A Progressive Flow Auction Approach For Low-Cost On-Demand P2P Media Streaming

Zongpeng Li, Anirban Mahanti†

Abstract— **Realizing on-demand media streaming in a Peer-to-Peer (P2P) fashion is more challenging than in the case of live media streaming, since only peers with close-by media play progresses may help each other in obtaining the media content. The situation is further complicated if we wish to pursue low link cost in the transmission. In this paper, we present a new algorithmic perspective towards on-demand P2P streaming protocol design. While previous approaches employ streaming trees or passive neighbour reconciliation for media content distribution, we instead coordinate the streaming session as an *auction* where each peer participates locally by bidding for and selling media flows encoded with network coding. We show that this auction approach is promising in achieving low-cost on-demand streaming in a scalable fashion. It is amenable to asynchronous, distributed, and light-weight implementations, and is flexible enough to provide support for *random-seek* and *pause* functionalities.**

I. Introduction

In this paper, we consider the problem of on-demand streaming of popular stored media files to clients on the Internet. Client requests for the same media may be asynchronous, and furthermore, clients may request playback from any position in the content and also engage in interactive operations such as pause and random-seek. In the conventional approach to media-on-demand, a separate unicast channel is allocated by the server for each client request. With this approach, the required number of server channels grows linearly in the client request rate, and the video server soon becomes the bottleneck.

Several solutions have been proposed to address the aforementioned scalability issue. One approach is to reduce the demands on server (and network) bandwidth by applying the "pull" strategy of caching media files, or portions thereof, at sites closer to the requesting clients. Alternatively, the "push" strategy of replicating media files at sites closer to requesting clients may be applied [4]. These solutions only partially address the problem; in extreme cases, these strategies may serve to only shift the load from the server to the proxies. Complementary to these infrastructure-based content distribution solutions are the scalable streaming proposals such as periodic broadcasts [1], [22], [26], patching [10], [11], and streaming merging [17], [18]. The basic idea behind these server-side solutions is to enable aggregation of client requests, wherein a large number of asynchronous client requests for a media file are served using a few multicast streams.

The limited deployment of IP multicast due to operational problems has resulted in considerable work on solutions that simulated multicast at the application layer. The scalable streaming protocols discussed above may be deployed, for example, with application layer multicast solutions that employ specialized overlay routers. Another option is to leverage the Peer-to-Peer (P2P) content distribution framework for on-demand media streaming. This approach has been widely used for efficiently downloading large files and enabling live streaming [5], [12], [28], [29], and more recently for media-on-demand applications [19]. Using the P2P framework has inherent advantages because the system scales with demand as more clients provide increased bandwidth, storage, and computational capabilities.

These previous streaming protocol design focus on system scalability and do not explicitly consider link cost. We argue that minimizing the aggregated link cost in media flow transmission is desired, since it in general leads to better quality streaming paths and stable data delivery. Application layer link cost can be defined upon end-to-end delay, loss rate, or a combination thereof. For instance, minimizing total transmission delay implicitly clusters physically close peers. Besides improving delay latency experienced by each peer, it also reduces bandwidth consumption at the IP layer. Minimizing loss rate based cost translates into stable throughput and less retransmission overhead. Therefore, we set

† The authors are affiliated with the Department of Computer Science, University of Calgary. Their email addresses are: {zongpeng,mahanti}@cpsc.ucalgary.ca.

both scalability and low transmission cost as the design goals of our on-demand streaming system. In particular, we wish to make sure that the cost minimization incurs minimal computation and communication overhead, so as not to sacrifice system scalability.

In this paper, we design a set of distributed algorithms that act in concert in the application layer to realize on-demand media streaming in a scalable fashion with low-cost. Our algorithms include a mesh building module and a flow auction module, and work in concert with randomized network coding [2], [21]. The mesh building module maintains an acyclic overlay mesh of participating peers, in approximate order of their relative media play progress. The acyclic property helps reduce algorithm complexity, while the play progress ordering ensures that neighbor peers in the mesh may help each other in obtaining the media content using their buffer. The flow auction and code construction modules together replace traditional multicast tree algorithms. Encoded media flows are routed along the mesh through 'auctions'. Each peer is guaranteed to receive a set of innovative flows, from which the original media flows are recovered for playback. The auction module ensures good-quality links are utilized in flow routing by minimizing aggregated link cost in the transmission. The auction solution subsumes tree-based approaches and may achieve lower overall cost. It also saves the overhead constructing and maintaining capacity-disjoint trees. Compared with network flow based routing solutions [32], [24], the auction approach has much lower computational complexity, and is more amenable to practical implementations.

The three major contributions of this paper are summarized in the following list:

- A new algorithmic perspective on solving low-cost media flow routing as auctions. We depart from previous tree based and network flow based routing solutions, and adopt the auction method for cost minimization with low overhead.
- The adaptation of auction algorithms to the on-demand streaming scenario with overlay dynamics, and the analysis of its correctness and optimality. We exploit the specific structure of the media flow routing problem to improve the convergence speed of the auction algorithm, and explain the dependence of its correctness on properties of the overlay mesh and the streaming application.
- The design of an accompanying streaming mesh construction algorithm tailored for scalable on-demand media streaming.

The rest of the paper is organized as the follows. We review related research in Sec II, describe the streaming system architecture and the mesh building mechanism in Sec. III, present the flow auction algorithms for media flow routing in Sec. IV, and conclude the paper with related discussions in Sec. V.

II. Related Work

Several recent work developed peer-to-peer streaming systems through application layer multicast [12], [16], [19], [29], [36], [31]. Zigzag [31] builds hierarchical application layer multicast trees to support live streaming to large numbers of receivers. The tree building approach used in Zigzag is similar to that of NICE [5]. CoopNet [29], [28] builds multiple distribution trees and uses multiple description coding (one coding per tree) to provide robustness, both with respect to network paths and data delivery. CoopNet, although developed primarily to support live streaming, can support on-demand streaming as well. SplitStream [12] is a high bandwidth content distribution system built on top of the Scribe protocol that like CoopNet uses multiple application layer trees to support streaming.

The aforementioned systems focus on live streaming. P2Cast [19] and P2VoD [16] explicitly consider the problem of on-demand media streaming systems in a peer-to-peer setting. P2Cast is based on patching [10], [12]. Using a tree-first approach, P2Cast builds application layer multicast trees that consists of peer that are close together in terms of their play progress. This tree is built by considering the bandwidth among participating peers. The P2VoD scheme also builds an application layer multicast tree. It also employs caching at peer nodes and grouping of peers, such that later arriving peers obtain the media from only one earlier arrived peer. Our streaming algorithm design is different in two aspects. First, we use auction and network coding, instead of multicast trees, for media flow routing. Second, we explicitly set cost minimization as one of the primary goals to achieve.

The auction algorithm was initially designed by Bertsekas [8] for the assignment problem, which is equivalent to weighted bipartite matching in graph theory [33]. Later it was adapted to solve other network optimization problems, including transportation [9], shortest path [6], and network flow [7], [3]. The algorithm resembles real-world auctions, is highly intuitive and easy to understand. It also achieves comparable or better time complexity than other classic algorithms (*e.g.*, the Hungarian algorithm in the case of assignment [30], the network simplex algorithm in the case of network flow [3]). In this paper, we transform the min-cost media flow dissemination problem into an auction, and adapt the classic auction algorithm to compute the optimal flow routing scheme in a dynamic overlay network environment.

In previous work [24], [25], [32], we have studied the design and implementation of network flow based data dissemination algorithms in the overlay environment, with general cyclic topologies and the presence of relay nodes. The computation and communication overhead imposed on each peer are still relatively high, even with efficient network flow modules applied, such as the push-relabel algorithm [3] or the ϵ-relaxation algorithm [7]. The streaming session considered in this paper is essentially a broadcast without relay nodes, which allows us to make a fundamental trade-off between complexity and optimality by further restricting flow routing in *acyclic* meshes. By doing so, the per-node complexity is decreased from $O(n^3)$ to $O(n \log n)$. A similar acyclic-topology-only trade-off was also made by Ho *et al.* in [20] for randomized network coding.

III. Auction Based On-Demand Streaming System Architecture

A. System Model and Assumptions

Our streaming algorithm design targets on-demand streaming of a popular media file. The number of requesting peers is on the order of thousands, and each peer freely selects which part of the media it wishes to watch at any time. Let T denote the time length of the media. Each peer u is associated with a play progress $\tau(u) \in [0, T]$, which gradually grows in the normal play mode, and jumps upon a random-seek. We assume the total inbound and outbound bandwidth capacity at a peer u is bounded by $c_i(u)$ and $c_o(u)$, respectively. The media file is separated into h orthogonal flows/streams before dissemination. We assume the values of $c_i(u)$ and $c_o(u)$ are in multiples of a unit media flow rate. Each peer u maintains both a forward buffer $B_\uparrow(u) = [\tau(u), \tau(u) + b_\uparrow]$ and a backward buffer $B_\downarrow(u) = [\tau(u) - b_\downarrow, \tau(u)]$, where $b_\uparrow$ and $b_\downarrow$ are the lengths of the two buffers respectively. The forward buffer $B_\uparrow(u)$ helps achieve smooth playback at u, and the backward buffer $B_\downarrow(u)$ makes it possible for u to help downstream peers. A potential mesh link $\overrightarrow{uv}$ exists if $\tau(v) \in B_\downarrow(u)$. We assume a third-party overlay link quality monitoring program reports the *cost* of each link $\overrightarrow{uv}$, in terms of delay, loss rate, or another cost parameter of choice. Finally, we focus on feasible streaming scenarios by assuming that $c_i(u) \geq h$ for each peer u, and that the total outbound capacity of a peer group is always sufficient to meet the total flow demand of the corresponding peer group during overlay flow routing.

B. Overview of Streaming System Design

Our streaming algorithm design is positioned at the application layer, to be run in a peer-to-peer fashion. It consists of a set of algorithm modules centered around a min-cost flow auction algorithm, as shown in Fig. 1.

The mesh construction module coordinates a large, dynamic set of peers into an acyclic overlay mesh with good connectivity, serving as the topology upon which media flows are routed. Topological order of nodes in the mesh conforms to temporal order of nodes in play progress. This guarantees that neighboring peers are connected by a potential media provision link, even with very mild assumptions on node buffer length. The mesh module directly handles node joins/leaves and random-seeks. It also maintains high end-to-end connectivity with moderate node degrees.

The central module of the streaming algorithm set is the flow auction algorithm. It takes as input the mesh topology and link costs, and computes a feasible media flow routing scheme at optimal cost. Each peer acts as a *'buyer'* for its incoming flow demand, and bids for available outbound flows at its upstream neighbors. Conversely, it also acts as the *'seller'* for its outgoing flow capacity, and accepts bids from its downstream neighbors. The highest bid wins a flow capacity and leads to an active flow connection. The auction algorithm maintains only 1-hop local information on each peer. A sequence of flow allocation and re-allocations in the auction gradually leads to a better flow routing scheme with lower global cost. The auction algorithm adapts au-

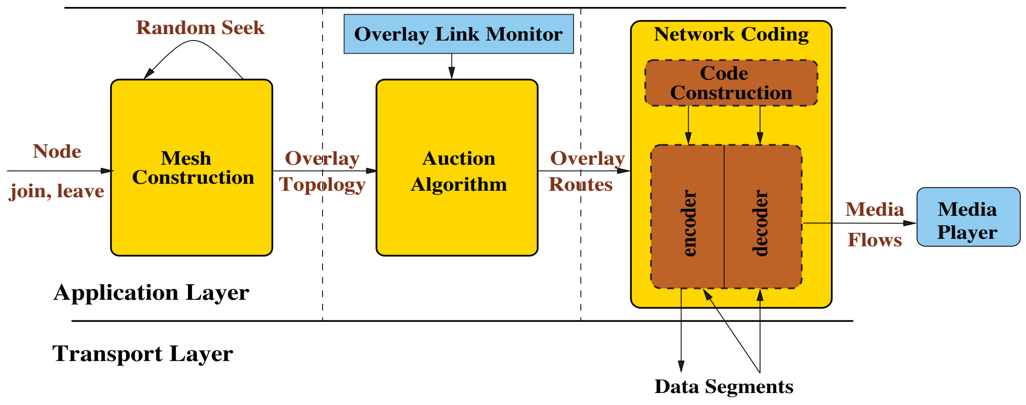

Fig. 1. The streaming algorithm modules and their interactions.

tomatically to node joins/leaves and random-seeks, and strives to progress towards the optimal operation point, which is a moving target. Besides asynchrony and distributed operation, the auction algorithm is also computationally much cheaper than network flow based routing algorithms [24], [32]. This improvement is due, in part, to the fact that the auction algorithms runs on an acyclic mesh instead of a general mesh topology.

The flow auction algorithm determines the flow rates between peers, we then rely on a randomized network coding module for determining the content of each flow, where each unit flow is set as a linear combination of the h source media flows. In the rest of this section, we briefly describe a mechanism for construction of an acyclic mesh, and then proceed to present the flow auction algorithm in Sec. IV.

C. Streaming Mesh Construction

There exist a large body of work studying general overlay mesh construction (*e.g.*, [34], [5], [14]). The focus is often on building a mesh with good connectivity and good performance in terms of delay, bandwidth, or node stretch. In the case of on-demand streaming, however, locality of playback progress overrides such conventional quality metrics, since a peer u may help a peer v only if the play progress of v falls within the range of the backward buffer at u. Therefore, it is natural to organize peers along the time dimension, and have media flows relayed from peers with more advanced play progresses to peers that are behind in play progress.

An acyclic mesh based on play progress

We organize peers participating in the on-demand streaming session into an acyclic overlay mesh such that peers with shortest distances in play progress become neighbors. Specifically, each peer u maintains a list of k other peers with shortest playback distance. These upstream neighbors in the mesh will serve as potential flow providers in the flow auction algorithm. Conversely, each peer will also be added into the neighbor list by k downstream peers with closet playback distance, as illustrated in Fig. 2.

For every time period $\Delta\tau$, a node u reports its IP address ip_u and current progress $\tau(u)$ to the media server s, with a probability p inversely proportional to the play progress span of u's neighbors, $\max_{v_1 \in N_\uparrow(u), v_2 \in N_\downarrow(u)}(\tau(v_1) - \tau(v_2))$. Upon receiving such a progress report at time t, the media server records a corresponding triple in the form of $<ip, \tau, t>$. A record is discarded after a certain time threshold. A new node or a random-seeking node u contacts the media server with a desired play progress $\tau_0(u)$. The media server replies with a closest peer v, based on information available in the list of progress reports. Then starting at v, u uses 1-dimensional geographical routing (on the axis of $\tau \in [0, T]$) to locate peers belonging to its neighbor sets $N_\uparrow(u)$ and $N_\downarrow(u)$.

Mesh maintenance

Upon joining the mesh, a peer u identifies peers in its neighbor set as described above, then sends a short notice message to them. Each upstream (downstream) neighbor then replaces the farthest peer in its downstream (upstream) neighbor list with u. In case the replacement results in the interruption of an active flow transmission, it will be automatically handled by the progressive flow auction algorithm, as we will describe in Sec. IV.

When a node u leaves its current position of the mesh due to disconnection or random-seek, we assume the departure of u is detected by a heartbeat mechanism or an intent-to-leave message, respectively. The repair of the mesh essentially involves a one-to-one mapping between peers in $N_\downarrow(u)$ and

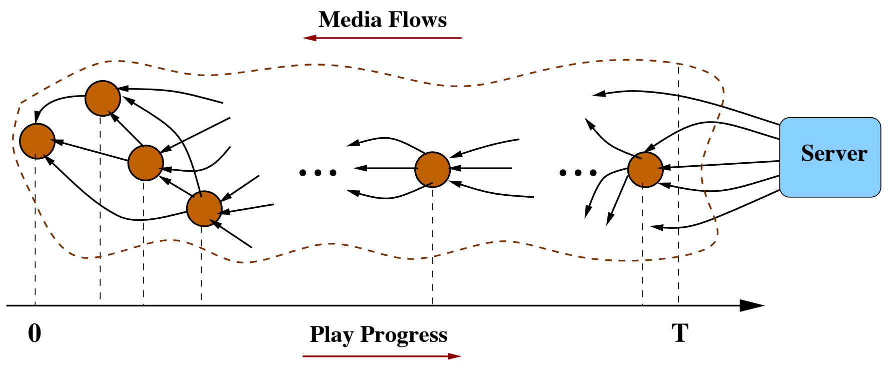

Fig. 2. Conceptual illustration of an acyclic mesh based on play progress, with $k = 3$.

$N_{\uparrow}(u)$, in order of play progress. Each former downstream neighbor replaces u with a correspondent former upstream neighbor of u.

End-to-end connectivity guarantee

Consider a cut of the mesh. The size of the cut is minimized when it separates the mesh into exactly two partitions. In this case, the number of links in the cut connecting two peers with 0, 1, ..., $k-1$ peers in between on the τ axis are 1, 2, ..., k, respectively, except at the k peers with earliest play progress. Therefore, the edge connectivity of the mesh is $k(k-1)/2$, quadratic to node degree $2k$. Only a moderate size of k is required to guarantee good end-to-end mesh connectivity.

A final remark is that algorithm details in the mesh module is transparent to flow auction. It is possible to fine-tune and improve the mesh module independently, as long as the output is acyclic and well-connected.

IV. Media Flow Auction

We now present the flow auction algorithms for min-cost media flow routing within the overlay mesh. We first review the simple auction algorithm, then transform the media flow routing into an auction. We next adapt the simple auction algorithm to solve it with improved convergence speed, and finally enhance the algorithm to handle overlay dynamics and random-seeks.

A. The Simple Auction Algorithm

The original auction algorithm solves the assignment problem, where a set of n objects ($\mathcal{O}$) are to be assigned to a set of n persons ($\mathcal{P}$). The goal is to find a one-to-one mapping between persons and objects such that the total value realized at all persons $\sum \delta(i, o)$ is maximized over a given person-object benefit function $\delta : \mathcal{P} \times \mathcal{O} \rightarrow \mathcal{Q}_+$. Here $\mathcal{Q}_+$ denotes the set of non-negative rational numbers.

The execution of the auction algorithm resembles a real-world auction, where the persons bid for the objects and the highest bids win. Auctions of all objects happen simultaneously in parallel. Each object is associated with a price that reflects the highest bid received so far. A person places a bid to the "best" object based on the profit an object provides and its associated price, trying to maximize the net profit. Each object is temporarily sold to the current highest bid, which may be overridden by a subsequent higher bid. The auction terminates when every object is sold, every person stops bidding, and every sale becomes final.

More specifically, to place a bid, a person i computes the net profit each object o provides, as $\beta(i, o) = \delta(i, o) - p(o)$. Next i identifies the most and the second most attractive objects o^* and o' as follows:

$$o^* = argmax_{o \in \mathcal{O}} \beta(i, o), o' = argmax_{o \in \mathcal{O} - o^*} \beta(i, o)$$

Then i sends a bid $b(i, o^*)$ to o^*:

$$b(i, o^*) = p(o^*) + \beta(i, o^*) - \beta(i, o')$$

Two facts are related to the choice of this bid value. First, i has no incentive to bid any higher — otherwise it may obtain a better net profit by competing for the second best object o'. Second, the auction turns to converge faster with higher bids and higher object price increases. Therefore it is common practice to choose the highest bid possible, although a lower bid between $p(o^*)$ and $b(i, o^*)$ also works. On the side of an object o, it simply accepts the highest bid $b(i, o)$, updates its price $p(o)$ to $b(i, o)$, associates itself with i, and removes its previously associated person, if any.

The auction algorithm is rather intuitive and easy to understand. Nonetheless, it is proved to be correct and efficient [8], [7]. It also allows natural parallel or distributed implementations, in either synchronous (persons bid in rounds) or asynchronous mode (each person bids at a time of its own choice). A final note is that if the best and second best net profits might be even, a small constant ϵ can be added on the bid $b(i, o^*)$ to break the tie. For further details, including the valid range of ϵ, we refer the readers to [7], [9].

B. Media Flow Auction - Static Version

In this section, we present a static version of the media flow auction algorithm. It is based on the simple auction algorithm discussed above. We assume a fixed mesh topology without node joins/leaves or random-seeks, and compute a min-cost flow routing scheme. Recall that the overlay mesh construction sub-layer coordinates the peers into a directed acyclic topology $G = (V, A)$, with $s \in V$ being the media server. The desired flow dissemination rate equals h, the number of original media flows. Let $w : A \rightarrow \mathcal{Q}_+$ be the link cost function. We wish to compute a feasible overlay flow routing scheme, respecting node capacity bounds, and minimizing the aggregated link cost. This problem can be cast into an integer linear program:

$$\text{Minimize} \quad \sum_{\vec{uv}} w(\vec{uv}) f(\vec{uv})$$
Subject to:
$$\begin{cases} \sum_{u \in N_\uparrow(v)} f(\vec{uv}) = h & \forall v \neq s \quad (4.1) \\ \sum_{u \in N_\uparrow(v)} f(\vec{uv}) \leq c_i(v) & \forall v \neq s \quad (4.2) \\ \sum_{v \in N_\downarrow(u)} f(\vec{uv}) \leq c_o(u) & \forall u \quad (4.3) \end{cases}$$
$$f(\vec{uv}) \in \{0, 1, 2, \ldots\} \qquad \forall\, \vec{uv}$$

Recall that every peer is assumed to have enough bandwidth to receive media flows at the playback rate h, therefore (4.2) is redundant given (4.1). Furthermore, it is critical to note that the coefficient matrix formed by constraints (4.1) and (4.3) is *totally unimodular* [30], [33], *i.e.*, every maximal square sub-matrix of it has determinant of 1 or -1. By linear optimization theory [30], this implies if we relax this integer program into a (continuous) linear program, all basic feasible solutions are integral and the same optimal solution will be obtained. This reduces the complexity of the problem from solving an IP (NP-hard in general) to solving an LP.

$$\text{Minimize} \quad \sum_{\vec{uv}} w(\vec{uv}) f(\vec{uv})$$
Subject to:
$$\begin{cases} \sum_{u \in N_\uparrow(v)} f(\vec{uv}) = h & \forall v \neq s \quad (4.4) \\ \sum_{v \in N_\downarrow(u)} f(\vec{uv}) \leq c_o(u) & \forall u \quad (4.5) \end{cases}$$
$$f(\vec{uv}) \geq 0 \qquad \forall\, \vec{uv}$$

The corresponding dual linear program is:

$$\text{Maximize} \quad \sum_{v \neq s} h r(v) - \sum_u c_o(u) p(u)$$
Subject to:
$$r(v) - p(u) \leq w(\vec{uv}) \quad \forall\, \vec{uv} \in A \quad (4.6)$$
$$p(u) \geq 0, r(u) \ \text{unconstrained} \qquad \forall u$$

Now the underlying connection between our media flow routing problem and the auction method is rather evident: primal variables $f(\vec{uv})$ can be viewed as flows 'sold' by u to v, and dual variables $p(u)$ can be viewed as node prices. It is natural to perform primal-dual optimization through flow auction. We now describe the detailed transformation of min-cost media flow routing into auction, adapt the simple auction algorithm to solve it, and show the correctness of the algorithm within the primal-dual framework.

There are two steps involved in the problem transformation. First, we need to relate cost minimization with value maximization. This can be achieved by replacing the link cost $w(\vec{uv})$ with a gain $W - w(\vec{uv})$, for some large constant W. The constant W can be dropped without affecting the solution. Second, we need to create a 'person' for each desired incoming flow at a peer, and an 'object' for each unit outgoing capacity of a peer, respectively. Fig. 3 illustrates how this transformation may be performed using a simple mesh topology.

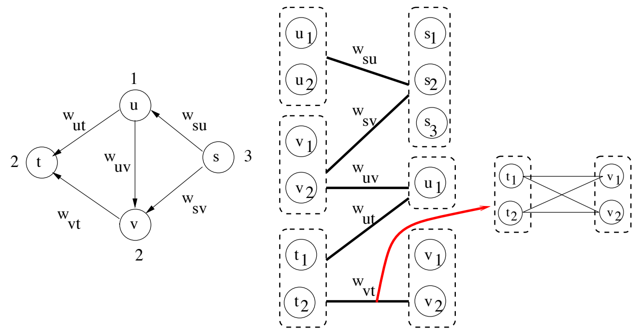

Fig. 3. Transformation of the min-cost media flow dissemination problem into an auction problem. Left: instance of flow auction. Numbers beside nodes denote outgoing capacities. Right: instance of auction. Left column contains persons, right column contains objects. Bold edges between two virtual nodes represent full connection between nodes therein.

After the transformation, we can apply the simple auction algorithm to compute the min-cost flows. However, our flow routing problem has a special structure that can be exploited to improve the convergence speed of the auction. If i and j are two persons created for flow demand at the same peer u, then for any object o, i is connected to o iff j is connected to o, and in that case $\delta(i, o) = \delta(j, o)$. This is similar to the auction algorithm design for the transportation problem in [9], where similar persons are created for stock at each source and similar objects are created for capacity at each sink. We apply the idea of coordinating bids from similar persons and to similar objects presented in the Auction-SOP algorithm in [9], to avoid unnecessary intra-node competition. Each node will bid on behalf of all its flow demand, and accept bids on behalf of all its flow supply, in a coordinated way as shown in table I. The algorithm maintains a single price $y(\overrightarrow{uv})$ for each current link flow $f(\overrightarrow{uv})$. A peer v that has not established all h incoming flows bids for flow capacities at an upstream neighbor u that is either unassigned or assigned to another competing peer k. The auction terminates when each peer successfully receives h incoming flows. For the ease of presentation, unassigned flow capacity on u is viewed as a flow from u to itself.

We now show the correctness of the auction solution in the static case.

Theorem 4.1 The static media flow auction algorithm converges to an optimal flow routing scheme with minimum aggregated link cost.
Proof: The proof consists of two parts: (1) the primal and dual LPs correctly model our static min-cost flow routing problem, and (2) the static flow auction algorithm correctly solves these LPs.

(1): We need to show that if each peer u receives h incoming flows, then it is possible to arrange the content of each flow in the routing scheme, such that every peer u receives h innovative flows, without duplication or redundancy. This can be done by a structural induction along the acyclic flow routing topology. If every upstream neighbor of u successfully receives h innovative flows, then it is obvious that u can arrange its incoming flows to be all innovative, too. Note that this is not true if the topology may contain cycles or if pure relay nodes exist.

(2): As in most correctness proofs of auction algorithms, we show that the auction terminates; and upon termination, there exists a feasible dual price vector p that jointly satisfy complementary slackness conditions with f. By results in linear programming, a pair of feasible primal and dual solutions are both optimal if and only if they satisfy the complementary slackness conditions [30], therefore the static flow auction algorithm is correct.

The fact that the auction eventually terminates can be shown by way of contradiction. Suppose it never terminates. Note that the set of assigned flows monotonically grows — an allocated flow capacity remains allocated throughout the auction. The total size of this set is upper-bounded by total flow demand in the network. By our assumption that the total capacity supply is sufficient, there will be flow capacities that are never assigned. Therefore, we have a peer bidding for an assigned flow whose price is growing unbounded, rather than bidding for an unassigned flow with price zero. This implies the link cost associated with the unassigned flow is infinite, contradicting the fact that each link cost is finite.

Note that vector y can not be directly interpreted as dual prices. It does not even fit into the dual LP, since it has an entry for each link instead of for each node. We instead construct a dual price vector $p(u) = \min_{v \in N_{\downarrow}(u)} y(\overrightarrow{uv})$ from y. The complementary slackness conditions for the static media flow auction LPs are:

$$\begin{cases} p(u) > 0 \rightarrow \sum_{v \in N_{\downarrow}(u)} f(\overrightarrow{uv}) = c_o(u) & \forall u \qquad (4.7) \\ f(\overrightarrow{uv}) > 0 \rightarrow r(v) - p(u) = w(\overrightarrow{uv}) & \forall \, \overrightarrow{uv} \quad (4.8) \end{cases}$$

The condition in (4.7) states that a node u with non-zero price $p(u)$ must have all its outgoing capacities assigned. This is obviously true in the auction. Further more, by observing that for a fixed price vector p, $r(v) = \min_{u \in N_{\uparrow}(v)}(p(u) + w(\overrightarrow{uv}))$ is required to maximize $\sum_{v \neq s} hr(v) - \sum_u c_o(u)p(u)$, (4.8) can be transformed into the proposition that every non-zero flow $f(\overrightarrow{uv})$ is locally lowest-cost possible. That agrees with the greedy nature peers bid for available flows in the auction. □

With careful implementation, the auction algorithm may achieve time complexity $O(nm \log n)$ [9], where n is the number of 'persons' and m is the number of links in the input bipartite. In our case, each node has a fixed degree $2k$, therefore the complexity is equivalent to $O(n^2 \log n)$, with per-node total complexity being $O(n \log n)$. On the other hand, for network flow based optimization [32], [25], the per-node complexity is equivalent to a max-flow computation, which is typically on the

TABLE I

THE STATIC MEDIA FLOW AUCTION ALGORITHM

```
**Initialization**
$f(\vec{uv}) = 0$, $y(\vec{uv}) = 0$, $\forall \vec{uv} \in A$; $f(\vec{uu}) = c_o(u)$, $y(\vec{uu}) = 0$, $\forall u \in V$

**Bidding at peer $v$**
*If* $\sum_{u \in N_{\uparrow}(v)} f(\vec{uv}) < h$, *then* :
    $\forall f(\vec{ut}) > 0$, $u \in N_{\uparrow}(v)$: $\alpha(v, \vec{ut}) \leftarrow y(\vec{ut}) + w(\vec{uv})$
    select $u_1 t_1, \ldots, u_l t_l, u_{l+1} t_{l+1}$ in decreasing order of $\alpha(v, \vec{ut})$, s.t. $\sum_{1 \le i \le l} f(\vec{u_i t_i}) \ge h$
    $\forall b = 1..l : b(v, \vec{u_i t_i}) \leftarrow y(\vec{u_i t_i}) - \alpha(v, \vec{u_i t_i}) + \alpha(\vec{u_i t_{l+1}})$
    *for* $i = 1..l - 1$:
        *if* $t_i \ne v$, send bid $b(v, \vec{u_i t_i})$ to $u_i$ for flow increment $f(\vec{u_i t_i})$
            *else* send price update $b(v, \vec{u_i v})$ to $u_i$
    *if* $t_l \ne v$, send bid $b(v, \vec{u_l t_l})$ to $u_l$ for flow increment $h - \sum_{1 \le i \le l-1} f(\vec{u_i t_i})$
        *else* send price update $b(v, \vec{u_l v})$ to $u_l$

**Flow Assignment at peer $u$**
Upon receiving a bid $b(v, \vec{uk})$ for increment $\Delta f$:
    *if* $b(v, \vec{uk}) \le y(\vec{uk})$, ignore; else:
        $f(\vec{uk}) \leftarrow f(\vec{uk}) - \Delta f$; $f(\vec{uv}) \leftarrow f(\vec{uv}) + \Delta f$; $y(\vec{uv}) \leftarrow b(v, \vec{uk})$
Upon receiving a price update $b(v, \vec{uv})$:
    $y(\vec{uv}) \leftarrow b(v, \vec{uv})$
```

order of $O(n^3)$ [3].

C. Media Flow Auction - Dynamic Version

While the static flow auction algorithm takes a fixed mesh topology as input, real streaming sessions often see highly dynamic node participation, with frequent joins, leaves, and random-seeks. We now further adapt the auction algorithm to handle such system dynamics. Each peer still bids for incoming flows in a similar way as in the static case. The algorithm progressively evolves towards an optimal solution. In cases of a high level of system dynamics, optimality rapidly varies over time, and the system may not be operating at the optimal state at any particular time point. In that sense, we have a constantly evolving auction responding to input interruptions, and pursuing a moving optimal operation point. However, if such dynamics slow down or stop, the auction algorithm will soon adjust the flow routing scheme to optimal.

Besides bidding and assignment steps specified in Table I, the dynamic flow auction algorithm further incorporates node join/leave, as well as two new mechanisms: *active rebidding* and *price drop*, as shown in Table II. A random-seek is equivalent to a leave followed by a join, assuming the new play progress is in general not close enough to be handled by the media buffer.

Basically, node joins and leaves informed by the mesh sub-layer are handled automatically, only variable initialization or reset is involved. However, a current set of optimal flows for a peer u may become sub-optimal after a competing peer leaves. Therefore, for every time window Δt (adjustable), u checks its local flow optimality. This involves computing the best possible flow set using essentially the same steps in the bidding procedure, and compare it to the current set. If a better set is possible, u releases a subset of high cost flows and resumes bidding. Correspondingly, price of the released flows is cleared to zero.

A nice property of the dynamic auction algorithm is that, no matter how long the streaming session (and hence the auction) lasts, flow prices remain bounded. More specifically, any price $y(\vec{uv})$ is upper-bounded by link cost difference $\max_{u \in N_{\uparrow}(v)} w(\vec{uv})$. The dynamic auction also terminates if the system state stops varying, since price drops and active re-bidding's do not violate complementary slackness.

TABLE II

DYNAMIC MEDIA FLOW AUCTION: EXTRA MECHANISMS

```
Upon join of $u$:
  $u$, and each $v \in N_{\uparrow}(u) \cup N_{\downarrow}(u)$: clear flows $f$ and prices $y$ to and from $u$ to $0$
    (Auction continues with participation of $u$)

Upon leave of $u$:
    Each $v \in N_{\uparrow}(u)$: $f(\vec{vv}) \leftarrow f(\vec{vv}) + f(\vec{vu})$
    Each $v \in N_{\uparrow}(u) \cup N_{\downarrow}(u)$: remove entries in $f$ and $y$ to and from $u$
    (Auction continues)

Active re-bidding:
A peer $v$ with $\sum_{u \in N_{\uparrow}(v)} f(\vec{uv}) = h$:
    after each time window $\Delta t$, compute new potential total incoming flow cost
    if new cost lower than current
      send flow release message to $u^* = argmax_u \alpha(v, u)$
      $f(\vec{uv}) \leftarrow 0$ (resume bidding)

Price drop:
Upon receiving a flow release message from $v$, a peer $u$:
    $f(\vec{uu}) \leftarrow f(\vec{uu}) + f(\vec{uv})$; $f(\vec{uv}) \leftarrow 0$, $y(\vec{uv}) \leftarrow 0$
```

D. Determining Flow Content

The flow auction module essentially computes a flow routing scheme, which specifies a flow rate (possibly zero) between each pair of neighboring peers. One may either decompose pair-wise capacity-disjoint trees from the flows, or apply randomized network coding on the flows, to determine the content of each flow. In the former case, each tree is used to transmit an uncoded media flow; in the later case, each link flow is determined as a linear combination of the h original media flows.

V. DISCUSSIONS AND CONCLUDING REMARKS

In this paper, we introduced a set of algorithms that jointly realize low-cost on-demand streaming in the application layer. The central module is a progressive flow auction algorithm that performs min-cost overlay flow routing in a distributed, light-weight fashion. This is to be contrasted with previous approaches based on network flows (higher per-node complexity) or streaming trees (lacks optimality guarantee on cost). We also presented an accompanying mesh building algorithm based on play progress, and discussed the application of network coding to achieve better robustness and lower transmission delay.

During the start-up phase of the streaming session, peer density on the playback axis τ may not be sufficiently high to guarantee mesh connectivity. To address this issue, we can introduce a link from the server s to every peer u into the mesh. However, in order to reduce the capacity burden on s, flow prices $y(\vec{su})$ from s should be set to a high value, so that peers are automatically encouraged to buy flows from each other and only consider the server as a last resort.

Throughout the design of our algorithms, we focused on only one media play functionality support other than normal play — random-seek. While fast-forward and rewind support are found on VCR players, their existence is highly related to the sequential access nature of video tapes. They are functionally subsumed by random-seek, and may become less relied on once random-seek is successfully implemented. Another useful control function is pause. In our system, a pause/resume can be treated as a leave followed by a re-join, with forward/backward buffers carried over.

As future directions, we plan to verify the performance of the proposed auction algorithms through empirical studies. In particular, we are interested in examining the computational and communication overhead each peer experiences in dynamic network settings. We also plan to incorporate a buffer restoring mechanism into the current algorithm design, such that a peer is allowed to bid for more than h flows, in order to build up or maintain its media buffer length. This is important in practical streaming applications, since flow switching during the auction may result in temporary low receiving rate, and therefore decreased buffer length.

In conclusion, we believe that the auction approach presents new opportunities in quality of service provisioning for P2P media flow dissemination, due to its effectiveness in cost optimization. This work is intended to exhibit such new directions to the networking community with a preliminary system design, while many further design trade-offs and practical concerns are still to be fully explored and examined.

REFERENCES

[1] C. Aggrawal, J. Wolf, and P. Yu. A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems. In *Proc. of ICMCS*, Hiroshima, Japan, June 1996.

[2] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[3] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993.

[4] J. Almeida, D. Eager, M. Ferris, and M. Vernon. Provisioning Content Distribution Networks for Streaming Media. In *Proc. of INFOCOM*, New York, USA, June 2002.

[5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proc. of SIGCOMM*, pages 205–217, Pittusburg, USA, August 2002.

[6] D. Bertsekas. The Auction Algorithm for Shortest Paths. *SIAM Journal on Optimization*, 1:425–447, 1991.

[7] D. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, Massachusetts, 1998.

[8] D. Bertsekas. A Distributed Algorithm for the Assignment Problem. Technical report, Lab for Information and Decision Systems, MIT, March 1979.

[9] D. Bertsekas and D. Castanon. The Auction Algorithm for Transportation Problems. *Annals of Operations Research*, 20:67–96, 1989.

[10] Y. Cai, K. Hua, and K. Vu. Optimized Patching Performance. In *Proc. of MMCN*, San Jose, USA, January 1999.

[11] S. Carter and D. Long. Improving Video-on-Demand Server Efficiency Through Stream Tapping. In *Proc. of ICCCN*, Las Vegas, USA, September 1997.

[12] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Content Distribution in Cooperative Environments. In *Proc. of IPTPS*, Berkeley, USA, February 2003.

[13] Y. Chawathe, S. McCanne, and E. Brewer. RMX: Reliable Multicast for Heterogeneous Networks. In *Proc. of INFOCOM*, Tel Aviv, Israel, March 2000.

[14] Y. Chu, S. Rao, and H. Zhang. A Case for End System Multicast. In *Proc. of SIGMETRICS*, Santa Clara, USA, June 2000.

[15] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1):78–88, January 2000.

[16] T. Do, K. Hua, and M. Tantaoui. P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment. In *Proc. of ICC*, June 2004.

[17] D. Eager, M. Vernon, and J. Zahorjan. Optimal and Efficient Merging Schedules for Video-on-Demand Servers. In *Proc. of MULTIMEDIA*, Orlando, USA, November 1999.

[18] D. Eager, M. Vernon, and J. Zahorjan. Minimizing Bandwidth Requirements for On-Demand Data Delivery. *IEEE Trans. on Knowledge and Data Engineering*, 13(5):742–757, September/October 2001.

[19] Y. Guo, K. Suh, J. Kurose, and D. Towsley. P2Cast: Peer-to-Peer Patching Solutions for VoD Service. In *Proc. of WWW*, Budapest, Hungary, May 2003.

[20] T. Ho, B. Leong, R. Koetter, and M. Médard. Distributed Asynchronous Algorithms for Multicast Network Coding. In *Proc. of the First Workshop on Network Coding*, Riva del Garda, Italy, April 2005.

[21] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On Randomized Network Coding. In *Proc. of Allerton*, Monticello, USA, October 2003.

[22] A. Hu. Video-on-Demand Broadcasting Protocols: A Comprehensive Study. In *Proc. of INFOCOM*, Anchorage, USA, April 2001.

[23] J. Jannotti, D. Gifford, and K. Johnson. Overcast: Reliable Multicasting with an Overlay Network. In *Proc. of OSDI*, San Diego, USA, October 2000.

[24] Z. Li and B. Li. Efficient and Distributed Computation of Maximum Multicast Rates. In *Proc. of INFOCOM*, Miami, USA, March 2005.

[25] Z. Li, B. Li, D. Jiang, and L. C. Lau. On Achieving Optimal Throughput with Network Coding. In *Proc. of INFOCOM*, Miami, USA, March 2005.

[26] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel. Scalable On-Demand Media Streaming with Packet Loss Recovery. *IEEE/ACM Trans. on Networking*, 11(2):195–209, April 2003.

[27] L. Mathy, R. Canonico, and D. Hutchison. An Overlay Tree Building Control Protocol. In *Proc. of COST264 Workshop on Networked Group Communication*, London, UK, November 2001.

[28] V. Padmanabhan, H. Wang, and P. Chou. Resilient Peer-to-Peer Streaming. In *proc. of ICNP*, Atlanta, USA, November 2003.

[29] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing Streaming Media Content using Cooperative Networking. In *Proc. of NOSSDAV*, Miami Beach, USA, May 2002.

[30] A. Schrijver. *Combinatorial Optimization: Polyhera and Efficiency*. Springer, New York City, New York, 2003.

[31] D. Tran, K. Hua, and T. Do. A Peer-to-Peer Architecture for Media Streaming. *IEEE Journal on Selected Areas in Communications*, 22(1):121–133, January 2004.

[32] M. Wang, Z. Li, and B. Li. A High Throughput Overlay Multicast Infrastructure with Network Coding. In *Proc. of IWQoS*, Passau, Germany, June 2005.

[33] D. West. *Introduction To Graph Theory, 2nd Edition*. Prentice Hall, Upper Saddle River, New Jersey, 2001.

[34] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. L. Peterson, and R. Wang. Overlay Mesh Construction Using Interleaved Spanning Trees. In *Proc. of INFOCOM*, Hong Kong, March 2004.

[35] B. Zhang, S. Jamin, and L. Zhang. Host Multicast: A Framework for Delivering Multicast To End Users. In *Proc. of INFOCOM*, New York, USA, June 2002.

[36] X. Zhang, J. Liu, and T. Yum. CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In *Proc. of INFOCOM*, Miami, USA, March 2005.