

A Peer-to-Peer On-Demand Streaming Service and Its Performance Evaluation¹

Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley
Department of Computer Science
University of Massachusetts, Amherst, MA 01003
yguo, kwsuh, kurose, towsley@cs.umass.edu

Abstract—Providing on-demand video streaming service over the Internet is a challenging task. In this paper, we propose DirectStream, a directory based peer-to-peer video streaming service that efficiently and cost-effectively provides video on-demand service with VCR operation support. We analytically and experimentally examine the system performance, and show that the proposed scheme can significantly reduce the workload posed on the server, and that it scales extremely well as the popularity of the video increases even if participating clients behave non-cooperatively. We propose a QoS parent selection algorithm to construct the appropriate peer-to-peer networks, and discuss how to provide continuous playback in the face of clients' early departures. Our study suggests that peer-to-peer networking is a promising technique to address scalability in on-demand streaming service.

I. INTRODUCTION

Providing on-demand streaming service in a scalable way is a challenging task due to the long durations and high bandwidth requirements of multimedia streams. In the past, IP multicast based approaches and replicated server/proxy based approaches have been developed to tackle the scalability issue. However, all these methods have their own drawbacks [1], due to limited deployment and access to IP multicast, or to costs associated with deployment and maintenance. An on-demand streaming service that is scalable and flexible while incurring low deployment and maintenance cost is desirable.

In this paper, we propose *DirectStream*, an on-demand streaming service. DirectStream is a directory-based peer-to-peer video streaming system that can provide instantaneous or near-instantaneous video on-demand service (VoD) and can enable basic VCR functionality. In DirectStream, a directory server works as a central administrative point that facilitates the construction of peer-to-peer networks formed among the server and clients. Each client is implemented with both server and client capabilities, and behaves like a peer - it caches a moving window of the latest received content, and serves latecomers by continuously forwarding the cached content. We show, through analysis and simulation, that the proposed scheme can significantly reduce the workload posed on the server, and that it scales extremely well as the popularity of the video increases.

We investigate several unique issues introduced by peer-to-peer on-demand streaming, namely constructing the peer-to-peer overlay appropriate for streaming, providing continuous stream playback in the face of disruption from early departing clients, and combating non-cooperative peer clients. We propose a QoS parent selection al-

gorithm to construct the appropriate overlay, and discuss the means to provide uninterrupted playback in the face of clients' early departures. Our study shows that DirectStream performs well even when a fraction of the participating clients behave selfishly or non-cooperatively.

The remainder of the paper is organized as follows. In Section II we describe the design and architecture of DirectStream. The performance of DirectStream is evaluated both analytically and through simulation in Section III. We discuss the means to provide continuous playback in the presence of clients' early departures in Section IV. Finally, related work, concluding remarks, and future work are included in Section V.

II. DIRECTSTREAM ON-DEMAND STREAMING SERVICE

DirectStream comprises a directory server, a content server (or content servers), and clients. The directory server works as a central administrative point. It maintains a database that keeps track of all servers and clients participating in DirectStream, and helps new clients to locate the required service. The content servers provide the same functionality as in the traditional client-server service model - storing contents in their repository and serving clients' requests so long as sufficient bandwidth is available. In contrast, the clients in DirectStream are implemented by providing both server and client capabilities. Clients cache a moving window of video, and can serve other clients by forwarding the stream.

Fig. 1 illustrates a DirectStream system with one directory server, one content server, and a number of clients at time t . We use solid line circles to represent active clients that are receiving and/or forwarding the stream; and dashed line circles to represent departed clients. We define *cluster* to be a set of active clients among which a peer-to-peer streaming overlay is established. The clients in a cluster share the same stream. For instance, clients B and C form a cluster. The streaming overlay, in this case a forwarding chain, is established.

Clusters in DirectStream evolve over time. For instance, client A was a member of the cluster that contains clients B and C , however it left the cluster after retrieving the entire stream from the server and finishing the playback. This cluster can also grow by admitting new clients into it.

- **Data caching at clients.** Clients in DirectStream cache a moving window of the most recent content that they have received. Assume a client can buffer b minutes worth of video (we assume that the playback rate is constant), and is watching the video at the position τ minutes from the beginning of the video at time t . It caches the most recent b minutes of the video, $[(\tau - b)^+, \tau]$, and continuously caches the most recent content as time goes along. This client can serve any client requesting the same video starting at the position within $[(\tau - b)^+, \tau]$.

- **The directory server.** The directory server maintains information about the content server and clients in order to facilitate the

¹This research was supported in part by the National Science Foundation under NSF grants EIA-0080119, ANI-0085848, ANI-9973092, ANI9977635, and CDA-9502639. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

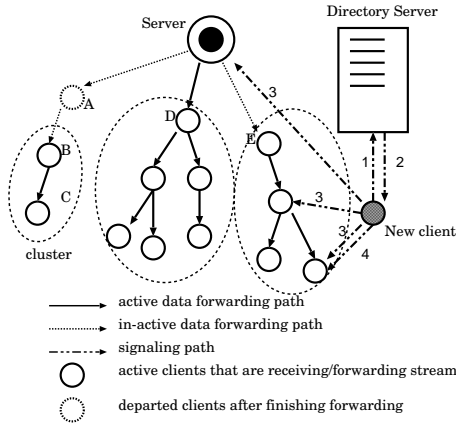


Fig. 1. DirectStream Architecture.

search for candidate parent clients for new requests. In the case of the content server, the directory server keeps track of its IP address. For clients, the directory server maintains one entry for each client. The entry for the client i is a tuple (a_i, t_i, τ_i, b_i) , where a_i is client i 's IP address, t_i is the time when this client started to receive the stream, τ_i is the position in the video that the client began at, and b_i is the client's buffer size. At any future time t , the content cached at client i is $\left[\min\{L - b_i, \tau_i + (t - t_i - b_i)^+\}, \min\{L, \tau_i + (t - t_i)\} \right]$, where L is the video length.

- **Serving a new request.** The service search process for a new request consists of the following four steps, as indicated in Fig. 1:
 - Step 1.* The new client sends a request to the directory server to ask for the video starting at position τ .
 - Step 2.* The directory server looks into its database and returns a list of candidate nodes, including both the content server and clients that have the content to serve this request.
 - Step 3.* The new client determines from which node to retrieve the stream using the QoS parent selection algorithm.
 - Step 4.* The new client contacts the selected candidate node and asks to forward the stream. After the connection is successfully set up, the new client signals back to the directory server and registers itself into the database.

A similar procedure can be used to provide VCR functionalities in DirectStream [1]. We omit its description here due to space limitation.

- **QoS parent selection algorithm.** The peer clients in DirectStream form peer-to-peer overlays over which the video stream is forwarded. Since a minimum guaranteed bandwidth equal to the playback rate is required over all connections in the overlay networks, a client has to select a parent node that has sufficient bandwidth to itself. Moreover, the selection of the parent node should allow the DirectStream to admit as many clients as possible in the long run.

Assume that client A sends a service request to the directory server and the directory server returns a list of n candidates, $\{c_i\}_{i=1}^n$, that can provide the service. Suppose the number of hops from candidate client c_i to the requesting node is n_i , and the measured available bandwidth is x_i . The QoS parent selection algorithm uses the *distance-bandwidth ratio*, defined as n_i^r/x_i where $0 \leq r < \infty$, as the metric in selecting the parent peer client. Intuitively, the selection of a parent with large available bandwidth helps to balance the workload; while that with short network distance reduces the traffic placed on the underlying network. QoS

parent selection tries to properly balance the above two factors. In [1] we show that the QoS selection algorithm performs better than other traditional selection algorithms, such as shortest-widest selection algorithm and widest-shortest selection algorithm [2].

III. PERFORMANCE EVALUATION OF DIRECTSTREAM

In this section, we examine the performance of DirectStream both analytically and experimentally. We pay particular attention to the occasions where clients behave non-cooperatively. We derive a closed form formula on average workload placed at the content server, and conduct extensive simulation experiments to further evaluate the performance of DirectStream in both cooperative and non-cooperative environments. The results show that DirectStream can significantly reduce the workload posed on the server, and scales extremely well as the popularity of the video increases even if the participating clients behave non-cooperatively.

- **Average server stress.** In DirectStream, the content server only needs to stream a single copy to the first client in a cluster. All other clients in the same cluster obtain the content from the earlier arrived peer clients. In the following analysis, we assume that if a client can receive the content from a client, it will do so rather than obtain the video from the server. In addition, we assume that the request arrival process is Poisson with arrival rate λ , and all clients have the same size buffer, denoted by b .

We denote the average workload placed at the content server, S , as the server stress. Let L be the video length, W be the normalized workload, $W = \lambda L$, and ρ be the effective buffer size, $\rho = b/L$. We have the following Proposition (see [1] for its proof).

Proposition III.1: Consider DirectStream with a single content server. If the arrival process is Poisson, then

$$E[S] = W e^{-\rho W}. \quad (1)$$

Furthermore, on average, each client serves $1 - e^{-\rho W}$ clients.

Remark: Average server stress reaches a maximum of $(e\rho)^{-1}$ when $W = 1/\rho$. For instance, if $\rho > 0.05$, the average server stress will not exceed 8 streams. Figure 2 plots the average server stress for different values of ρ , and compares them with a lower bound for all IP-multicast based schemes, which is $\ln(1 + W)$ as derived in [3]. We observe that the average server stress in the DirectStream decreases as the normalized workload increases once $W \geq 1/\rho$, while the lower bound for an IP-multicast based scheme confines to increase logarithmically.

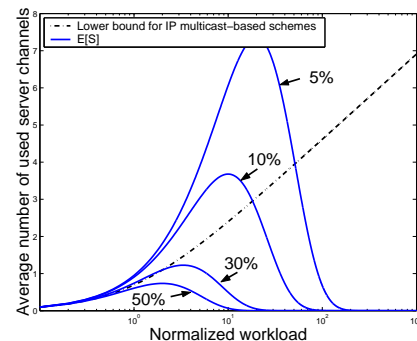


Fig. 2. Average server stress vs. normalized workload with the buffer size of 5%, 10%, 30%, and 50% of video length

Now we study the effect of clients that do not behave cooperatively. Clients that receive the stream and refuse to serve other clients

are denoted as *freeloaders*. Assume that a client is a freeloader with probability p , we have the following results.

Corollary III.1: Consider a DirectStream system with a single server. The arrival process is Poisson with rate λ . Suppose a client is a freeloader with probability p , and a normal client otherwise. Then:

$$E[S] = W e^{(1-p)\rho W}. \quad (2)$$

Furthermore, on average, each non-freeloader client has to support $\frac{1 - e^{-(1-p)\rho W}}{1-p}$ clients.

Remark: First, note that freeloaders decrease the effectiveness of data buffering. For instance, DirectStream with no freeloaders in which 5% of the video can be stored into buffer is equivalent to DirectStream in which 50% of the clients are freeloaders and the buffer can store 10% of the video. Second, the maximum stress is $1/(e\rho(1-p))$ when $W = 1/(1-p)\rho$. Finally, each non-freeloader has to serve more clients with freeloaders than without freeloaders since $(1 - e^{-(1-p)\rho W})/(1-p) \geq 1 - e^{-\rho W}$. This can cause the concentration of network traffic and thus reduce the system's scalability (we show this using experiments in [1]).

A. Simulation experiments

- **Simulation settings.** We evaluate DirectStream using a network topology generated by GT-ITM [4] with 100 nodes. It consists of one transit network (with 4 nodes) and 12 stub domains. We assume that each node represents a local network that can host an unlimited number of clients, and that there is sufficient bandwidth within a local network to support media streaming. Link capacities are assigned to be integer multiples of the playback rate. We choose the capacity of each core link (between transit nodes or between transit nodes and stub domain nodes) to be 10, i.e., core links can support up to 10 streams simultaneously; and that of each edge link to be 3 for the simulation results reported in this paper. The video length is set to 100 mins, and the parameter r in the QoS selection algorithm to 0.5.

We simulate the on-demand service of one video to clients whose arrival process is Poisson. Each client is equally likely to be placed at any node in the network. We place the server at one of the stub nodes. In our simulation results, the half-width of the 95% confidence interval of the data shown in this paper is always less than 5% of the point estimate.

- **Client rejection probability.** Fig. 3(a) depicts the rejection probability vs. the normalized workload, W , for unicast (client-server service model) and DirectStream with different buffer sizes, ranging from 5% to 20% of the video length. Compared to the traditional client-server service model, DirectStream significantly serves more clients, especially when the client request rate is high. Furthermore, we observe that the rejection probability for DirectStream exhibits an interesting bimodal behavior, with one mode in a low workload region and the other in a high workload region.

We argue that the first mode is due to limited bandwidth at the server, while the second one is caused by a combination of limited bandwidth at the server and limited capacity of the entire network. We denote by *cluster headers* the clients whose candidate list returned from the directory server only contains the content server. Fig. 3(b) shows the rejection probability caused by the cluster headers that intend to obtain the video from the server directly. Comparing the two graphs in Fig. 3, one observes that the rejection of the cluster headers by the server mainly contributes to the first mode. Proposition III.1 also indicates that when the load equals $1/\rho$, the server bandwidth requirement reaches the maximum, which coincides with the maximum rejection probability in Fig. 3(b).

Regarding the second mode, as the client request rate increases, the server stress (Fig. 4(a)) and network usage (Fig. 4(b)) also in-

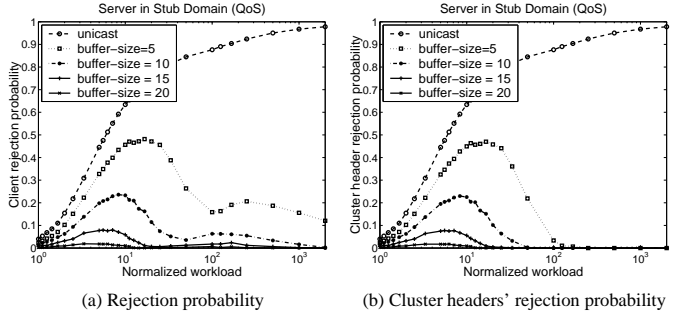


Fig. 3. Overall rejection probability and cluster header rejection probability in DirectStream

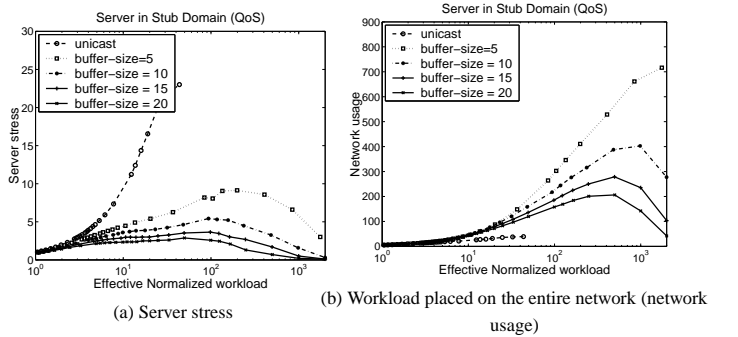


Fig. 4. Server stress and network usage in DirectStream

crease. Thus client requests are more likely to be rejected due to limited available bandwidth at the server as well as between clients. However, as the workload further increases, the likelihood that there is a peer client in the same local network increases. Hence the rejection probability decreases as the arrival rate further increases.

- **Effect of freeloaders.** Here we investigate the effect of freeloaders on the performance of DirectStream. As depicted in Fig. 5, the rejection probability and the server stress increase as the freeloader probability increases. However, even when the freeloader probability is high, the rejection probability exhibits a similar trend as it does with lower freeloader probability, consistent with Corollary III.1.

- **Effect of greedy clients.** Another type of non-cooperative

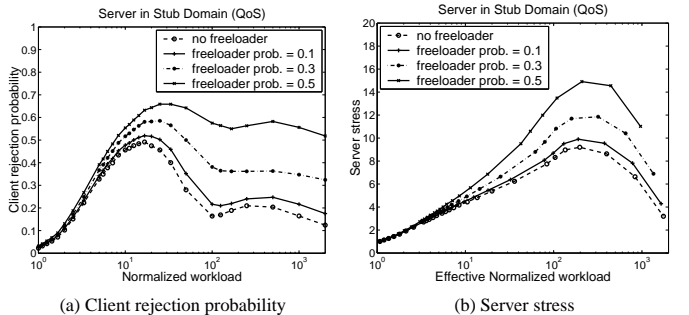


Fig. 5. Performance comparison with freeloaders

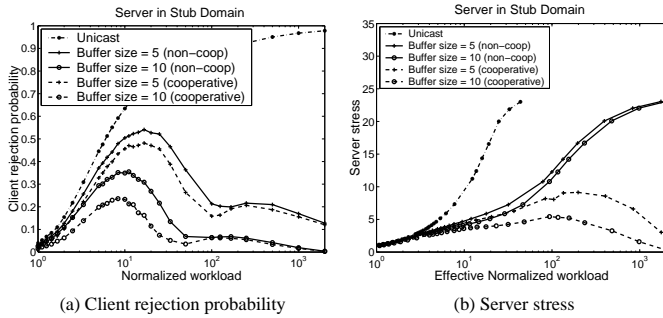


Fig. 6. Performance comparison with greedy clients.

client is a *greedy client*. A greedy client connects to the server directly whenever possible since the server cannot depart in the middle of streaming and thus does not impose the client early departure problem, as defined in Section IV. Fig. 6(a) compares the rejection probability for a system with cooperative clients to that for a system with greedy clients. Although the rejection probability for a system with greedy clients is higher than that with cooperative clients, it still exhibits a similar shape, and scales well as the request rate increases. Intuitively, the server can be modeled as a $M/D/C/C$ queue, where the service time is the video length, and C is the number of channels available at the server. In a cooperative environment, each server channel is used to serve a cluster; while in the non-cooperative environment, the greedy clients try to grab an idle channel whenever possible. However, the greedy clients are forced to form a cluster when all channels are occupied. This explains why the rejection probability curves exhibit the similar shapes. As to the server stress (see Fig. 6(b)), DirectStream with greedy clients imposes a much greater workload on the server, which does not decrease even when the request rate is high due to the greediness of clients.

IV. PROVIDING CONTINUOUS PLAYBACK

As in any P2P application, a participant can leave at any time without advance notice. We denote this as the *early client departure*. Early client departure can disconnect the peer-to-peer overlay, hence the reconstruction of overlay is usually required. In DirectStream, we let the direct children of departing client contact the directory server and find the new parent client. During this process, the downstream clients' continuous playback can be disrupted. Below we examine the effect of early client departure in DirectStream, and propose a simple method to deal with such disruptions. Because clusters are independent of each other in DirectStream, early client departures in one cluster will not affect the clients in other clusters. We focus on a single cluster.

One effective way to deal with this is to buffer a short period of data and delay the start of playback, say for d time units, so that the client can locate an alternative parent client before it uses up the buffered data. We believe that the length of d should be at the order of 10s of seconds. If a client's ancestor clients depart early, the above technique is also effective since the buffered data at the intermediary clients can help to shield the disruption.

V. RELATED WORK, CONCLUSION AND FUTURE WORK

There have been some efforts, both in academia and industry, to address the scalability issue presented in the streaming media service using peer-to-peer networking techniques [5–9]. Data buffering at clients corresponds to interval caching that was first proposed in [10, 11] to efficiently utilize memory for video streaming. It has been ap-

plied at the application level in several occasions ([8, 9, 12]). To our best knowledge, we are the first to propose a framework that allow clients to take full advantage of the benefits of interval caching, and carefully examine the system's performance in terms of server stress, network usage, and the effect of clients' non-cooperative behavior unique to the peer-to-peer networks. Furthermore, we investigate how to appropriately construct the peer-to-peer networks in order to improve the system's scalability, and provide continuous playback in face of clients' early departures.

In this paper we propose DirectStream streaming media service that can efficiently and cost-effectively provide video on-demand service with VCR operation support. We show, through both analysis and simulation experiments, that DirectStream significantly decreases the workload posed on the server as well as on the network, and scales much better than the traditional client-server unicast service model even when the participating clients behave non-cooperatively. We investigate QoS parent selection algorithms in constructing the streaming overlay so as to maximize the system's scalability. We also investigate the means to provide uninterrupted playback at client side in the face of clients' early departures.

Our preliminary results suggests that peer-to-peer networking is a promising technique to address the scalability issue in VoD service. It also bring up several interesting technical challenges, such as peer client early departure problem, overlay construction problem, and non-cooperative peer problem. In this paper we show that our design can solve some of these problems. However further research is required.

VI. ACKNOWLEDGEMENTS

The authors are grateful for the discussions with Ren-Hung Hwang on QoS routing, and comments from Subhabrata Sen.

REFERENCES

- [1] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "A directory-based peer-to-peer on-demand streaming service," Tech. Rep. UM-CS-Tech-Report, Department of Computer Science, University of Massachusetts Amherst, 2002.
- [2] Q. Ma and P. Steeniste, "On path selection for traffic with bandwidth guarantees," in *Proc. International Conference on Network Protocols*, October 1997.
- [3] D. Eager, M. Vernon, and J. Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery," in *IEEE Transactions on Knowledge and Data Engineering*, Sep/Oct 2001.
- [4] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internet," in *Proc. IEEE INFOCOM*, April 1996.
- [5] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over peers," Tech. Rep. 2002-21, Stanford University, March 2002.
- [6] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. IEEE Workshop on NOSSDAV*, May 2002.
- [7] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P²cast: P2p patching scheme for vod service," Tech. Rep. UM-CS-2002-034, Department of Computer Science, University of Massachusetts Amherst, 2002.
- [8] D. A. Tran, K. A. Hua, and T. T. Do, "Layered range multicast for video on demand," in *Proc. International Conference on Computer Communications and Networks (IC3N'02)*, October 2002.
- [9] S. Jin and A. Bestavros, "Cache-and-relay streaming media delivery for asynchronous clients," in *International Workshop on Networked Group Communication*, October 2002.
- [10] M. Kamath, K. Ramamritham, and D. Towsley, "Continuous media sharing in multimedia database systems," in *Proc. of 4th International Conference on Database Systems for Advanced Applications (DASFAA'95)*, April 1995.
- [11] A. Dan and D. Sitaram, "A generalized interval caching policy for mixed interactive and long video environments," in *SPIE Multimedia Computing and Networking Conference*, January 1996.
- [12] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A generalized batching technique for video-on-demand systems," in *Proc. IEEE International Conference on Multimedia Computing and Systems*, June 1997.