OSMOSIS: Scalable Delivery of Real-Time Streaming Media in Ad-Hoc Overlay Networks[*]

Azer Bestavros   Shudong Jin
best@cs.bu.edu   jins@cs.bu.edu
Computer Science Department
Boston University

Abstract

Ad-hoc overlay networks are increasingly used for sharing static bulk content but their promise for scaling the delivery of on-demand, real-time content *is yet to be tapped. In this paper, we show that overlay networks could be used efficiently to distribute popular real-time streaming media on-demand to a large number of clients. We propose and evaluate* OSMOSIS *a cache-and-relay end-system multicast approach, whereby a client joining a multicast session caches the stream, and if needed, relays that stream to neighboring clients which may join the multicast session at some later time.* OSCMOSIS *is fully distributed, scalable, and efficient in terms of network link costs. We present analytical and empirical results of our evaluation of* OSMOSIS*. Our analysis establishes* OSMOSIS *scalability characteristics under a variety of assumptions. Our simulations are over large, synthetic random networks, power-law degree networks, and small-world networks (all of which could well be representative of ad-hoc overlay topologies, as well as over large real router-level Internet maps.*

1 Introduction

Ad-hoc overlay networks are increasingly used for sharing static multimedia content (e.g., music and video content), whereby an end-point (client) would simply search for the content and download such content from another client, using a variety of existing protocols (e.g., gnutella) or proposed P2P architectures (e.g., CAN and Chord). A basic assumption underlying these systems is that the content to be retrieved is stored in full at one or more clients in the overlay network. As a result, the the problem of bulk content retrieval reduces to (1) finding one (or more) set of peers who have the content, and (2) downloading that content from one (or more) of these peers.

Motivation: For many applications, the assumption that the content be stored in full at one or more of the nodes in the network is not realistic. This is particularly true of applications requiring the delivery of *live* real-time content. Example applications include video on-demand, video feeds of live events, streaming sensor data in ad-hoc sensor networks, among others.

For synchronous access to real-time media, multicast solutions (whether using native network support or using end-system support) are attractive. Multicast reduces both network link costs and server bandwidth requirements for serving a large number of clients [6, 5, 24, 14, 13, 20].

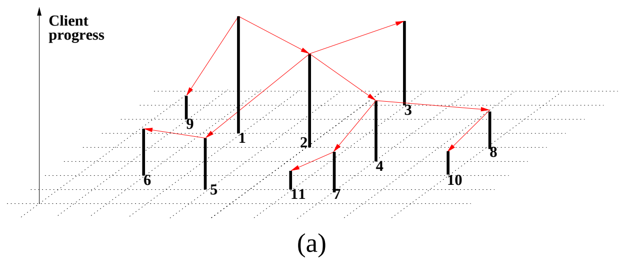
[*]This work was partially supported by NSF research grants ANI-9986397, ANI-0095988, and ANI ANI-0205294.

However, for asynchronous access, no such "attractive" solutions exist, especially using overlay networks. One of the obstacles that may have hindered the use of overlay networks in the distribution of live, real-time content is the long-held belief that the delivery of streaming media objects presents a formidable strain on server and network capacity (due to the need to store-and-forward the entire content at various peers and the need to judiciously allocate the rather miniscule resources available to all peers in the network). In this paper, we show that it is indeed possible to efficiently support *asynchronous* access to real-time content to a large number of clients in ad-hoc overlay networks, subject to realistic constraints on client bandwidth and storage capacity.

Current Techniques: To enable asynchronous access to streaming media objects, various periodic broadcasting and stream merging techniques [24, 14, 13, 20] have been proposed. Using these techniques, scalability in terms of network link cost is assured by virtue of multicast messaging, whereas scalability in terms of server bandwidth requirement is achieved by ensuring that a relatively small number of multicast sessions (possibly coupled with short unicast sessions) are enough to cater to a large number of asynchronous client requests.

Periodic broadcasting and stream merging techniques assume the availability of a network infrastructure that is supportive of multicast delivery—IP multicast, for example. While such an assumption may be practical within the boundary of a multicast-enabled intranet, it is not a viable alternative in today's Internet. This realization has led to a large body of work on application layer (or end system) approaches. However, existing end system multicast solutions [6, 5] have focused on synchronous real-time delivery—i.e., all clients receive the same content at the same time. As such, these techniques can not be used to service asynchronous clients.

Paper Contributions and Overview: In this paper, we propose and evaluate OSMOSIS—a scalable "cache-and-relay" end system multicast protocol for the asynchronous delivery of streaming media objects. Unlike existing periodic broadcasting and stream merging techniques, our protocol relies only on unicast messaging, and unlike existing end system multicast techniques, our protocol supports asynchronous delivery. Using our approach, upon joining an ongoing end system multicast session, a client caches the stream either partially or entirely, and if needed, relays that stream to neighboring clients which join the multicast session at some later time. The paper mainly analyzes the network link cost of this approach, studies



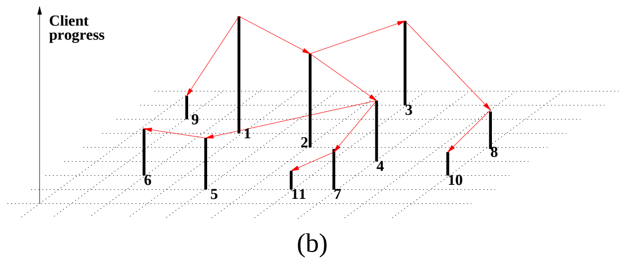


Figure 1. Illustrations of the OSMOSIS "cache-and-relay" approach. Earlier clients temporarily keep the objects and relay them to later clients. (a) Scenario with unconstrained clients. (b) Scenario with bandwidth-constrained clients, whereby client 2 is limited to receive and send at most 3 streams.

the effect of limited client-side bandwidth and limited client cache capacity, and uses simulations to validate our findings.

2 The OSMOSIS Cache-and-Relay Protocol

Using OSMOSIS, end-hosts are responsible for the caching and distribution of streaming media. Here, end-hosts can be client machines or proxies thereof. End-hosts keep retrieved media objects in their local caches temporarily, as the results of client requests. If another client requests the media objects later, the original server can redirect the request to those end hosts who are geographically closer to the client.

We illustrate the cache-and-relay approach of OSMOSIS using the example in Figure 1(a). In that example, there are 11 clients requesting an object. These clients are placed on a two-dimensional grid to visualize network "distance" between these clients.[1] Clients arrive at different times. In Figure 1(a), each client is marked with a number denoting the order of its arrival. Also, the value of the z-axis for a given client indicates the progress of the playout for that client. The figure shows how an object is forwarded from "earlier clients" to "later clients". Clearly, such an approach assumes that clients (or client-side caching proxies) have enough cache space to temporarily keep the received media objects either partially or entirely. Also, it assumes that a non-leaf client, while receiving and playing out an object, has additional bandwidth to forward that object to (one or more) neighboring clients who may arrive later.

Without loss of generality,[2] we assume that the objective of our OSMOSIS cache-and-relay approach is to minimize the total network link cost, or hop-distance. It is not difficult to establish that the cache-and-relay solution shown in Figure 1(a) is optimal when client-side bandwidth and client cache capacity are unlimited. Each client receives the object from the nearest on-going peer. The total hop-distance is 28.

[1]Distance could be measured in physical terms (e.g., number of feet), which may be meaningful for wireless communication as it reflects power consumption, or it could be measured in topological terms (e.g., number of hops).

[2]Specifically, our discussion and results can be easily adapted to allow for the minimization of other metrics such as delay, packet loss rates, etc.

3 OSMOSIS: Scalability and Instantiations

In this section, we first show how effectively OSMOSIS could reduce network link cost, given unlimited client-side bandwidth and cache space. Then we formalize the problem when either client-side bandwidth or cache size is limited. In each case, a specific instantiation of OSMOSIS is proposed.

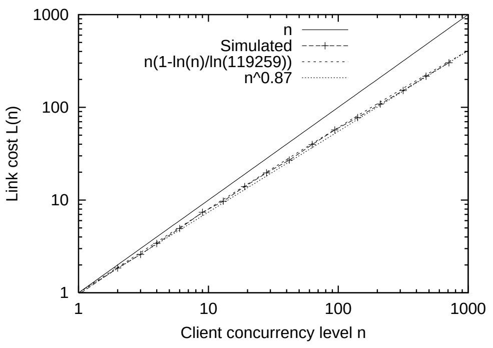
Network Link Cost: Assuming unlimited client-side bandwidth and cache capacity, a new client can always fetch the object from the nearest ongoing peer client—a peer that started receiving that object but have not finished. We define the cost of serving the new client to be the hop-distance between that client and the nearest ongoing peer. Let $L(n)$ denote the total network link cost for n consecutive client arrivals within a unit time, whereby each client fetches the object from the nearest ongoing peer. Here, a unit time is defined as the duration of the media object, and hence n is the average client concurrency level. $L(n)$ reflects how OSMOSIS scales (in terms of network link cost) as the level of client concurrency n increases.

In random networks, which have exponential neighborhood expansion functions, we have computed the following asymptotic scaling behavior (see [16] for a detailed derivation):

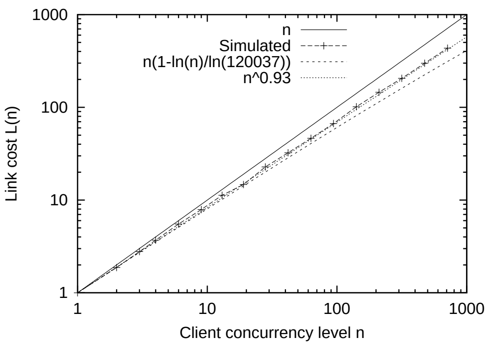
$$L(n) \sim n\left(1 - \frac{\ln n}{\ln N}\right), \tag{1}$$

where N is the total number of nodes in the network. This result implies that the increase in network link cost is a sub-linear function of the client arrival rate n. This underscores a clear reduction in network link cost compared to unicast service whose cost is linear in n.

The key to the derivation of $L(n)$ is the neighborhood expansion function $E(d)$ of the network, which is defined as the average fraction of vertices reachable in d hops, starting from an arbitrary vertex. In random networks, this function is approximated by an exponential function. While the derivation of $L(n)$ for an arbitrary network is impossible, we have also considered networks whose neighborhood expansion functions follow a power-law. As detailed in [16], we found that if the neighborhood expansion function is a power-law with exponent



(a) Random network



(b) Power-law network

Figure 2. Comparisons of theoretic network link cost and simulation results using synthetic networks.

H, then $L(n)$ increases asymptotically as

$$L(n) \sim n^{1-\frac{1}{H}} \tag{2}$$

Example networks with power-law neighborhood expansion function include two-dimensional and three-dimensional grids.

Handling Limited Client Bandwidth: In practice, clients may have limited bandwidth to receive and send streams. Therefore, it may be infeasible for a client to receive a stream from the nearest on-going peer. For example, in Figure 1(a), if we assume that each client can receive and send at most three streams, then the solution shown in the figure becomes infeasible since client 2 cannot receive and send four streams in total. In Figure 1(b), a feasible solution is shown.

It is difficult to find the optimal solution when bandwidth is limited. Indeed, even the off-line algorithm (with prior knowledge of client arrivals) is NP-hard. To explain why this is the case, it suffices to note that the construction of a degree-constrained spanning tree[3] is an NP-complete problem [12]. By restricting our problem to synchronous clients and integer bandwidth values, finding an optimal cache-and-relay solution is equivalent to finding a solution to the degree-constrained spanning tree problem.

A simple greedy solution for the bandwidth-constrained cache-and-relay problem works as follows. Each new client receives the object from the nearest ongoing peer client who still has abundant bandwidth. The solution in Figure 1(b) is obtained using this greedy algorithm. Client 5 receives a stream from client 4 and client 8 receives data from client 3. The total hop-distance is 32. Our simulation results suggest that this greedy algorithm usually finds good solutions.

Handling Limited Cache Capacity: In practice, clients may have limited cache capacity. For example, in a caching proxy, it might be unrealistic to cache a whole video whose size is up to Giga bytes, especially when many such media objects may be competing for cache space.

[3]Given a graph $G = (V, E)$ and a positive integer $K \leq |V|$, find a spanning tree for G in which no vertex has degree larger than K.

When cache space is limited, the solutions in Figures 1(a) and 1(b) may become infeasible. For example, if client 1 has a cache capacity that enables it to keep only 50% of the object, then when client 9 arrives, it is already too late for that client to fetch the object from client 1. Instead, a feasible solution is for client 6 to relay the object to client 9.

To handle cache capacity constraints, it is necessary to determine a *cache replacement* policy. It is straightforward to use a FIFO policy: a sliding window indicates the current segment in the cache. The client can relay the object to other clients who start slightly later (i.e., within that window). In the next section, we show how constraints on cache capacity impact the reduction of network link cost.

4 Performance Evaluation

In this section, we validate through simulation the network link cost of OSMOSIS, and study the effect of limited client-side bandwidth and cache capacity. Large, synthetic and real networks are used in simulation. Experimental results are compared to theoretical bounds given earlier.

Networks Used in Our Simulations: In our simulations, four synthetic and real networks were used. All four networks have approximately 110000-120000 nodes and have an average degree of 3.2. The topologies we consider are:

- A random network generated using the ER model [10]. In this model, there is a uniform probability of having an edge between any pair of vertices in the graph. The model does not guarantee that the network is connected. So, we use the largest connected component with 119,259 vertices.
- A random power-law degree network with 120,037 vertices generated using the model in [1]—namely, the probability of having node degree larger than d is proportional to $d^{-\alpha}$ (we set $\alpha = 2.5$).
- A small-world network with power-law degree distribution, generated using the model in [18]. The network has 120,000 vertices. The resulting topology is different from random power-law degree networks as it features a large clustering coefficient. In generating this network, we not

only used power-law vertex degree with $\alpha = 2.5$, but also considered the physical distance of the vertices in creating edges.

- A router-level Internet map (Lucent) available from [23]. This map has 112,269 vertices and it less strictly follows a power-law degree distribution. In addition, it has a high clustering coefficient [18]. We have found that our small-world network is the closest to this real Internet in terms of average path length and clustering coefficient.

Network Link Cost Validation: Our simulation proceeds as follows. Client arrivals are Poisson, with each client residing at a random node of the simulated network. We vary the client arrival rate (or concurrency level n) and obtain the corresponding network link cost $L(n)$ scaling as a function of n.

We first assume unlimited client bandwidth and cache capacity in validating the network link cost presented in the last section. Here only the results using the random network and using the power-law random network are presented.

Figure 2(a) shows that when the random network is used, the network link cost is well-predicted by Equation (1). In addition, it appears that Equation (2) also provides a good fit. This is explained by our discussion in the last section, i.e., in limited scales, this two equations are close. Figure 2(b) shows that the network link cost is clearly higher than that predicted by Equation (1). Notice the log-scale of $L(n)$ in the figure. This is because power-law random networks do not have an exponential neighborhood expansion function.

Effect of Limited Client Bandwidth: Figure 3 shows the resulting scaling behavior when the client-side bandwidth is chosen in different ways. Also, for comparison purposes, it shows the cost of unicast delivery.

When client-side bandwidth is chosen uniformly between object playback rate and four times that rate, the network link cost is not significantly higher than what is achievable with infinite bandwidth. This result suggests that our approach is effective even when client-side bandwidth is low.

Again, the simulation results using power-law networks appear to be rather different from those obtained using router-level Internet maps. In the power-law network, the network link cost reduction is less than that in router-level maps. However, we found that the simulation results using a small-world, power-law network is close to that obtained using router-level Internet maps. This underscores the importance of capturing small-world behaviors in Internet topologies–namely *clustering in networks is important to the scaling behavior of multicast delivery* [18].

Effect of Limited Client Cache Capacity: Figure 4 shows the resulting scaling behavior when the client cache capacity is constrained. In our simulations, we choose different cache capacities, corresponding to 10%, 30%, and 100% of the object size. Buffer management uses a FIFO replacement policy.

The results in Figure 4 indicate that: (1) Even when cache space is limited, the reduction in network link cost is still significant compared to that of unicast delivery. (2) There is still room for improvement when cache capacity is small. Notice that we use a simple FIFO policy which can be less efficient than others. In addition, it is also possible to combine prefetching techniques to better utilize limited cache spacee. For example, assume the client cache can only store a S-minute segment of the object. When a later client starts, it may prefetches the object from any client who started less than $2S$ minutes earlier. Therefore, it works as if the cache size is doubled.

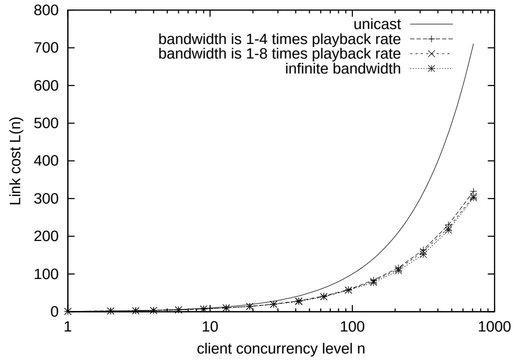
5 Related Work

End-system multicast was advanced by the authors of [6] as a deployable alternative to IP multicast. In their Narada protocol, end systems self-organize into an overlay network using a fully distributed protocol, with fairly low delay and bandwidth overheads. Recently, they conducted an extensive evaluation of schemes for constructing overlay networks on a wide-area testbed [5]. This study demonstrated that end system multicast is promising for conferencing applications in a dynamic and heterogeneous Internet environment, and highlighted the importance of adapting to latency and bandwidth while constructing overlays optimized for the real-time delivery of content to synchronous clients.

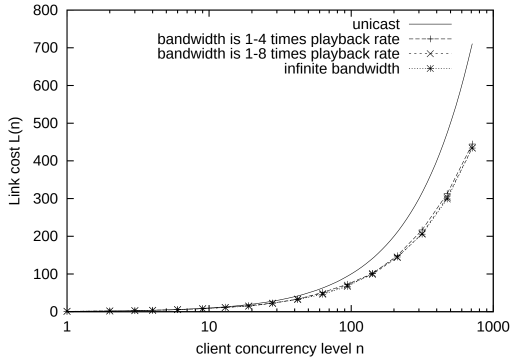
Delivery of content to asynchronous clients is the focus of many recent studies, including periodic broadcasting [24, 14, 13, 20] and stream patching/merging techniques [4, 11, 7, 8]. These approaches are targeted mainly at video-on-demand applications. In periodic broadcasting techniques, segments of an object (with increasing sizes) are repeatedly transmitted on dedicated channels, and asynchronous clients simply join one or more broadcasting channels to receive this data. Using stream patching/merging techniques, asynchronous clients are merged into larger and larger groups that share a single multicast channel. Unlike our approach, both techniques assume the availability of a lower-level multicast delivery infrastructure. As such, they are scalable in minimizing server bandwidth requirement [9, 17], but do not specifically attempt to optimize for network link cost.

The idea of utilizing client-side cache space was developed in a number of recent works [22, 21]. Here, the main objective was to reduce server load rather than network link costs. A network level scheme was presented in [15], which caches data at routers in the network to service subsequent requests. It is therefore different from our application layer approach. In addition, it also aimed at lightening the demand on the server bandwidth.

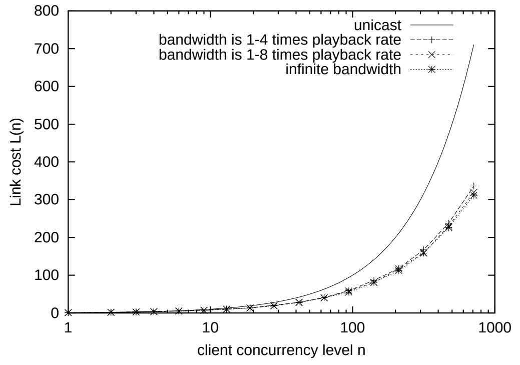
Another class of content delivery techniques originated with the use of periodic broadcasting of encoded content as was done over broadcast disks [2], and as was done later through the Digital Fountain approach [3]. These approaches enable end-hosts to efficiently reconstruct the original content of size n from a subset of any n symbols from a large universe of encoded symbols. Such approaches enable reliability and a substantial degree of application layer flexibility. The primary weakness of these techniques is their inability to efficiently deal with real-time (live or near-live) streaming media objects due



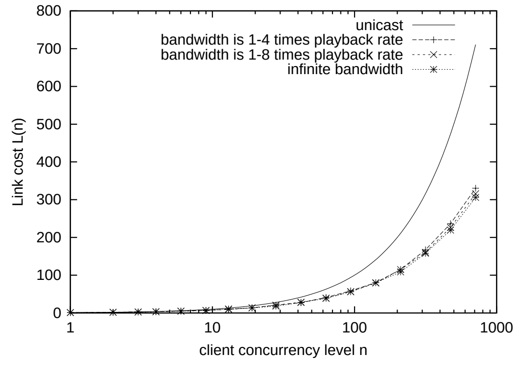
(a) Random network



(b) Power-law network

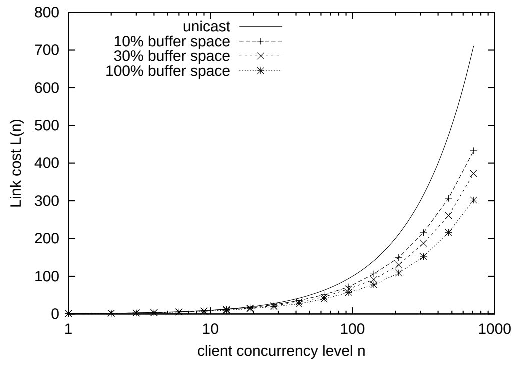


(c) Small-world power-law network

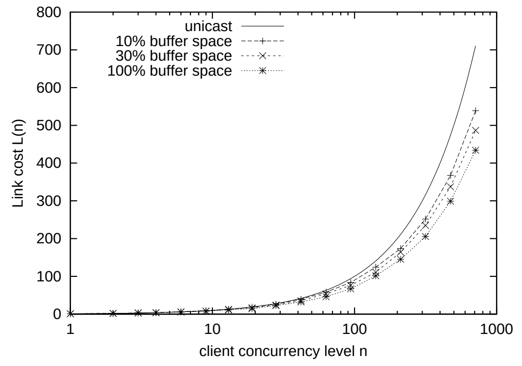


(d) Router-level Internet map

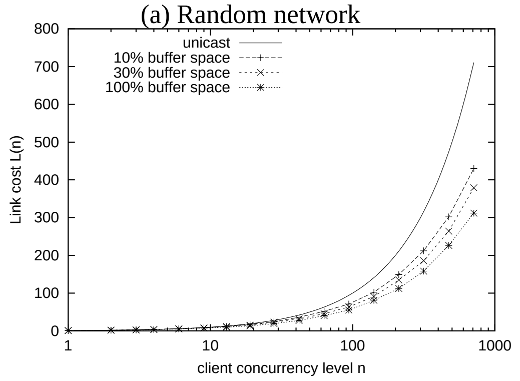
Figure 3. Simulation results when client-side bandwidth is limited.



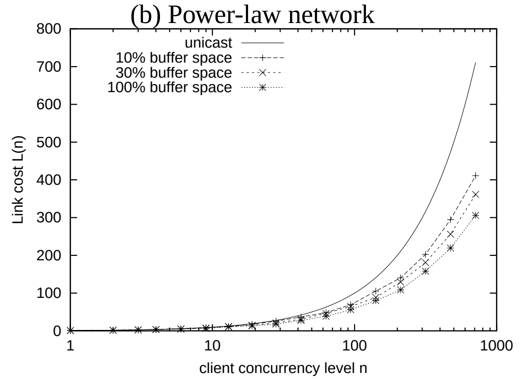
(a) Random network



(b) Power-law network



(c) Small-world power-law network



(d) Router-level Internet map

Figure 4. Simulation results when client-side cache space is limited.

to the necessity of encoding/decoding rather large stored data segments.

6 Conclusion

We proposed and evaluated OSMOSIS—a cache-and-relay application layer multicast delivery mechanism for streaming media objects. Delivery by OSMOSIS minimizes the total network link cost and is especially tailored for applications featuring asynchronous client requests to live, real-time streaming media objects. We are currently investigating several issues that need to be addressed in order for OSMOSIS to be deployable in ad-hoc overlay networks. We briefly discuss these below.

First, one requirement the cache-and-relay approach used in OSMOSIS is the need for an effective discovery mechanism for close-by peers who can satisfy a request—that is, how to find the closest client with a cached copy of the requested stream and with sufficient bandwidth to serve that stream. One simple peer discovery mechanism works as follows. The root server maintains a set of addresses of on-going clients. When a new client requests the media object, the server provides a subset of candidates (from its list of on-going clients) based on efficient clustering techniques, for example, the one in [19]. The new client may then choose one of these candidtaes based on measurements of the characteristics of its paths to those candidates.

Second, another concern regarding OSMOSIS is reliability. For example, if one client is relaying some media object to a another client, then if the first client dies, the latter needs to figure out how and from where to receive the remainder of the object. One solution for this vulnerability is for the latter client to contact the original server, and establish a connection with another client. In the worst case, the client may have to download the remainder of the object from the original server. To minimize the implications of this "switch-over" on real-time playout, clients may wish to actively maintain a list of alternate sources, and to factor in the delay of a switch-over into their buffering requirements.

Finally, security is a common concern of application layer approaches, including our proposed cache-and-relay protocol. Since clients can access the caches at other end-hosts, the system must prevent unauthorized accesses, for example access without digital rights. Security support can be implemented by either the original server (alone or assisted by trusted clients).

References

[1] W. Aiello, F. R. K. Chung, and L. Lu. A random graph model for massive graphs. In *ACM Symposium on Theory of Computing (STOC)*, pages 171–180, 2000.

[2] A. Bestavros. AIDA-based real-time fault-tolerant broadcast disks. In *Proceedings of IEEE RTAS'96*, Boston, Massachusetts, May 1996.

[3] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of ACM SIGCOMM*, 1998.

[4] S. W. Carter and D. D. E. Long. Improving video-on-demand server efficiency through stream tapping. In *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN)*, September 1997.

[5] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *Proceedings of ACM SIGCOMM*, August 2001.

[6] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, June 2000.

[7] D. Eager, M. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. In *Proceedings of Workshop on Multimedia Information Systems (MIS)*, 1998.

[8] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth skimming: A technique for cost-efficient video-on-demand. In *Proceedings of S&T/SPIE Conference on Multimedia Computing and Networking (MMCN)*, January 2000.

[9] D. Eager, M. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Transactions on Data and Knowledge Engineering*, 13, 2001.

[10] P. Erdős and A. Rényi. The evolution of random graphs. *Pupl. Math. Inst. Hungar. Acad. Sci.*, 7:17–61, 1960.

[11] L. Gao and D. Towsley. Supplying instantaneous video-on-demand services using controlled multicast. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, June 1999.

[12] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[13] A. Hu. Video-on-demand broadcasting protocols: A comprehensize study. In *Proceedings of IEEE INFOCOM*, April 2001.

[14] K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proceedings of ACM SIGCOMM*, September 1997.

[15] K. A. Hua, D. A. Tran, and R. Villafane. Caching multicast protocol for on-demand video delivery. In *Proceedings of S&T/SPIE Conference on Multimedia Computing and Networking (MMCN)*, 2000.

[16] S. Jin and A. Bestavros. Cache and Relay Streaming Media Delivery for Asynchronous Clients. In *Proceedings of the 4th International Workshop on Networked Group Communication*, Boston, MA, October 2002.

[17] S. Jin and A. Bestavros. Scalability of multicast delivery for non-sequential streaming access. In *Proceedings of ACM SIGMETRICS*, June 2002.

[18] S. Jin and A. Bestavros. Small-world internet topologies: Possible causes and implications on scalability of end-system multicast. Technical Report BUCS-2002-004, Boston University, 2002.

[19] B. Krishnamurthy and J. Wang. On network-aware clustering of web clients. In *Proceedings of ACM SIGCOMM*, August 2000.

[20] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel. Scalable on-demand media streaming with packet loss recovery. In *Proceedings of ACM SIGCOMM*, August 2001.

[21] S. Ramesh, I. Rhee, and K. Guo. Multicast with cache(mcache): An adaptive zero-delay video-on-demand service. In *Proceedings of IEEE INFOCOM*, April 2001.

[22] S. Sheu, K. Hua, and W. Tavanapong. Chaining: A generalized batching technique for video on demand. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, 1997.

[23] USC Information Sciences Institute. Internet maps. http://www.isi.edu/div7/scan/mercator/maps.html.

[24] S. Viswanathan and T. Imielinski. Pyramid broadcasting for video on demand service. In *Proceedings of S&T/SPIE Conference on Multimedia Computing and Networking (MMCN)*, 1995.