Decentralized Periodic Broadcasting for Large-Scale Video Streaming

K. K. To and Jack Y. B. Lee
Department of Information Engineering
The Chinese University of Hong Kong
{kkto3, yblee}@ie.cuhk.edu.hk

G. S. H. Chan
Department of Computer Science
Hong Kong University of Science and Technology
gchan@cs.ust.hk

Abstract

Periodic broadcasting (PB) schemes are the most promising solution for building large-scale video streaming services. Existing PB schemes are all built around the traditional client-server model. This paper proposes a radically different architecture where only end-user hosts are used to build the PB system, thereby eliminating the need for a central server or even a service provider. Two unique problems pertaining to this decentralized, peer-to-peer architecture are addressed: transmission scheduling and peer synchronization. A new decentralized synchronization algorithm is proposed and results obtained from simulations confirm the architecture's feasibility and performance.

1. Introduction

Provisioning video streaming services to a large user population has long been a challenge to researchers in multimedia and networking research. The traditional client-server, point-to-point streaming architecture, while matured and well-understood, is not scalable to serve tens of thousands of concurrent users. To address this scalability challenge researchers have developed sophisticated periodic broadcasting (PB) schemes [1-3] to vastly improve scalability and bandwidth efficiency through the use of intelligent multicast streaming algorithms.

Unlike traditional unicast video streaming, the video streams in a PB system are being multicast according to a fixed schedule that is predetermined and is independent of the user arrival pattern. A new user will undergo a startup delay (e.g., in seconds to minutes depending on the broadcasting scheme) to cache some of the video data before playback begins, and continue to receive multicast video data from one or more multicast streams to sustain continuous video playback. Interested readers are referred to the study by Hu [4] for a review and comparison of various PB schemes.

These existing PB schemes all share one property, i.e., they are all based on the client-server model where a central video server (or a few servers working as a cluster) schedules and transmits the video streams over multiple multicast channels. In this paper we investigate a radically different architecture for building a PB system – one without any dedicated server at all.

This server-less architecture is motivated by the rapid developments in computing and networking which made today's commodity computing hardware comparable to yesteryear's server hardware, with an increasing amount of storage, computation, and bandwidth capacities to spare. By exploiting these often available and idle resources in end-user computers, many peer-to-peer streaming systems have been proposed [5-6]. In SplitStream [5], multiple multicast overlays are built on a unicast infrastructure. The source splits the data into a number of stripes and distributes them through different overlays to the receivers. PROMISE [6] focused on multi-sources media streaming which dynamically select a good sender set. Unlike these previous work which focused on conventional streaming, this work investigates a new server-less *periodic broadcasting* architecture that takes advantage of multicast-enabled network to lower the network bandwidth utilization. Moreover, through the use of redundancies the system can even sustain frequent peer failures [7].

As PB algorithms are well-studied, our goal in this work is not to develop new broadcasting algorithms. Instead, we develop a server-less architecture where existing PB algorithms can be deployed and illustrate the procedures to adapt existing broadcasting algorithms for use in this decentralized architecture.

In particular, our work reveals one unique problem in decentralizing any PB algorithms – transmission synchronization. Specifically, end-user machines, or called peers are generally not clock synchronized and so their video data multicast may not be time-aligned. Our simulation studies showed that such mis-aligned multicast can lead to significant network congestion and consequently packet loss (e.g., from 9% to 90% depends on the synchronization level of peers). Note that this problem does not exist in the client-server architecture as the same central server is responsible for scheduling all the multicast video streams based on its own internal clock. Moreover, it cannot be solved by using a large buffer at the receiver as the packet losses occur at the bottleneck routers rather than at the receiver.

In this work we tackle this peer synchronization problem by adapting an existing clock-synchronization protocol for use in the server-less architecture. Our results show that with synchronization the congestion-induced packet loss rate can be reduced to less than 3%. Moreover, to remove the single point of failure of existing synchronization algorithms, we develop a new completely decentralized peer synchronization algorithm that does not

This work is funded in part by a Direct Grant, an Earmarked Grant (CUHK4211/03E) from the HKSAR Research Grant Council, and the UGC Area of Excellence in Information Technology Scheme (AoE/E-01/99).

require any dedicated peer to serve as clock reference, and yet can still achieve excellent synchronization accuracy.

2. Decentralization of Periodic Broadcasting

In this section we present the procedures to decentralize a PB scheme. In particular we use a modified Staggered Broadcasting (SB) scheme as an example to illustrate the design issues in the decentralization process. The same procedure can be applied to other PB schemes.

The modified SB scheme depicted in Fig. 1 is among the simplest form of PB system. A video title of bitrate b and of duration L seconds is divided into N fixed-size segments and the segments are further divided into a number of fixed-size data blocks, which will be distributed to multiple peers in a round-robin manner. For a system with M peers, each peer is then responsible for streaming $1/M^{\text{th}}$ of data of each segment in N different multicast channels repeatedly.

To begin a new video streaming session, the client simply join the first multicast channel and cache for L/N seconds, then the peer starts playback while at the same time joins the next multicast channel for caching the next video segment. More importantly, as the multicast transmission schedules are fixed irrespective of the number of receivers in the system, this modified SB scheme can virtually support an unlimited number of receivers. Note that there are many far more sophisticated PB schemes that can achieve significantly shorter start-up latency, lower bandwidth consumption, or both [4] but the procedures to decentralize them are similar.

Under the modified SB scheme, the upstream bandwidth requirement of each peer is then equal to Nb/M. The peer packetizes the video segment j into packets with size P_k and periodically broadcast them in an interval of

$$U_j = \frac{MP_k}{b}, \ j = 0,1,\ldots,N-1 \qquad (1)$$

to achieve an average bitrate of b/M per channel. As a result, all peers would have the same transmission schedule which is very much alike the original centralized one.

However, a previous work by Chan and Lee [8] showed that when the number of peers increases, the aggregate network traffic from the many peers could congest the network routers and/or the receivers, leading to significant packet loss (e.g., from 9% to 90% depending on the alignment of packet transmissions from different peers).

To address this problem we can explicitly space out the packet transmissions from the peers to achieve a smoother aggregate traffic. Specifically, the schedules of packets of segment j in peer i are shifted by

$$S_{i,j} = U_j \cdot \frac{i}{M} \qquad (2)$$

where i=0,1,…,M-1 and j=0,1, …,N-1.

This transmission schedule can reduce the congestion-induced packet loss to as low as 0.1% if the peers are clock-synchronized. However, synchronizing peers in a decentralized system is far from trivial. The next section addresses this issue.

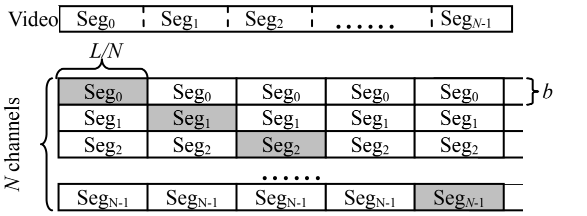


Fig. 1. Modified Staggered Broadcasting transmits a single video segment in each channel

3. Peers Synchronization

There are two problems inherent in peer synchronization. First, the clock of the peers are initially unsynchronized or only loosely synchronized. Second, even after synchronization the hardware clock of the peers will still slowly drift away [9]. The more general problem of clock synchronization has been studied extensively in the literature. One such synchronization algorithm called PCS proposed by Arvind [10] estimates the remote clock by asking the remote machine to send a series of m messages back to the estimator. The estimator then obtains an estimate, T_{est}, of remote machine's clock from

$$T_{est} = R_m - \overline{R}(m) + \overline{T}(m) + \overline{d} \qquad (3)$$

where $\overline{d}$ is an estimate of expected message delay and

$$\overline{T}(m) = \frac{1}{m}\sum_{i=1}^{m} T_i \text{ and } \overline{R}(m) = \frac{1}{m}\sum_{i=1}^{m} R_i \qquad (4)$$

where T_i is the transmission timestamp of the i^{th} message as recorded by the remote machine while R_i is the reception timestamp of the i^{th} message as recorded by the estimator. This synchronization procedure is repeated periodically to refresh the estimates of the remote clocks' values.

3.1. Integrating PCS with Periodic Broadcasting

While we can run the PCS algorithm independently from and concurrently with the PB algorithm, we note that a peer in a decentralized PB system is continually injecting data into the network through the multicast channels. Thus we can exploit this property to piggyback the broadcasting position (i.e., the time with respect to the broadcasting schedule) on the video packets and thus eliminate the message overheads associated with the PCS. In this modified PCS (MPCS), the number of synchronization messages increases with the video bitrate, and peers refresh remote clock estimates from every received packet.

Similar to the original PCS algorithm, MPCS selects one of the peers as the reference peer and sends out the synchronization messages (through piggybacking) for all other peers to estimate the reference peer's broadcasting position, which then adjust their positions accordingly to maintain synchronized broadcasting schedules.

However, as the reference peer is just an ordinary end-user host, it may fail or shut down from time to time. This creates a serious problem as the MPCS algorithm cannot operate without the reference peer. Given that end-user hosts are likely to have significantly lower reliability than dedicated central servers, this reliance on a reference peer is clearly undesirable.

3.2. Distributed PCS

To tackle the single-point-of-failure problem, we present in this section a new distributed PCS (DPCS) that does not require any peer to be designated as a reference peer. Instead, peers under the proposed DPCS algorithm progressively converge and so the effect of individual peer failure will not disrupt the synchronization process.

All peers under the DPCS algorithm piggyback their synchronization messages in the video packets they sent. As the synchronization message size is much smaller than the video packet (e.g., 16 bytes versus 8192 bytes) the additional bandwidth consumed is insignificant. Thus every peer in the system can estimate the broadcasting positions of all other peers by extracting the synchronization messages from the received video packets. After receiving the estimated broadcasting positions of other peers (T_{j,est_i}), peer j then resynchronizes its broadcasting position according to

$$\overline{T}_{j,est} = \frac{1}{M}\sum_{i=0}^{M-1} T_{j,est_i} \tag{5}$$

By setting the broadcasting position to the average of the estimates, we can reduce the variance of the peers' broadcasting positions. However, (5) assumes that all the estimates are taken simultaneously – clearly impossible as synchronization messages arrive at different time instants. To address this problem, we allow peer j to collect more than m synchronization messages before obtaining an estimate T'_{j,est_i} of peer i. After it collected at least m synchronization messages from each peer, then peer j resynchronizes itself using (5) and

$$T_{j,est_i} = T'_{j,est_i} + \left(T_{j,resync} - R_{j,i,last}\right) \tag{6}$$

for i=0,1,…,M–1, where $T_{j,resync}$ and $R_{j,i,last}$ are the timestamps at resynchronization and the last received synchronization message from peer i respectively. From (6) we readjust the estimated T'_{j,est_i} before using it to calculate T_{j,est_i} to compensate for the time lag between the resynchronization time and the time of taking the estimated T'_{j,est_i}. However, as the compensation is calculated locally, this will reduce the accuracy of T_{j,est_i} as the drift rate of the local clock may differ from the remote clocks.

Furthermore, as the peers resynchronize independently and asynchronously, the accuracy of other peers' estimates may be affected as well. For example, assume peer 0 is in the process of collecting synchronization messages from peer 1 when peer 1 resynchronizes itself. This will renders the messages previously received by peer 0 inaccurate as peer 1's broadcasting position has already been adjusted to a new value. To address this issue we add an offset field in the synchronization message to specify the difference of broadcasting positions before and after resynchronization. With this additional field the other peers can then adjust the estimate of that peer without significant loss of accuracy.

If a peer fails, it will cause a timeout in the estimation procedure which only affects the accuracy of $\overline{T}_{j,est}$ but will not disrupt the synchronization process. Given that there are many peers in the system, the failure will be effectively masked by the averaging step in computing $\overline{T}_{j,est}$.

4. Performance Evaluations

In this section, we present simulation results to evaluate the proposed server-less PB architecture under different synchronization algorithms. We apply the extended BA model proposed in [11] to generate network topologies with 500 routers for use in the simulations. The peers are randomly attached to one of the edge routers in the network. The core network has abundant bandwidth and introduces random delay which simulates the effect of cross traffic. Edge routers have separate buffers for each outgoing link and packets are dropped if their corresponding outgoing link buffer is full. Each set of results is obtained from the average of six randomly-generated network topologies and peer placements. Table 1 summarizes the other default parameters used in the simulations.

We employ a simple multicast routing algorithm which constructs a source-based shortest-path tree for every peer. This resembles many of the existing multicast routing protocols such as DVMRP, MOSPF and PIM-SM. The clock skew of each peer, defined as the different between peers' clocks, is modeled as a normally-distributed random variable with zero mean; and the clock drift rate of each peer is modeled as a normally-distributed random variable with a mean of one. Note that a clock with drift rate one represents a correct real-time clock.

We first investigate the packet loss rate of the server-less PB system under different synchronization algorithms. Fig. 2 plots the packet loss rate versus the variance of the peers' clock drifts. A larger variance represents that the peers' clocks drift away more rapidly from one another, thus leading to asynchrony in the broadcasting positions. The results show that if no synchronization is employed the packet loss rate can reach 8%, which is clearly unacceptable. The high packet loss rate remains even if we randomize (within the multicast period of length L/N seconds) the transmission time of packet because the burstiness of the aggregate traffic is due to the large number of peers sending data simultaneously.

By contrast, when the peers are synchronized using MPCS and DPCS the loss rate drops to 2.8% and 5% respectively. Note that in general MPCS outperforms DPCS at the expense of requiring a particular peer to be designated as the reference peer for the synchronization process to work. By contrast, DPCS is completely decentralized and does not suffer from single point of

Table 1. Default simulation parameters

Parameter	Value	Parameter	Value
Avg. Network Delay	0.05 s	Peer Uplink BW	4Mbps
Edge Router Buffer	32 kB	Peers Count, M	100
Avg. Clock Skew Var.	1 s^2	Video Length, L	3600 s
Avg. Clock Drift	1 s/s	Video Bitrate, b	4Mbps
Avg. Clock Drift Var.	1^{-12} s^2/s^2	Startup Latency	600 s
Sync. Msg. Count, m	10	Sys. Utilization	95%
Avg. Sync. Msg. Delay	0.05 s	Packet Size, P_k	8kB
Peer Downlink BW	4.4Mbps	Simulation Time	10 hrs

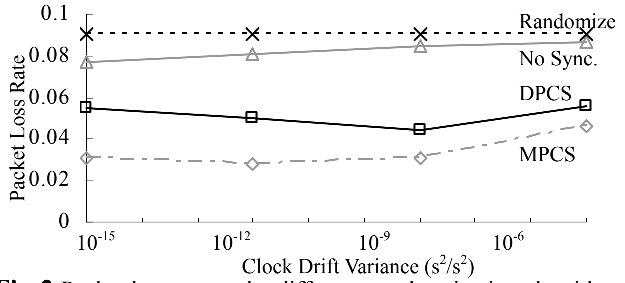


Fig. 2. Packet loss rate under different synchronization algorithm

failure. We are currently investigating the effect of peer failures on the MPCS algorithm to develop efficient recovery protocols to improve its robustness.

Another observation in Fig. 2 is that the packet loss rates are all generally insensitive to the variance of clock drifts. This is because inter-transmission time of synchronization messages is relatively short (e.g., 1.5 seconds), thus the asynchrony caused by the clock drift is corrected quickly.

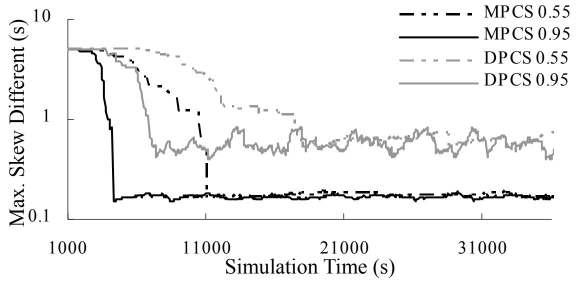
In Fig. 3 we turn our attention to the dynamic behavior of the system. In particular, we study the maximum skew among peers' broadcasting positions and the skew variance against time under 55% and 95% system utilization respectively. Without synchronization the maximum skew (not plotted) will increase with time as the peers' broadcasting positions continue to drift away from each other (e.g., reaching 6s after a simulation time of 6,000s).

By contrast, the maximum skew under MPCS and DPCS quickly converge to 0.16 seconds (after 4,500 seconds) and 0.5 seconds (after 7,300 seconds) respectively at 95% system utilization. MPCS converges faster than DPCS and achieves a lower maximum skew because peers under DPCS calculate the new broadcasting position by averaging the estimates of all other peers' positions. As each peer synchronizes asynchronously, there are cases where a peer synchronizes to a new position when other peers are still using old estimates. This results in the slower convergence rate and the higher skew of DPCS.

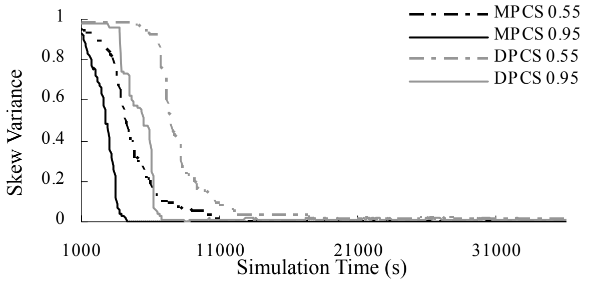
Another observation is that the system utilization has a substantial effect on the synchronization performance. This is a direct consequence of piggybacking synchronization messages within the video packets, where only the active peers streaming and playing back video participate in the synchronization process. To address this problem we can either let the idle peers receive video data periodically to maintain synchronization or decouple the synchronization messages from the video data altogether.

5. Conclusions

With the growing popularity of peer-to-peer systems and applications, it is only a matter of time for many existing applications to migrate to decentralized architectures. This paper tackled the transmission scheduling and synchronization challenges in decentralizing periodic broadcasting schemes for large-scale video streaming applications. The early results are very encouraging and more work is needed to further investigate (a) the decentralization of more sophisticated periodic broadcasting algorithms; (b) applications other than streaming (e.g., software update, data distribution); and (c) multicasting over application-layer multicast for deployment over the current Internet, which does not yet support native multicast at the global scale.



(a) Maximum Skew Different



(b) Skew Variance

Fig. 3. Dynamic behavior of MPCS and DPCS

References

[1] L. S. John and L. M. Tseng, "Staircase Data Broadcasting and Receiving Scheme for Hot Video Service," *IEEE Trans. on Consumer Electronics*, vol. 43, no. 4, Nov 1997, pp.1110-7

[2] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," *Proc. of the ACM SIGCOMM '97*, vol. 27, no. 4, Sep 1997, pp. 89-100.

[3] S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting," *IEEE Multimedia Systems*, vol. 4, no. 4, Aug 1996, pp. 197-208.

[4] A. Hu, "Video-on-Demand Broadcasting Protocols: A Comprehensive Study," *Proc. of the IEEE Infocom 2001*, Anchorage, AK, Apr 2001, pp. 508-517.

[5] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," *Proc. of 19th ACM SOSP'03*, New York, Oct., 2003.

[6] M.Hefeeda, A. Habib, B. Botev, D. Xu and D. B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," *Proc. of ACM Multimedia*, 2003.

[7] Jack Y. B. Lee and W. T. Leung, "Design and Analysis of a Fault-Tolerant Mechanism for a Server-Less Video-On-Demand System," *Proc. 2002 International Conference on Parallel and Distributed Systems*, Taiwan, Dec 17-20, 2002, pp.489-494.

[8] C. Y. Chan and Jack Y. B. Lee, "On Transmission Scheduling in a Server-less Video-on-Demand System," *Proc. International Conference on Parallel and Distributed Computing*, Klagenfurt, Austria, August 26-29, 2003.

[9] F. Cristian, "A Probabilistic Approach to Distributed Clock Synchronization," *Distrib. Comp.*, vol. 3, 1989, pp. 146-158.

[10] K. Arvind, "Probabilistic Clock Synchronization in Distributed Systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 5, May 1994, pp. 474-487.

[11] R. Albert, and A.-L. Barabási, "Topology of Evolving Networks: Local Events and Universality," *Physical Review Letters*, vol. 85, 2000, pp. 5234-5237.