, we show that the density of Tier-1 and Tier-2 parents is an important parameter which drives the success of a particular overlay.

Table 3. Impact of the wealthier parents' availability on the successful overlay generation

| Tier-1 (%) | Tier-2 (%) | Tier-3 (%) | Tier-4 (%) | Success Rate (%) |
|---|---|---|---|---|
| 10 | 0 | 0 | 90 | 16-20 |
| 20 | 0 | 0 | 80 | 22-30 |
| 30 | 0 | 0 | 70 | 40-46 |
| 40 | 0 | 0 | 60 | 57-60 |
| 50 | 0 | 0 | 50 | 81-85 |
| 10 | 40 | 0 | 50 | 60-68 |
| 10 | 30 | 10 | 50 | 55-60 |
| 10 | 20 | 30 | 50 | 50-55 |

In Figure 9, 10 and 11 we have compared the results with two approaches, i.e. Single parent approach and Dual parent approach. Figure 9 illustrates the new node joining success rate. In both cases, batching (first half – every 5 time units, second half – every 15 time units) is incorporated to see the performance variation. From Figure 9, we see that the node joining success rate is consistent during different runs considering the peer heterogeneity. Dual-parent approach with batching offers higher success rate (even though it causes additional startup latency). However, if batching is enabled, controller can select and assign tasks to wealthier nodes from a pool of incoming peers.

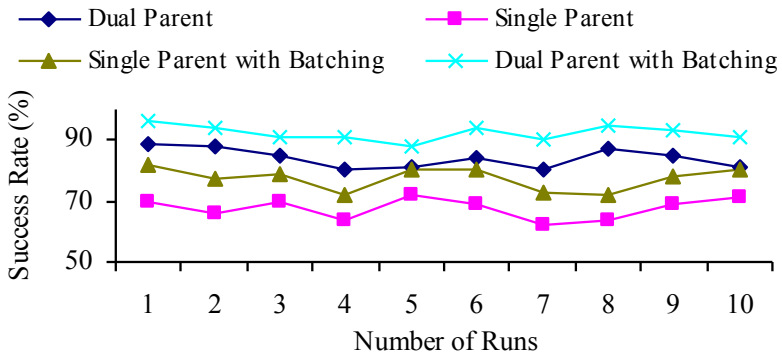


Figure 9. New node joining success rate

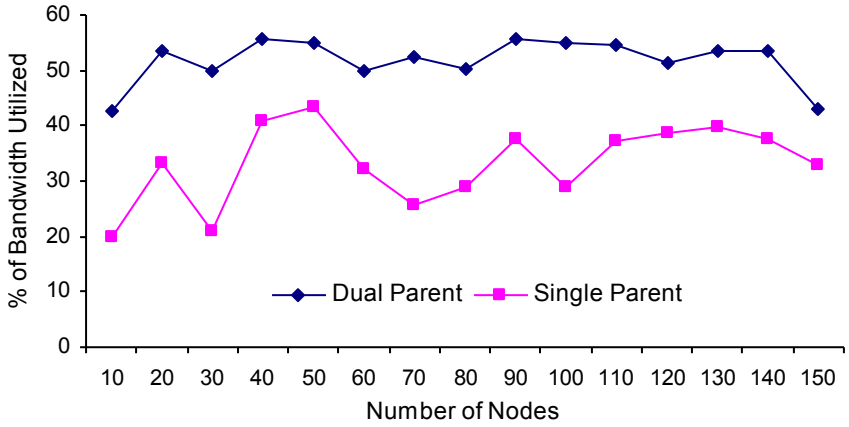


Figure 10. Percentage of total bandwidth usage

In Figure 10 and Figure 11, we show the percentage of total system bandwidth and adaptation time slots utilized to serve peers, respectively. Total bandwidth utilization (Figure 10) is higher in dual parent approach which is quite pertinent to the previous result that the success rate is also higher in dual parent option. In Figure 11, we can see that the total adaptation slots utilized are low. This is because of the fact that there can be an ordinary peer requesting low quality video which can then serve other peers requesting the same video quality without any further adaptation. Moreover, if a parent is serving a child with some specific frame rate then it can serve other incoming peers requesting the same quality video, provided that the parent has enough upload bandwidth. This fact is illustrated in Figure 12, which shows

that the number of peers requesting adapted video and served by the system is small over the simulation time in the dual parent approach.

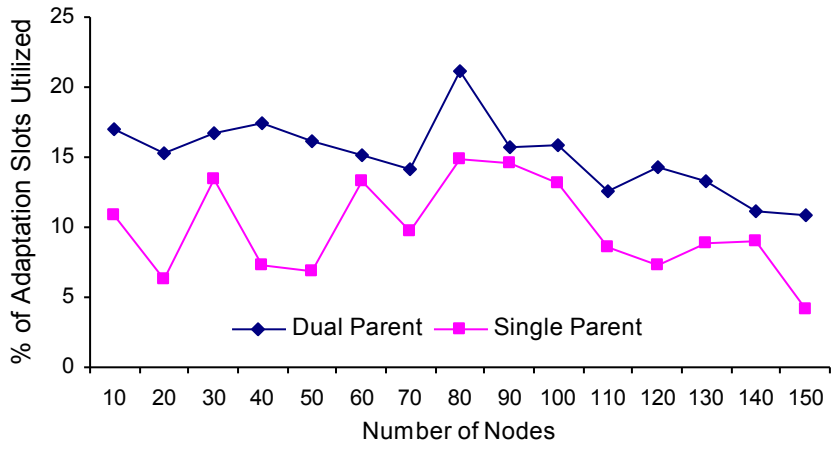


Figure 11. Percentage of total adaptation slots usage

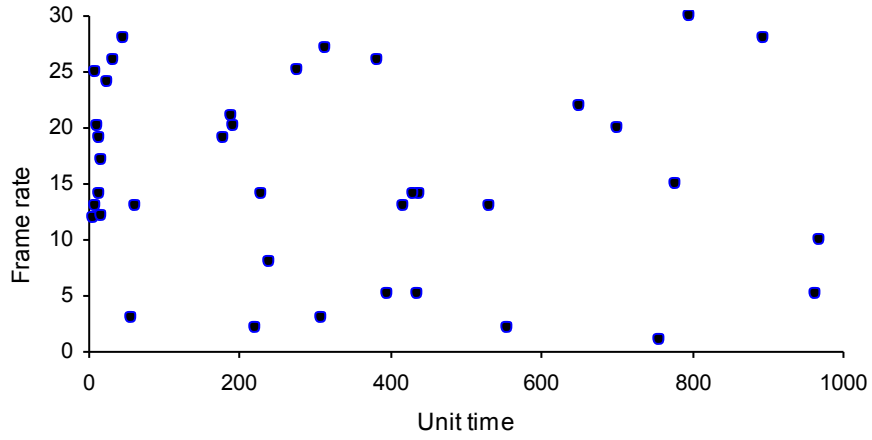


Figure 12. Peers requesting adapted video in Dual-parent approach

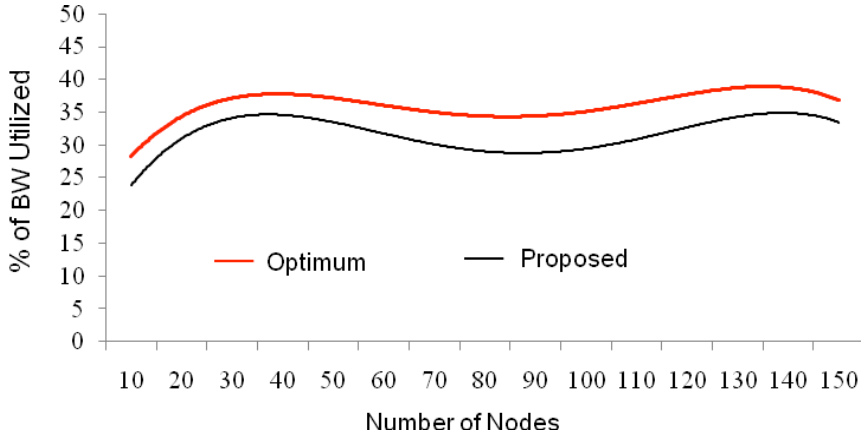


Figure 13. Comparison of bandwidth utilization single parent approach - Optimum vs. Proposed

To validate the evaluation of resource utilization presented above, in Figure 13 and Figure 14 we also present a comparison of our single parent approach with a theoretical optimal approach based on an analytical computation using Integer Linear Programming, the details of which can be found in [18]. From Figure 13 and Figure 14, we can see that the performance of the single parent approach closely follows the optimal boundary in terms of bandwidth utilization and adaptation slot utilization, respectively.

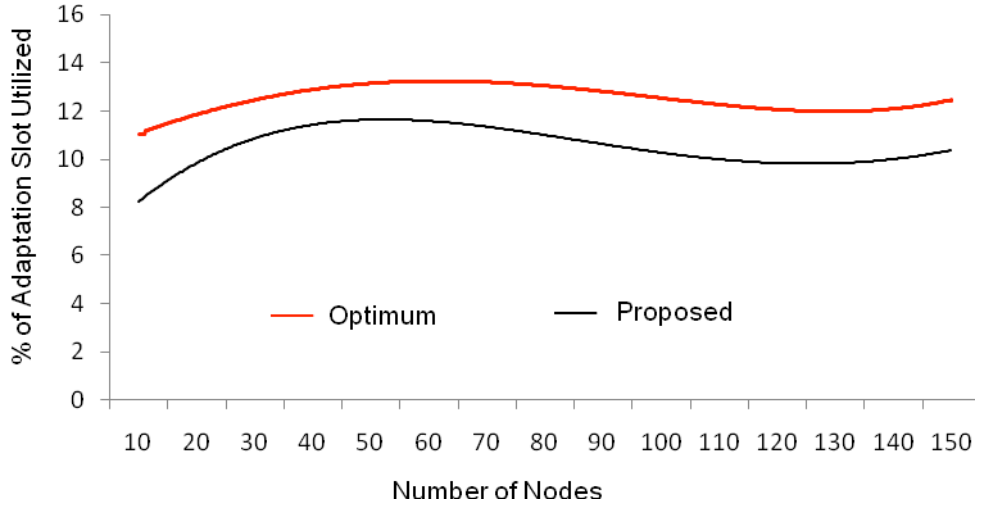


Figure 14. Comparison of adaptation slot utilization for single parent approach - Optimum vs. Proposed

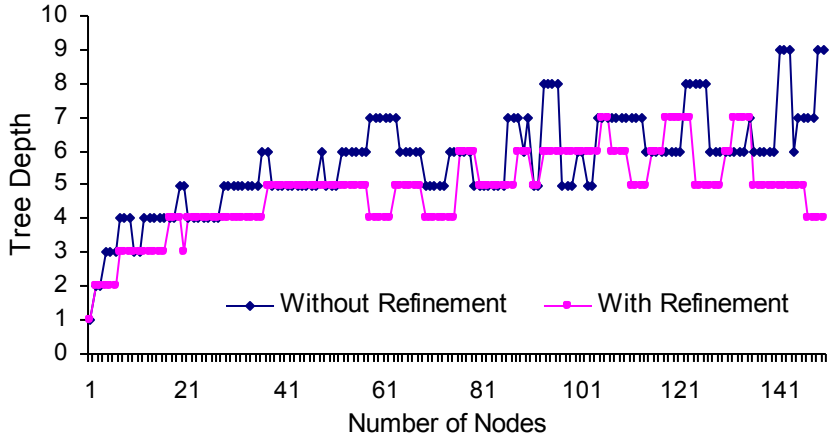


Figure 15. Tree refinement performance

The tree refinement operation used by DAg-master also enhances system performance by minimizing tree height through the shifting of resourceful peers at the top of the tree. Figure 15 compares tree depth before and after refinement operation against the number of peers. In Figure 15, we can see that before

refinement, highest tree depth is 9, whereas, after refinement highest tree depth is 7. We have observed that cumulative delay of some peers is more than the threshold (i.e. 300ms) because of the fact that DAg-master could not satisfy the refinement conditions. Therefore, no replacement operations took place. From the simulation results, we computed that the average peer stretch, which is 4.5 and 6.5 for single parent and dual parent approach, respectively. The peer stretch of the above system is actually quite promising, especially when we need to consider the joining constraint, upload bandwidth and CPU requirements for adaptation operations.

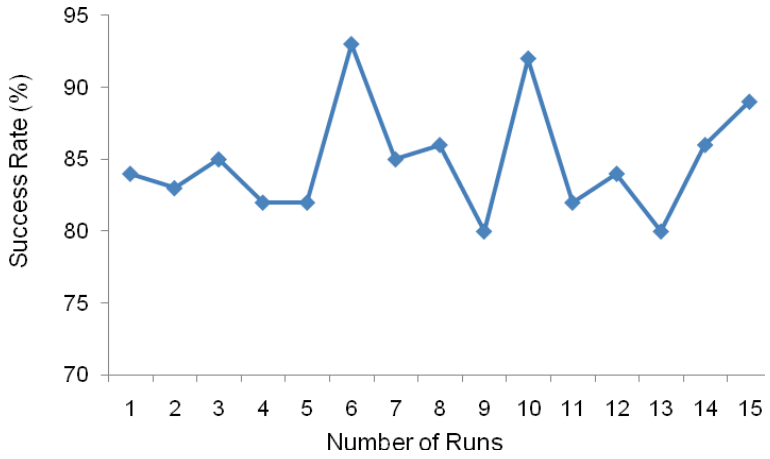


Figure 16. Effect of Churn: Internal node joining success rate

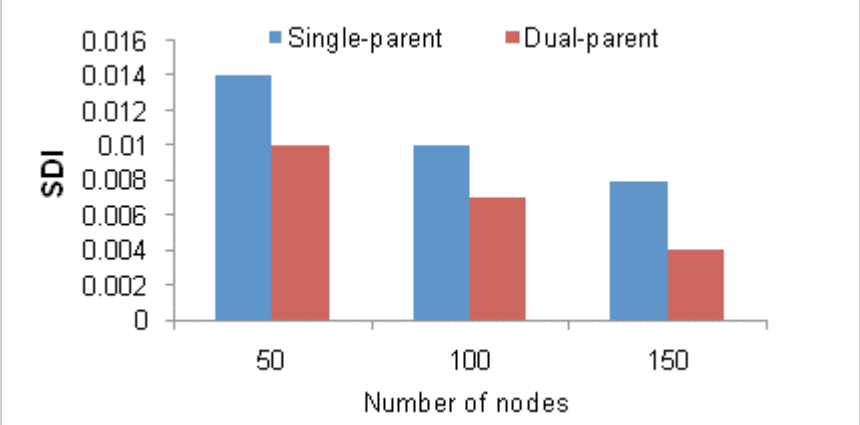


Figure 17. Effect of Churn: SDI for different number of parents

Regular peer disconnection (graceful or ungraceful), peer mobility, and presence of handheld devices affect the system performance due to the high frequency of node joining and leaving the network (i.e. Churn rate). Obviously, churn rate of handheld devices are much higher than that of fixed nodes in the wired networks. Since Tier-1, Tier-2 and Tier-3 peers are responsible for adaptation and streaming

services, and Tier-4 peers are mostly handheld devices who join the overlays as Free-rides, nodes joining and leaving the network from Tier-4 do not really hamper the topological properties. Figure 16 shows the effect of churn due to which peers get disconnected and subsequently request reconnection. We put higher priority on servicing disconnected peers than new connection requests. The effect of churn is measured for the steady-state peers (both static and mobile) who were selected as service nodes. Figure 17 shows the experimental results for the SDI. In this experiment, we consider the extreme case in which both the parent nodes may depart ungracefully. Obviously, dual-parent option is better than the single-parent approach because a receiver peer does not starve entirely unless both of its parents have left the overlay.

# 6. Discussion

The initial idea behind P2P streaming was that viewers will contribute their bandwidth to the overlay and act as a relay for the video streams, which means that users will upload video contents to other users while viewing it. Now, we can call this as the first generation P2P video streaming systems. Such P2P video communities are now formed in practice, as evident by various application groups such as PPMate (*http://www.ppmate.com*), SopCast (*http://www.sopcast.org*), TVUPlayer (*http://www.tvunetworks.com*), and TvAnts (*http://tvants.en.softonic.com*). While still not of the highest quality, these first generation platforms allow for reasonable live P2P video broadcasting of sporting events and TV channels in a community with a large number of global users.

"*What is missing*" however is to include in this community the users that do not use a desktop PC; i.e., the community needs to move towards supporting mobile and heterogeneous devices. In this paper, we attempt to add up heterogeneous receivers to overlay streaming; the architecture will provide every intended recipient with the required quality that they could sustain in terms of bitrate. This is an important issue, since different recipients may experience different bandwidth and device constraints. Therefore, adding the functionality of adapting video contents on the fly can be denoted as second-generation P2P systems.

Considering the peer heterogeneity, adding online adaptation capability in the streaming system seems to be a practical and feasible solution. In our design, the tree controller does not need to be a high-end computing server. Its presence is viable since thereby we can avoid a peer to communicate with all the existing peers except the tree controller and also to maintain scalability of the overlay. Benefit of serving a peer by two parents is that a receiver peer will not starve unless both of its parents have been disconnected.

From [16] we can see that in general adaptation process requires high computation load. However, from our previous experience of developing an adaptation system, we have found that processing load for gBSD generation and transformation is negligible compared to video adaptation operations [1]. Moreover, compressed domain adaptation makes it feasible to perform necessary operations on the fly. Here, we have shown that if we can properly schedule the processing time, then we can maximize the computing resource utilization to adapt contents and to serve heterogeneous peers in an ALM overlay.

To make the architecture more robust we could propose a complicated scheme like replicating the contents of the tree controller in a secondary node or may be even avoiding such controllers. In the latter scenario, peer discovery, handling disjoint network etc. could increase the chance of an unsuccessful topology for a video streaming. Now the question is - who will provide the tree controller? May be this could be a commercial service provider. Their benefit is that they could earn revenues by showing advertisements in the application window as we see for free VOIP software.

In our design, we have assumed that the rendezvous point is the tree controller and it will not fail, which in practice might not hold true all the time. This issue can be further discussed and many solutions can be proposed. One solution is that the tree controller may send a copy of the graph to the stream starter, which might then act as the rendezvous point only for the peers connected to that overlay in that particular time. In terms of node joining, the controller may send a list of all the available neighboring peers to the new incoming node or it may send a list of incoming nodes to the sender peers to select immediate peers. But in both the cases, as we mentioned before, a peer may overwhelm all other nodes with message flooding in case of a node departure or new node joining. Therefore, from application development perspective it can be left as an option.

Surely, latency is increased in our design for processing the video on the fly. Some may suggest that multi-layer encoding can accommodate heterogeneous peers without additional overhead of bandwidth or CPU power, but it is also true that not many devices are capable of decoding such streams. Moreover, layer-encoding does not permit us to have a fine grained adaptation since the more the number of layers, the less the coding efficiency. Furthermore the encoding process may consume high CPU power at the DAg-starter. With our design, regular peers perform the adaptation operations, and adapted video is still playable by a standard player. In general, we have tried to keep our design practical and applicable to both regular and mobile peers.

# 7. Conclusion

Researches entailing P2P systems for small handheld devices are increasingly becoming common but the development of efficient systems to support them is still in its early stages, and therefore, the full potential

is yet to be seen. In this paper, we elaborate our design of adaptive video streaming for mobile/heterogeneous devices. We focus on creating video distribution overlays enabling peer-assisted online adaptation. Our design attempts to couple the basics of P2P streaming concept with the mechanisms to adapt live non-interactive videos. This is an initiative towards second generation P2P streaming concept that incorporates video adaptation in the streaming architecture utilizing MPEG-21 gBSD. Our approach increases adaptation efficiency significantly and incorporates less-capable devices which were previously ignored. Using the gBSD based adaptation approach, it is very convenient to handle late comers and early leavers since we process the video in small segments. From the experimental results, we see that along with the bandwidth, computing power of the participating peers is also utilized which does not exhaust a peer's own resource entirely. The key of our approach is its simplicity and its ability to accommodate different client requirements on the fly.

# References


1. R. Iqbal, S. Shirmohammadi, A. El Saddik, and J. Zhao, "Compressed Domain Video Processing for Adaptation, Encryption, and Authentication", IEEE MultiMedia, vol. 15, no. 2, pp. 38-50, Apr-Jun, 2008.
2. D. Liu, E. Setton, B. Shen, and S. Chen, "PAT: Peer-Assisted Transcoding for Overlay Streaming to Heterogeneous Devices", in Proc. of NOSSDAV, 2007.
3. ISO/IEC 21000-7:2004, Information Technology Multimedia Framework Part 7: DIA.
4. M. Hosseini, D.T. Ahmed, S. Shirmohammadi, and N.D. Georganas, "A Survey of Application-Layer Multicast Protocols", IEEE Comm. Surveys and Tutorials, Vol. 9, Issue. 3, 2007, pp. 58 – 74.
5. S. Banerjee, B. Bhattacharjee, and C. Kommareddy, \Scalable application layer multicast", in Proc. of SIGCOMM, 2002, pp. 205 - 217.
6. D.A. Tran, K.A. Hua, and T.T. Do, "A peer-to-peer architecture for media streaming", IEEE Journal on Selected Areas in Communications, 2004, pp. 121- 133.
7. V.N. Padmanabhan, H.J. Wang, P.A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking", in Proc. NOSSDAV, 2002.
8. Y. Zhu, B. Li, and J. Guo, "Multicast with network coding in application-layer overlay networks", IEEE J. Selected Areas Communication, no. 22, 2004, pp. 107-120.
9. M. Zhang, J. G. Luo, L. Zhao, and S.Q. Yang, "A peer-to-peer network for live media streaming using a push-pull approach", in Proc., ACM intl. Conf. MM, 2005, pp. 287-290.
10. E. Setton, P. Baccichet, and B. Girod, "Peer-to-Peer Live multicast: A Video Perspective", Proc. of IEEE, Vol. 96, No. 1, Jan. 2008.
11. X. Tan, and S. Datta, "Building multicast trees for multimedia streaming in heterogeneous P2P networks", in Proc. of Systems Communications, 2005, pp. 141 - 146.
12. A. Rodriguez, D. Kostic, and A. Vahdat, "Scalability in adaptive multi-metric overlays", in Proc. of Intl Conf. on Distributed Computing Systems, 2004, pp. 112 – 121.

13. R. Rejaie, and A. Ortega, “PALS: Peer-to-Peer Adaptive Layered Streaming”, in Proc. of NOSSDAV, 2003, pp. 153 - 161.
14. X. Xiaofeng et al. “A peer-to-peer video-on-demand system using multiple description coding and server diversity", in Proc. of ICIP, 2004, pp. 1759-1762.
15. G. Exarchakos, and N. Antonopoulos, “Resource Sharing Architecture for Cooperative Heterogeneous P2P Overlays”, in Journal of Networks and Systems Management, vol. 15, pp. 311-334, 2007.
16. B. Shen, W. Tan, and F. Huve, “Dynamic Video Transcoding in mobile environments", IEEE Multimedia, 2008.
17. R. Iqbal, D.T. Ahmed, and S. Shirmohammadi, “Distributed Video Adaptation and Streaming for Heterogeneous Devices”, Proc. IEEE Workshop on Mobile Peer-to-Peer Computing, in Proc. IEEE Conference on Pervasive Computing and Communications Workshops, 2008, pp. 492 – 497.
18. R. Iqbal, B. Hariri, and S. Shirmohammadi, “Modeling and Evaluation of Overlay Generation Problem for Peer-assisted Video Adaptation and Streaming”, Proc. ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2008, pp. 87 – 92.
19. http://www.w3.org/TR/xslt
20. http://ftp3.itu.ch/av-arch/jvt-site/reference software/
21. http://www.sun.com/software/jxta/
22. K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, “The feasibility of supporting large-scale live streaming applications with dynamic application endpoints”, in Proc. of SIGCOMM, 2004.