Challenges and Solutions in Peer-to-peer Live Video Streaming

Masoud Moshref, Hamid R. Rabiee, and Saeed Nari
Department of Computer Engineering, Sharif University of Technology
Tehran, Iran.
Email: moshref@ce.sharif.edu, rabiee@sharif.edu, nari@sharif.edu

***Abstract*— Peer-to-peer networks have attracted considerable attention from researchers both in academia and in industry as an infrastructure for distributed computing and multimedia broadcasting. In recent years, many protocols using P2P networks to implement a video multicasting application have been proposed. In this paper, we present a problem-based viewpoint to survey the challenges in designing P2P live video streaming applications through a comprehensive analysis of negative and positive points of their solutions. We categorized these points in an innovative hierarchical structure based on four categories: topology, send/receive data mechanisms, incentive, and group management.**

***Index Terms*— Peer-to-Peer Networks, Live Video, Multimedia Streaming.**

I. Introduction

As the Internet technology is expanding, the content providers are investigating better ways of distributing multimedia contents through the Internet. These investigations have led to apply some new methods including video coding, data compression and multicast video streaming. In the recent applied methods, the server attempts to send video data using multiple unicast connections, IP Multicast, Application Layer Multicast (ALM), or a hybrid method. Nowadays, there are very strong servers and large bandwidth in use; however, the number of clients and their expectations have been considerably increased correspondingly. Therefore, the scalability issue seems to be the major drawback in methods using unicast connections. Furthermore, unicast video distribution does not use network resources efficiently and is not reliable because of single point of failure deficiency. The IP Multicast technology is network efficient, but it is network dependent, needs routers to keep per group state, and is unable to provide security, flow control, and congestion control as higher levels do. Besides, it seems that the ISPs are reluctant to activate it in their networks (However, it may be implemented in application specific networks (IP Multicast islands)). ALM takes advantage of Uplink bandwidth of clients and their processing capacity to gain scalability and flexibility, and it is network independent. There are three approaches to implement ALM: Content Distribution Networks (CDN), Peer-to-Peer networks, and a combination of both. Peer-to-peer networks do not need a particular investment and almost everyone can leverage it to multicast their contents.

A peer-to-peer overlay network is a kind of distributed system, which relies on resources of client machines to self-organize an overlay network as opposed to using network infrastructure or centralized control. These networks cover a large usage spectrum from grid computing, distributed data storage, and game networks to media broadcast of TV channels (live, on-demand, or interactive), and distance learning. Although these networks have many advantages such as high availability, low cost, and scalability, there are some issues which make design of P2P streaming protocols challenging. Examples of such challenges are dynamic membership, heterogeneous bandwidth and processing capacity, and existence of selfish and malicious nodes.

Non-interactive video streaming can be divided in two categories: live and on-demand. In a live video streaming session, video contents are being distributed among the users in a real-time synchronous manner. In contrast, users may prefer watching different parts of a video asynchronously or even different video clips at a time. In such cases, video on demand applications would be alternative solutions. Although these applications share some

common challenges, there are certain distinct areas. This paper focuses on live streaming while we may give a few solutions for common challenges usable in on-demand streaming as well.

Many researchers have studied this field and proposed a variety of credible protocols in recent years. A few surveys are being published either in peer-to-peer streaming protocols or about peer-to-peer streaming challenges within each year. Eng Keong Lua et *et al.* in [1] surveyed P2P overlay networks and categorized them in structured and unstructured P2P networks. They compared different overlay protocols only based on their topology construction process. [2] compared three error-resilient scheduling approaches in live video streaming supported by simulation and experimental results. [3] surveyed topology related challenges and solutions in live and VoD P2P streaming while it did not consider scheduling, coding, and incentive problems. Hosseini *et al.* [4] tried to compare ALM with alternative solutions. They also categorized ALM protocols based on their properties such as . Liu *et al.* [5] compared P2P multicast solution with alternative ones, listed the requirements for a P2P video broadcasting application, and presented the pros and cons of tree-based and data-driven structures. They also reviewed the challenges in P2P streaming in seven categories. Gao in [6] considered three parts for a P2P streaming protocol: topology construction, routing and scheduling, and membership management. However, its authors did not dig further to categorize challenges and solutions in each section. [7] considered two main challenges in P2P networks including locating potential neighbors and maintaining topology structure.

In this paper, authors have tried to take a new viewpoint on this field and study it based on a problem-based approach. We have chosen this approach because we think that the peer-to-peer streaming field is mature enough that repetitive problem-solution patterns can be extracted from the proposed protocols. We gathered and categorized these patterns and their instances in protocols to pave the way for proposing a useful pattern language [8] in this field. A pattern language is an organized way to describe a set of efficient design solutions to repetitive problems in a professional field.

The rest of the paper is organized as follows: In the next section, our presentation approach has been introduced. Subsequently, in the four later sections, we presented the challenges and their solutions in four general categories. In Section VII, the paper is concluded and some open research problems for P2P streaming are reviewed.

II. Categorization approach

A video streaming protocol can be divided into four tasks: 1) topology construction process, 2) data delivery process, 3) incentive mechanism 4) group management process. Note that these tasks are not independent and may have common topics. For example, they can have common challenges from different viewpoints; a solution in one task may have negative or positive effects on another one.

Initially, we present our data model and the logic behind our categorization before focusing on each part. Our presentation approach is based on a knowledge tree which has four edge types: challenge, solution, instance, and categorization. A challenge edge shows a challenge in its parent node, which can be the root node, a solution, or a subcategory. A solution edge defines a solution for a challenge, and an instance edge presents protocols using this solution. We have two purposes by showing an instance of a solution in a protocol. First, we like to show the context of the problem where the solution is proposed. Context may include the assumptions about the underlying network and the clients' machine, protocol application, or other related solutions used.

III. Building the topology

This section discusses the challenges to build a topology in designing a peer-to-peer live video streaming protocol. We divided these challenges based on the resulting topology in three categories: Tree, Mesh, and hybrid. Although we will mention later that the first two types can be very close in some properties and make a spectrum in the solution space, this categorization helps us review the literature better and focus on the proposed solutions.

A. Tree

In the tree topology, nodes make a tree-like structure rooted at the video source. Each node joining the tree receives video data from its parent positioned in the upper layer and sends them to its children located in the lower layer.

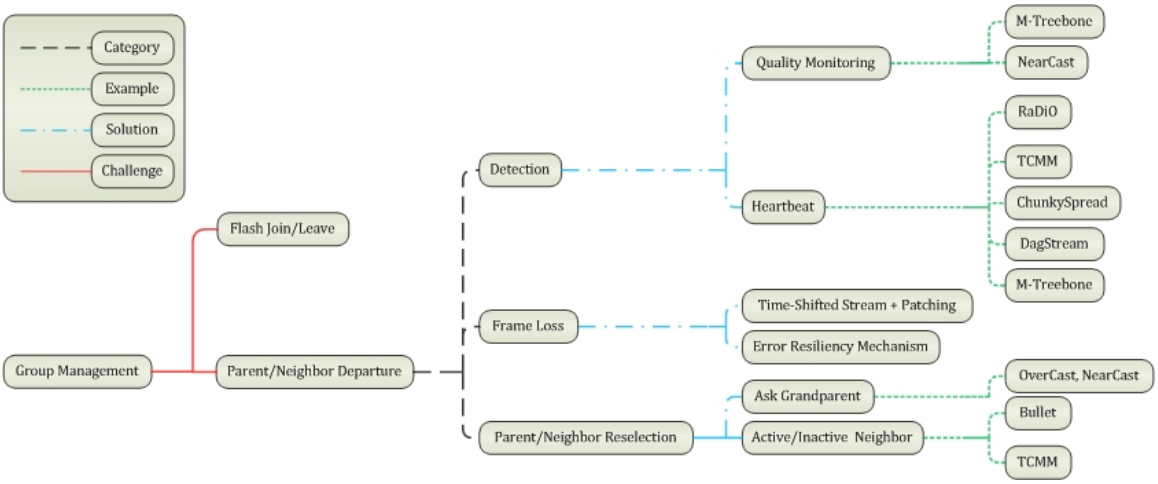

Fig. 1. A sample of the knowledge tree presenting the group management task challenges, solutions, and instances of the solutions

The first challenge, in tree structure, is obtaining the network address of potential parents. There are two well-known solutions: first, asking the source or bootstrap oracle node(s) [9] (The deployer may use load balancing for the server). Having a comprehensive global view and being able to run centralized topology management algorithms are advantages of this method. On the other hand, it has its own centralized related disadvantages such as the single point of failure and bandwidth/process scalability problem. The second solution is walking from the root node to the potential parent(s). In each step, the new node walks down the tree and gets the information of the current node's children. This method has been used in Overcast [10], NearCast [11], Nice [12], and Zigzag [13] protocols.

The second design problem is selecting the right parent. As mentioned in [14], finding a minimum spanning tree with constrained maximum degree on a graph is NP-Hard even in a centralized off-line situation. Therefore, protocol developers have tried historic algorithms to find a sub-optimal tree. Many proposed solutions can be categorized into locality-aware class. Locality-awareness has various techniques, which will be discussed briefly in the following section. Many other protocols such as [15], yoid [16], and SpreadIt [17] used end-to-end delay to choose the best parent. However, Overcast [10] uses hop-count measured by traceroute as a measure of network distance. NearCast [11] selects the parent node using geographical coordination, which is appropriate for regular national networks. DagStream [18] and LSONet [19] use hash functions on IP/ISP/Network-conditions information. Using a locality-aware DHT protocol (such as Pastry [20]) in Trickle [21] and SplitStream [22] is another method, which its efficiency in heterogeneous networks is debatable [23]. Using some landmark/lighthouse nodes is the final method which is proposed in Highways [24] and used in [25]. In this method, each new node finds its end-to-end delay from some predefined/dynamic landmark nodes and locates itself in a virtual space used to define locality. Locality-aware protocols are different not only in the method of finding the locality but also in how they use this measure. While nodes in mentioned protocols try to find the nearest parent to themselves, RaDiO [9] nodes favor parents near the source node and DHCM [26] nodes find a cluster in which nodes' distance to their parent is equal to its distance to the parent in order to create homogenous monotonic clusters in the tree. Finally, there are other protocols that mix a locality-aware method with a heuristic technique to use uplink bandwidth efficiently. For example, Overcast tries to put the new node as far away from the root while the available bandwidth of the parent node is greater than what the root node offers.

Loop avoidance is the third challenge to overcome in creating a tree structure. A simple solution is sending a detector packet containing the path from the root node to the leaves. Using Bloom filter [27] to reduce the packet size is suggested in [28]. DagStream uses another innovative method: Every node sends its level in the heartbeat packets to its children. If a loop is formed in a tree refinement

mechanism, this level will increase after sufficient refresh periods. Every node keeps track of its parent(s) (DagStream uses a multi-tree structure which will be discussed later.) using a variable and when finds the unusual value changes tries to break the loop.

The last design problem in the tree topology is its refinement. As there is the node churn in every P2P network, and we do not know when a "good" (strong, reliable, near, etc. depending on the protocol) node will join/leave the tree, it may lose its efficient structure; therefore, most protocols have a refinement algorithm. Some protocols move "good" nodes near to the root in order to reduce the tree depth resulting in a reduction of average delay. It is usually implemented using a parent-child switch mechanism (Overcast [10] and mTreebone [29]). A negative effect of these refinement algorithms is the frame losses during the switch period, solutions of which will be discussed in Section VI.

Low delay in sending the video (using the push mechanism) and simple send scheduling algorithm are two remarkable advantages for the tree structure. In the push mechanism, a parent-child link usually lives as long as the video stream does; as a result, the parent can make scheduling decisions easier. However, the single-tree topology has low uplink bandwidth efficiency because it does not use the leaves' uplink bandwidth, which also makes it unfair. Moreover, download rate of a node is limited by the download rate of its parent, which increases the importance of the level of nodes. It means that if an unstable low bandwidth link connects the root of a big sub-tree to others, all nodes of the sub-tree will receive low quality video while there is no way to compensate in case of errors. The last but not least point is about churn, which is an indispensable feature of a P2P network: if a node leaves the network, video reception in all its children will stop until its departure is detected and a new parent is found (Section VI explains this challenge more).

Multi-tree structure was proposed to solve single-tree problems. In multi-tree streaming, the server divides the stream into multiple sub-streams (using FEC, MDC, or layered coding, which will be discussed later) and creates a tree for each one. The sub-stream flows in the sub-tree down to its leaves. Each node joins to all/some sub-trees to get the video with the desired quality. Selecting the right trees is an additional challenge in this structure. Nodes may select the sub-tree randomly or by running a periodic quality check process (such as relative delay).

Multi-parent solutions are used to solve parent's bandwidth and loss constraints (if uses FEC). Multi-tree tries to solve the fairness problem by changing the role (interior/leaf) of nodes in each tree. However, these protocols have to manage multiple trees instead of one, which increases group management overhead and complexity of scheduling algorithms.

B. Mesh

Here, "mesh" means an irregular network which all links are active and may be used to receive video data or their related information. Each node may have multiple neighbors and exchanges its video contents with its neighbors, which means that it can send and receive a video stream from a node simultaneously. Mesh protocols usually use "Pull" method to get the video; in other words, each node sends video packets based on the received block-request. The block may be a frame, multiple frames, a GoP, multiple GoPs, or even a sub-stream (which closes this structure to the multi-tree topology [30]). As the size of block decreases, the protocol designer has more control in the scheduler module, but this may increase the overhead of control packets (usually because of request packets).

Challenges in mesh P2P networks are locating neighbors, neighbor selection, and mesh refinement. Protocols such as CoopNet [31], and Dagster [32] use bootstrap node solution to locate neighbors. In contrast, a received join request may be redirected to a new node in each step of random walk. This method may impose extra joining delay and in its general way, increases the overhead of initial/high degree nodes. (SCAMP [33] used in DONet [34]) and Biased Forwarding (Swaplinks [35] used in ChunkySpread [36]) have been proposed to address the asymmetric overhead problem. Note that random walk locates and chooses a neighbor within one step. The most common technique to locate neighbors is exchanging neighbors/known nodes list using gossiping (DONet, DagStream [18]), which is usually used after the first join. Other solutions such as suggesting a neighbor (DagStream) or using a DHT based middleware is also applicable.

A node must choose some neighbors after locating the potential ones. DagStream, LSONet, and

[15] use locality-aware techniques (discussed in Section III-A). Choosing nodes with similar bandwidth (Bullet' [37]), low-level nodes, topology-aware (PROMISE [38]), and data aware are other solutions. Data-aware methods try to join nodes based on a data-related *parameter* using a *mechanism* in order to reduce delay and network load. This parameter may be shared available data (PULSE [39] and DONet) or shared interests (current window) (PULSE and [15]). Some protocols (DONet and PULSE) use gossiping and others such as [15] rely on heartbeat messages to advertise the parameter. DHT-based networks and centralized solutions (like trackers in BitTorrent) are some other applicable techniques proposed for the mechanism. It is noteworthy that although in a live streaming session all nodes may want to see one part of a video, data aware method is useful to form a lag based mesh and applicable with scalable video streams or multi-session P2P structures (AnySee [40] and Trickle). Avoiding unreliable mesh clusters and handling NATs are the points which should be taken into consideration in this topic. It is possible to cluster nodes behind a NAT to use their uplink capacity in order to serve each other (STUN [41] and TURN [42]). Mesh refinement, as the last design problem, has been handled by probing the quality related parameters such as relative delay of the advertised data among different neighbors or simply their overlay distance to the source node (DagStream and ChunkySpread).

Mesh reliability against the churn thanks to the pull method (with small block size), efficient network resource usage, ability to use low uplink bandwidths (by swarming), and ability to use FEC codes are the advantages of the mesh structure against the single-tree topology. However, due to the pull method, it imposes long start-up delay and has large control overhead, especially in case small block size is used. Besides, mesh protocols need more complex scheduling procedures (in both sender and receiver).

C. Hybrid

Hybrid structures include P2P-CDN networks [43], mesh-DHT (structured) networks (Push-to-Pull [44]), and mesh-tree networks. We will discuss the last option more. ChunkySpread, LSONet, and [45] tried to make a multi-tree structure over a mesh network based on data-aware methods while the tree links may have shorter life than a stream to be flexible about changes using information received through the mesh network. MTreebone [29] creates a tree backbone near the source formed by reliable nodes while enjoys reliability of mesh structure for other nodes. It uses mesh structure in pulling data only for retransmission if a data outage occurs in tree-bone. TCMM [46] and Bullet [28] create a universal single-tree structure to deliver control messages. TCMM uses the tree structure only for delivery of control messages, so it avoids tree refinement in short sessions. On the other hand, Bullet uses both the tree and the mesh structure for data delivery. The idea is distributing as much data as possible through the tree using the push mechanism and exploiting the unused uplink bandwidth by pull requests. Although the Push-to-Pull protocol does not have a mesh-tree structure, it uses the same idea and tries to scatter fresh video data uniformly in the network using the prefixed structure (tree-like one) in the push phase and pulls them in the mesh network. Table I summarizes hybrid protocols specification.

IV. Data Delivery

After the overlay topology has been created, the mechanisms for sending and receiving video data should be designed. This section explains the challenges mostly related to the scheduling problems.

A. Bandwidth heterogeneity

Peer-to-peer nodes have various types and cover many clients with different bandwidth access. Each type may have its own quality requirements, but all clients want to see the video almost continuously. Using multiple video versions is not scalable because it increases user groups and imposes much load on the source node. However, the main well-known solution for this challenge is "adaptive quality". In Adaptive quality methods, each node receives the video stream with a quality based on its downlink capacity without imposing any/much bandwidth overhead on the nodes, especially the source node. Scalable video coding is the main approach for adaptive quality and can be accomplished by providing multiple versions of a video in terms of either amplitude resolutions (SNR scalability), spatial resolutions, temporal resolutions, frequency resolutions, or combination of these options and

TABLE I
HYBRID PROTOCOLS SPECIFICATIONS

Protocol	Data reception mechanism	Mesh usage	Tree usage	Coding	Tree link lifetime
ChunkySpread	Pull	Info [a] dissemination & find parent	Data delivery	MDC	Stream
Bullet	Push-pull	Pull data	Push data & info dissemination	MDC/FEC	Session
LSONet	Pull	Info Dissemination	Data delivery	Layered	Layer
Layered for incentive [45]	Pull	Info dissemination	Data delivery	Layered	Block (GoP)
TCMM	Pull	Info & data delivery	Control messages	-	Session
mTreebone	Push-pull	Pull data (only retransmission)	Push data	-	Session
Push-to-pull	Push-pull	Pull data	Push data (prefixed)	-	-

[a]info means any information about data in nodes or their properties

may be accessed either at frame level or object level (in the MPEG4 standard) [47]. On the other hand, there are two techniques for packetizing these versions:

1) Layered Video: 1) Layered Video: It divides the video into multiple layers each of which complements the lower one. This means that the base layer is decodable by its own, but the upper ones are only decodable if their dependencies are met. This dependency requires complex scheduling algorithms in order to use the bandwidth efficiently. Besides, it is likely that a top layer becomes blocked in the network due to the uplink bandwidth constraints; therefore, the protocol must guarantee that all layers would be available for high bandwidth nodes. [45] tries to deal with this problem by moving the high bandwidth peers near to the source node and clustering the peers by their bandwidth.

2) MDC: This technique creates multiple video versions similar to Layered Coding while trying to remove the troublesome dependency by adding redundant data to the layers. A simple way to do this is to run an FEC-like process on the layered video. Using this technique with redundancy parameter $p = 1 - k/n$, while a node receives $i < k$ different video packets for a frame, the video quality improves. The downside of MDC against layered coding is its large bandwidth overhead, which is illustrated in [45] for a nearly unique protocol with different video codings.

Transcoding the video frames in each level (Dagster [32]) and prioritizing the frames based on a distortion metric are other less favorite techniques. Transcoding is re-encoding the video frames with different qualities for differing nodes. Although MDC and layered video coding techniques have some computational overhead in the source and destination nodes, the computational overhead of transcoding is so high that makes it unsuitable for live video broadcasting. The latter technique is based on the distortion concept. Distortion is defined as the loss of quality in case of frame loss. It can also be defined using the PSNR measure and its value is transmitted as a control flow, simply as the bytes becoming undecodable, or even as a rough value assigned to a frame based on its type or dependent frames.

How to get general information about the streams (size, PSNR, etc.) is the design choice which is usually introduced while working with scalable video coding techniques. In video on-demand application, each node could download it along with the startup nodes list. However, in the live video streaming sessions, they should ask the source once in a while or receive it via a universal broadcast structure. Tree topology in the Bullet protocol is a good example of such structure. More research is, still, needed in this area of P2P streaming.

Dealing with low bandwidth links is the last point which needs mentioning in this subsection. A protocol may divide a video stream into multiple substreams in order to use low uplink bandwidth links. Moreover, this technique can be complemented by network coding. Network coding attracted attentions, especially in wireless P2P streaming; however, it needs more work to be compatible with MDC and layered coding, loss tolerant, and able to respect importance of different packets. Besides, protocols such as Trickle and AnySee try to use

inter-overlay capacities while they are running on multiple streaming sessions.

B. Error resiliency

Error is an inevitable part of a P2P network using ordinary machines connected by error-prone residential links. Besides, tree family structures are very vulnerable to the errors and propagate them down to the leaves. As a result, protocols use error resilient solutions. Retransmission is the first idea which springs to mind. A node may ask the incomplete frames from new parent(s)/partners (CoDiO [48]) or pull it in mesh in hybrid protocols (mTreebone [29]). CoDiO prioritizes the retransmission requests based on a distortion model to avoid creating congestion due to the extra requests. However, retransmission can increase delay, especially for live streaming and requires a larger play buffer in its simple version. In contrast to retransmission, other solutions engage sender nodes more: FEC adds data redundancy $(1 - \frac{k}{n})$ to enable clients to regenerate a healthy stream. However, it does not help clients to play video before receiving k blocks while it imposes a fixed bandwidth overhead. Protocols which use MDC coding also profit from this idea while they enable receivers to decode even one stream (adaptive quality feature). Layered coding also has this adaption, but it is more vulnerable to errors due to layers dependency and lack of protective redundancy. Furthermore, there are some codec specific features such as H.264 SP and SI frames [9], which have been used to stop error propagation.

Making the redundancy of FEC adaptive based on the network overhead is an immature idea in this field, which needs more elaboration. Besides, it is possible to protect the frames asymmetrically based on their distortion (frame type, PSNR, number of depending frames, layer number, etc.).

C. Congestion & Throughput Change

Video streaming consumes bandwidth intensively and its playback quality can be very sensitive to the link congestion. Besides, the topology dynamicity can change the load of shared links. On the other hand, most protocols use UDP transport protocol, which does not have any rate control policy, to deliver video packets. Some protocols use rate control methods to deal with this challenge. For example, Bullet uses TFRC [49]. [45] uses probing requests, high priority small packets, to evaluate the network condition.

Adaptive quality techniques have also been useful in this situation. PALS [50] uses layered coding. It monitors the quality of each overlay link while requesting video blocks. When it finds out congestion in a link, firstly, it tries to decrease bandwidth allocated to upper video layers (using a distortion model). Subsequently, it drops a layer by overriding its requests if the congestion lasts more. PALS will do the reverse actions if the situation improves. This technique is also applicable in MDC coding. As another technique, Bullet' has adaptive request size (block size) which enables it to guide the traffic quickly and precisely. Using a congestion-aware protocol, especially along scalable video coding, quality vibration becomes a sensible challenge. One of the interesting solutions is proposed by Dimitrios Miras in [51]. He uses an artificial neural network to predict the quality of future video and a fuzzy rate-quality controller considering properties of human quality perception to smooth streaming quality.

D. Data Delivery Models

We try to explain the data delivery models and their related challenges in this subsection. The first model is the pull model, which often accompanies the swarming technique. In the pull model, the client finds the appropriate peers which have the required video blocks, then *selects* a subset of them to send its *request* to. Then the selected peers *schedules* the answers to the received requests. Selecting the neighbors, filling the request packets, and scheduling the answers are the key design problems in this model. These scheduling tasks can be as hard as an NP-Complete problem [19] and not being aware of the future content properties due to the live nature of video would aggravate the situation.

Data-aware mesh protocols use the advertised information and choose the container node. Although there are many data-aware protocols such as LSONet and Bullet, DoNet is the well-known representative for this strategy. Data advertisement, here, means to let others know which video blocks a node has. Nodes may broadcast advertisement packets in one hop like in DONet, rely on a universal broadcasting structure (usually tree) like in Bullet (using a collect & distribute approach), or

inform some super-nodes like tracker approach as in BitTorrent-like protocols in order to distribute data advertisement. Overhead of Advertisement packets is the main challenge of this method. Bullet uses Bloom filter to reduce the size of these packets and PULSE takes a two-phase strategy. PULSE nodes, firstly, send their play buffer lag time according to the current video play-time in the server. Then each node selects a subset of its neighbors and exchanges detailed buffer map. LSONet uses a locality-aware approach while there are protocols which try a random walk query passing to reach the data holder.

The second design problem is filling the request packets. If nodes try to ask the beginning block in their interest window, neighborhood relation loses its efficiency and there will be few blocks to exchange with neighbors (Note that video playback time is almost synchronous in all nodes). The second choice is exploiting a random approach. However, there are other approaches which are more intelligent. DONet and PULSE nodes request the rarest chunk among their neighbors in hope for distributing the video blocks faster. Again, a node may take a content-aware approach and choose a distortion model to prioritize frames to be requested. LSONet uses layered coding and takes a greedy approach to maximize each node's received layers knowing available layers of its neighbors.

The third design problem relates to the sender and scheduling answers for the received requests. Besides the random approach, a PULSE sender node counts the number of send attempts for each video block and schedules the least sent blocks first in order to speedup delivery process. CoDiO [48] prioritizes frames based on the number of descendants in the receiver's sub-tree to increase the simultaneousness in the network. Content awareness and distortion models can also be useful at the server-side. Still, server-side scheduling is mostly affected by incentive policies, which will be discussed in Section V.

Pull model is based on the data dispersal in a node neighborhood to make an interactive exchange among the node and its neighbors; however, data clustering based on the neighborhood relation can degrade the pull model efficiency. Therefore, avoiding clustering of the video blocks in the network is a probable challenge which needs more consideration. Requesting the rarest block among neighbors and sending the least sent block can help break these clusters. Besides, there are hybrid protocols which use the push-pull model to overcome this challenge. They (Bullet and Push-to-Pull) use a structure to inject (push) fresh video blocks to the network. As a complementary technique when they use MDC coding, different streams will be pushed into children to increase the block diversity. The third model is the push model, scheduling of which is very simpler and only the server-side scheduling techniques mentioned earlier are applicable in it.

V. Incentive

This section talks about incentive which compels nodes to contribute in the peer-to-peer network. We considered three parts for an incentive policy: mechanism, input data for decision, and output behavior. Many researchers have worked on the mechanism. Using some monitor/super/trusted nodes [52] to monitor others' behavior is a simple semi-centralized solution which brings other design problems such as choosing, maintaining, and finding these nodes. In the reputation-based model, nodes' behaviors depend on the node's contribution to the whole network. The mechanism designer should choose a method to distribute the reputation value in the network and guarantee its integrity. In contrast to the reputation-based model, ChunkySpread and PULSE use a two-sided model like the well-known tit-for-tat model in which each node serves others based on the services it gets from them. Efficiency penalty and selfish behavior of the partners of the source node in this model are points that need more investigation. Generally, we can categorize these policies in the field of game theory and mechanism design, which are hot topics these days. After the designers chose the mechanism, they should determine input data and output behavior for it. Accepted requests, received video blocks, decodable/useful video blocks, or query forwarding resulting in a video reception or finding a "good" neighbor are good examples of input parameters. Increasing the responses delay and degrading the video quality are the main approaches to punish selfish nodes. Manipulating the sender-side scheduler to delay responses to selfish nodes, blocking their requests in random walk models, blocking them from good parent/partners, or limiting advertisement packets in data-aware approaches are possible solutions to increase the delay. In order to change the video

quality, the protocol can use a distortion model to prioritize the frames in the server-side scheduler or use an adaptive quality method. [45] used layered coding and [53] used MDC as a tool for applying an incentive policy.

VI. Group Management

Group management module tries to deal with the dynamicity of the P2P network throughout the video streaming session. In other words, it should define what a node should do if its parent/partner leaves. First, it should detect this event if the parent/partner nodes leaved ungracefully. Nodes in TCMM, ChunkySpread, RaDiO, mTreebone (in mesh), and DagStream use heartbeat packets to inform the others about their existence. Advertisement packets can also play the heartbeat role in data-aware protocols such as Bullet. Moreover, a quality monitoring process like what is implemented in NearCast and mTreebone (in tree) can detect the departure.

The next step is to find a new parent/partner. Re-running the topology creation process is a deficient choice as it has its own bandwidth overhead and causes a long gap in video reception. Overcast and NearCast nodes keep the grandparent node reference and ask it for a new parent. Other protocols such as Bullet and TCMM keep some inactive/candidate neighbors to use in case a parent detached from the overlay.

The last step is to recover the lost frames. If the protocol has used some redundancy in video coding or packetizing, the loss can be less severe, but when the loss is detected, first, the node should decide whether the frame could be recovered in the available time or not. Then it may ask the new parent/partner for the frames. Moreover, Hybrid protocols such as mTreebone can ask for lost frames in mesh. CoDiO uses a distortion model to prioritize the retransmission requests in the receiver side. [54] proposed a time-shifted stream which patches the lost frames. This solution could be useful even when the new parent/partners do not have the required packets although it is used in on-demand video streaming protocols [55] and its efficiency in live streaming needs to be evaluated more specifically.

Irrespective to the fact that the proposed algorithms are able to handle a few join/leave, their efficiency is questionable if many nodes join or leave (Flash). Flash join/leave is very likely, particularly in the beginning and ending of a popular show in live TV programs [56]. These flash joins/leaves can disfigure the topology and make it inefficient. Comparing different group management policies in the presence of this phenomenon would be an interesting topic to be surveyed in the literature.

VII. Conclusion

In this article, we noted the important challenges in designing a peer-to-peer live video streaming protocol. We categorized the challenges in four problem classes each of which considered the protocol from a particular point of view: topology creation, data delivery, incentive, and membership management. Moreover, this paper tried to gather the key proposed solutions for each challenge and explain the advantages and drawbacks of each solution. In addition, the solutions had been accompanied by their instances in existing protocols.

There are remaining subjects in wireless P2P video streaming, streaming on mobile/embedded systems, secure streaming in critical ad-hoc networks, video quality-aware congestion handling, Flash join/leave handling, and structures for broadcasting control packets parallel to data broadcasting, which need more investigations. Moreover, there are other semi-technical issues becoming increasingly important hereafter: How an on-line video streaming service should be priced considering the traditional TV broadcasting services as a rival/complement service? How earnings should be shared between content producers, source node, ISPs, and even hosts collaborating in video streaming? How to manage digital content copyright? Considering these challenges shows that there is a long way to go in the research area of P2P based live media streaming systems.

VIII. Acknowledgement

The authors would like to thank Seyyed Morteza Mousavi and Reza Motamedi for his meticulous editing comments.

References

[1] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys & Tutorials, IEEE*, pp. 72–93, 2005.

[2] E. Setton, P. Baccichet, and B. Girod, "Peer-to-Peer live multicast: A video perspective," in *Proceeding of the IEEE*, ser. 1, vol. 96, 2008, pp. 25–38.

[3] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-Peer Networking and Applications*, vol. vol. 1, 2008.

[4] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of Application-Layer multicast protocols," *Communications Surveys & Tutorials, IEEE*, vol. 9, no. 3, pp. 58–74, 2007.

[5] J. Liu, S. Rao, B. Li, and H. Zhang, "Opportunities and challenges of Peer-to-Peer internet video broadcast," *Proceeding of the IEEE*, vol. 96, no. 1, pp. 11–24, 2008.

[6] W. Gao and L. Huo, *Challenges on Peer-to-Peer Live Media Streaming*, ser. Lecture Notes in Computer Science, 2007, vol. 4577/2007, pp. 37–41.

[7] W. P. K. Yiu, X. Jin, and S. H. G. Chan, "Challenges and approaches in Large-Scale P2P media streaming," *IEEE MultiMedia*, vol. 14, no. 2, pp. 50–59, 2007.

[8] J. Price, "Christopher alexander's pattern language," *IEEE Transactions on Professional Communication*, vol. 42, no. 2, pp. 117–122, 1999.

[9] E. Setton, J. Noh, and B. Girod, "Rate-distortion optimized video peer-to-peer multicast streaming," in *Proceeding of the ACM workshop on Advances in peer-to-peer multimedia streaming*. Hilton, Singapore: ACM, 2005, pp. 39–48.

[10] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. J. O'Toole, "Overcast: reliable multicasting with an overlay network," in *Proceeding of operating systems design and implementation*, 2000, pp. 197–212.

[11] X. Tu, H. Jin, X. Liao, and J. Cao, "Nearcast: A locality-aware P2P live streaming approach for distance education," *ACM Transactions on Internet Technology*, vol. 8, no. 2, pp. 1–23, 2008.

[12] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *SIGCOMM '02: Proceeding of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, vol. 32. ACM Press, Oct. 2002, pp. 205–217.

[13] D. Tran, K. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *Proceeding of IEEE INFOCOM*, vol. 2, 2003, pp. 1283–1292.

[14] J. Dinger and H. Hartenstein, "On the challenge of assessing overlay topology adaptation mechanisms," in *Proceeding of the Fifth IEEE International Conference on Peer-to-Peer Computing*. IEEE Computer Society, 2005, pp. 145–147.

[15] F. Pianese and D. Perino, "Resource and locality awareness in an incentive-based P2P live streaming system," in *Proceeding of the 2007 workshop on peer-to-peer streaming and IP-TV*. Kyoto, Japan: ACM, 2007, pp. 317–322.

[16] P. Francis, "Yoid: Extending the internet multicast architecture," Tech. Rep., Apr. 2000. [Online]. Available: http://www.aciri.org/yoid/docs/index.htm

[17] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a Peer-to-Peer network," *Technical Report. Stanford InfoLab*, 2002.

[18] J. Liang, K. Nahrstedt, S. Chandra, and C. Griwodz, "DagStream: locality aware and failure resilient peer-to-peer streaming," in *Multimedia Computing and Networking*, vol. 6071. SPIE, 2006.

[19] H. Guo, K. Lo, and C. Cheng, "Overlay networks construction for multilayered live media streaming," in *Proceeding of the Eighth IEEE International Symposium on Multimedia*. IEEE Computer Society, 2006, pp. 427–436.

[20] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for Large-Scale Peer-to-Peer systems," *Lecture Notes in Computer Science*, vol. 2218, p. 329, 2001.

[21] Y. Guo, J. K. Zao, W. Peng, L. Huang, F. Kuo, and C. Lin, "Trickle: Resilient Real-Time video multicasting for dynamic peers with limited or asymmetric network connectivity," in *Proceeding of the Eighth IEEE International Symposium on Multimedia*. IEEE Computer Society, 2006, pp. 391–398.

[22] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in *Proceeding of the nineteenth ACM symposium on Operating systems principles*. ACM, 2003, pp. 298–313.

[23] S. Birrer and F. E. Bustamante, "The feasibility of DHT-based streaming multicast," in *Proceeding of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE Computer Society, 2005, pp. 288–298.

[24] E. Lua, J. Crowcroft, and M. Pias, "Highways: Proximity clustering for scalable Peer-to-Peer network," in *Proceeding of the Fourth International Conference on Peer-to-Peer Computing (P2P'04)*. IEEE Computer Society, 2004, pp. 266–267.

[25] M. Kleis, E. K. Lua, and X. Zhou, "Hierarchical Peer-to-Peer networks using lightweight SuperPeer topologies," in *Proceeding of the 10th IEEE Symposium on Computers and Communications*. IEEE Computer Society, 2005, pp. 143–148.

[26] S. sheng Yu, X. wei Zheng, and J. li Zhou, "A P2P scheme for live media stream multicast," in *Multi-Media Modelling Conference Proceedings, 2006 12th Internationa*, 2006, p. 4.

[27] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[28] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proceeding of ACM symposium on operating systems principles*. ACM, 2003, pp. 282–297.

[29] F. Wang, Y. Xiong, and J. Liu, "mTreebone: a hybrid Tree/Mesh overlay for Application-Layer live video multicast," in *ICDCS '07: Proceeding of the 27th International Conference on Distributed Computing Systems*. IEEE Computer Society, 2007, p. 49.

[30] N. Magharei and R. Rejaie, "Understanding mesh-based peer-to-peer streaming," in *Proceeding of the 2006 international workshop on Network and operating systems support for digital audio and video*. Newport, Rhode Island: ACM, 2006, pp. 1–6.

[31] V. N. Padmanabhan and K. Sripanidkulchai, "The case for cooperative networking," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, 2002, pp. 178–190.

[32] W. T. Ooi, "Dagster: Contributor-aware end-host multicast for media streaming in heterogeneous environment," in *Proceeding of SPIE Multimedia Computing and Networking (MMCN)*, 2005.

[33] A. J. Ganesh, A. Kermarrec, and L. Massouli, "Peer-to-Peer membership management for Gossip-Based protocols," *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 139–149, 2003.

[34] X. Zhang, J. Liu, B. Li, and T. Yum, "DONet/CoolStreaming: a data-driven overlay network for live media streaming," in *Proceeding of IEEE INFOCOM*, vol. 3, 2005, pp. 2102–2111.

[35] V. Vishnumurthy and P. Francis, "On heterogeneous overlay construction and random node selection in unstructured P2P networks," in *Proceeding of IEEE INFOCOM*, 2006, pp. 1–12.

[36] V. Venkataraman, P. Francis, and J. Cal, "Chunkyspread: Multi-

tree unstructured Peer-to-Peer multicast," *Proceeding of The 5th International Workshop on Peer-to-Peer Systems*, 2006.

[37] D. Kostic, R. Braud, C. Killian, E. Vandekieft, J. W. Anderson, A. C. Snoeren, and A. Vahdat, "Maintaining high bandwidth under dynamic network conditions," in *Proceeding of the annual conference on USENIX Annual Technical Conference*. Anaheim, CA: USENIX Association, 2005, p. 14.

[38] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast," in *MULTIMEDIA '03: Proceeding of the eleventh ACM international conference on Multimedia*. ACM Press, 2003, pp. 45–54.

[39] F. Pianese, J. Keller, and E. W. Biersack, "PULSE, a flexible P2P live streaming system." in *Proceeding of IEEE INFOCOM*. IEEE, 2006, pp. 1–6.

[40] X. Liao, H. Jin, Y. Liu, L. Ni, and D. Deng, "AnySee: Peer-to-Peer live streaming," in *Proceeding of IEEE INFOCOM*, 2006, pp. 1–10.

[41] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN: Simple traversal of user datagram Protocol(UDP) through network address translators (NATs)," Mar. 2003.

[42] J. Rosenberg, R. Mahy, and C. Huitema, "Traversal using relay NAT (TURN)," Feb. 2005.

[43] E. Setton and J. Apostolopoulos, "Towards quality of service for Peer-to-Peer video multicast," in *Proceeding of IEEE International Conference Image Process (ICIP)*, vol. 5, 2007, pp. V–81 – V–84.

[44] T. Locher, R. Meier, S. Schmid, and R. Wattenhofer, "Push-to-Pull Peer-to-Peer live streaming," in *21st International Symposium on Distributed Computing (DISC), Lemesos, Cyprus, Springer LNCS 4731*, Sept. 2007.

[45] Z. Liu, Y. Shen, S. Panwar, K. Ross, and Y. Wang, "Using layered video to provide incentives in P2P live streaming," in *P2P-TV '07: Proceeding of the 2007 workshop on Peer-to-peer streaming and IP-TV*. ACM, 2007, pp. 311–316.

[46] H. Jin, X. Tu, C. Zhang, K. Liu, and X. Liao, "TCMM: hybrid overlay strategy for P2P live streaming services," in *Proceeding of Second International Conference of Advances in Grid and Pervasive Computing*, vol. 4459. Paris, France: Springer, 2007, pp. 52–63.

[47] Y. Wang, J. Ostermann, and Y. Zhang, *Video Processing and Communications: Scalable video coding*. Prentice Hall, Oct. 2001.

[48] E. Setton, J. Noh, and B. Girod, "Congestion-distortion optimized peer-to-peer video streaming," in *Proceeding of IEEE International Conference Image Process (ICIP)*, Atlanta, GA, Oct. 2006.

[49] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceeding of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. Stockholm, Sweden: ACM, 2000, pp. 43–56.

[50] R. Rejaie and A. Ortega, "PALS: peer-to-peer adaptive layered streaming," in *Proceeding of the 13th international workshop on Network and operating systems support for digital audio and video*. Monterey, CA, USA: ACM, 2003, pp. 153–161.

[51] D. Miras and G. Knight, "Smooth quality streaming of live internet video," in *Global Telecommunications Conference*, ser. 29, vol. 2. IEEE, 2004, pp. 627–633.

[52] W. Conner and K. Nahrstedt, "Securing peer-to-peer media streaming systems from selfish and malicious behavior," in *Proceeding of the 4th on Middleware doctoral symposium*. Newport Beach, California: ACM, 2007, pp. 1–6.

[53] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "P2P video live streaming with MDC: providing incentives for re-distribution," in *IEEE International Conference on Multimedia and Expo*, 2007, pp. 48–51.

[54] M. Guo and M. Ammar, "Scalable live video streaming to cooperative clients using time shifting and video patching," in *Proceeding of IEEE INFOCOM*, vol. 3, Hong Kong, 2004, pp. 1501–1511.

[55] K. Hua, Y. Cai, and S. Sheu, "Patching: a multicast technique for true video-on-demand services," in *Proceedings of the sixth ACM international conference on Multimedia*. Bristol, United Kingdom: Proceeding of ACM multimedia, 1998, pp. 191–200.

[56] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *Proceeding of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. Portland, Oregon, USA: ACM, 2004, pp. 107–120.