Placement of Continuous Media in Wireless Peer-to-Peer Networks

Shahram Ghandeharizadeh, *Member, IEEE,* Bhaskar Krishnamachari, *Member, IEEE,* and Shanshan Song, *Student Member, IEEE*

***Abstract*— This study investigates a novel streaming architecture consisting of home-to-home online (H2O) devices that collaborate with one another to provide on-demand access to large repositories of continuous media such as audio and video clips. An H2O device is configured with a high bandwidth wireless communication component, a powerful processor, and gigabytes of storage. A key challenge of this environment is how to place data across H2O devices in order to enhance startup latency, defined as the delay observed from when a user requests a clip to the onset of its display. Our primary contribution is a novel replication technique that enhances startup latency while minimizing the total storage space required from an environment consisting of $\mathcal{N}$ H2O devices. This technique is based on the following intuition: the first few blocks of a clip are required more urgently than its last few blocks and should be replicated more frequently in order to minimize startup latency. We develop analytical models to quantify the number of replicas required for each block. In addition, we describe two alternative distributed implementation of our replication strategy. When compared with full replication, our technique provides on average greater than 97% (i.e. several orders of magnitude) savings in storage space while ensuring zero startup latency and a hiccup-free reception.**

***Index Terms*— Continuous display, data placement, replication, peer-to-peer networks.**

I. Introduction

Advances in computer processing, storage performance and high speed wireless communications have made it feasible to consider peer-to-peer network of economical devices that provide access to a large volume of data. Intel, for example, offers a small device that consists of a 500 MHz processor and a wireless component that operates in the 5 GHz spectrum, offering transmission rates in the order of tens of Megabits per second, Mbps. The cost of this device is approximately $85. Similar to desk top personal computers, one may extend this device with mass storage.

One application of these devices is to stream continuous media, e.g., audio clips, movies, news clips, etc., for home entertainment systems. When compared with traditional data types such as text and still images, continuous media consist of a sequence of quanta, either audio samples or video frames, that convey meaning when presented at a pre-specified rate [7], [12]. Once the display is initiated, if the data is delivered below this rate then a display might suffer from frequent disruptions and delays, termed hiccups. This paper assumes constant bit rate (CBR) continuous media with a fixed display bandwidth requirement. A novel feature of this framework is that each home-to-home online (H2O) [9], [11] device may employ its resources to participate in delivery of multimedia content to an actively displaying H2O device. One example deployment of H2O might be that of Figure 1. A household may store its personal video library on a H2O cloud. This would make the library widely available to enable a user to retrieve their content anywhere, e.g., at a friend's home. The system might encrypt the content to either protect it from un-authorized access, i.e., authentication, or implement a business model for generating revenues.

Fig. 1. Home-to-home on-line devices streaming continuous media.

Note that H2O framework complements the existing wired solutions based on xDSL technology or cable networks [23], [8]. In [19], it was noted that the overall bandwidth required to implement an interactive video-on-demand solution based on a naive design that employs one centralized server would be as high as 1.54 Petabytes per second for the entire United States. By replicating blocks so that they are closer to a consuming H2O device, the number of accesses to a centralized server is minimized. This means that a H2O device might act in three possible roles: 1) producer of data, 2) an active client that is displaying data, and 3) a router that delivers data from a producer to a consumer of data. At times, there might be congestion in the system limiting the bandwidth available to a H2O device, say H2O$_j$. If H2O$_j$ references clip X with bandwidth requirements ($B_{Display}$) in excess of the bandwidth of the connection to H2O$_j$ (B_{Link}) then H2O$_j$ must prefetch enough data to prevent H2O$_j$ from starving for data. Assuming S_C denotes X's size, the amount of prefetch data is [10]: $S_P = S_C - \lfloor \frac{B_{Link}}{B_{Display}} \times S_C \rfloor$. The time required to stage S_P bytes of data at the H2O device dictates the startup latency

incurred by that device. To illustrate, if $B_{Link} = 1.5$ Mbps and $B_{Display} = 4$ Mbps then the display of a 2 hour movie must prefetch 2.2 Gigabytes of data, resulting in a 75 minutes delay. To simplify discussion, this paper assumes a wireless peer-to-peer network of H2O devices where the transmission bandwidth of each device is in the order of tens of Mbps and each device is able to estimate the available bandwidth to its neighbors. The wired infrastructure continues to serve as the backbone of a H2O cloud.

The H2O devices might be configured in a variety of ways to support a hiccup-free display. One may require a displaying H2O device to download a clip in its entirety from one or more remote mirrored servers prior to initiating its display, using a technique such as [21]. This paradigm suffers from the following limitations. First, the user might perceive loss of data when the mirrored servers containing a referenced data item become unavailable due to either hardware failures, network partitioning, high system load, etc. Second, a H2O device that is many hops away from the base station (and hence the remote mirrored servers) might observe a long delay from when its user references a clip to onset of its display, termed startup latency. It is long because (1) the first block of a clip must make multiple hops to arrive at the target H2O device, and (2) the requesting H2O device may not start display until sufficient data has arrived to compensate for network bandwidth fluctuations (due to its load). One may minimize startup latency by prefetching data. A H2O device may further reduce the startup latency by caching the prefetch portion of as many clips as possible. One extreme form of prefetching is to gradually replicate the entire repository on each H2O device. However, even if bandwidth is not a limiting factor, the storage capacity of each device might limit access to the entire repository. For example, to store 1000 two hour movies with $B_{Display}$=4 Mbps, each H2O device must be equipped with more than 3 Terabytes of storage. This storage requirement increases when one considers each household's video library. Our proposed replication strategy addresses this storage limitation.

A recent study [5] investigates replication of popular data items in an unstructured peer-to-peer system for a non-streaming framework. It employs analytical models to observe that a technique based on the square root of the popularity of a data item yields the best Mean Search Size (MSS), defined as the number of walks necessary to locate the referenced data item. It does not consider either hiccup-free display of continuous media, its average startup latency, or partial replication of a data item.

Numerous studies have analyzed the role of proxy servers and partial caching of continuous media in the context of Internet [28], [24], [2], [22]. These efforts are in the context of unicast delivery of a stream to a client. In our framework, a request floods the network and multiple H2O devices might produce different blocks of a referenced clip. Other proxy caching studies have focused on the use of multi-cast or broadcast protocols [6], [13], [27]. These studies are different from our framework because the radio range of each H2O device dictates its connectivity with other H2O devices. Thus, a block may potentially make multiple hops in order to arrive at a target destination. Pyramid [26], Permutation-based pyramid [1] and Skyscraper [14] are broadcasting techniques for delivery of data. These techniques are complementary to placement of data and our proposed replication strategy. An adaptation of these techniques to a H2O cloud is a future research direction.

Term	Definition
Continuous media	A sequence of quanta, either audio samples or video frames, that convey meaning when presented at a pre-specified rate.
Hiccup	Disruptions and delays encountered by a display when its H2O device starves for data.
Startup latency	Delay from when a H2O device requests a clip to the time the display begins.
Throughput	Number of simultaneous displays supported by participating H2O devices.
Kbps	Kilo bits per second.
Mbps	Mega bits per second.

TABLE I
TERMS AND THEIR DEFINITIONS

In this study, we assume H2O devices may participate either as a client, a proxy cache server for other H2O devices, or both. Our primary contribution is a novel replication technique that is a hybrid of partial replication and prefetching. Its main insight is that the first few blocks of a clip are required more urgently than its last few blocks. Hence, it replicates these blocks more frequently in order to minimize startup latency. While a H2O device is displaying these blocks, other H2O devices with relevant blocks transmit their blocks to facilitate a hiccup-free display. Two key assumptions of our proposed technique are as follows: First, the total size of available audio and video clips exceeds the storage capacity of one device. Second, the bandwidth between two H2O devices exceeds the bandwidth required to display a clip. This assumption is realistic because: (a) the average bandwidth required for DVD-quality video is typically quoted at 4 Mbps, and (b) emerging wireless protocols such as 802.11a provide transmission rates in the order of tens of Mbps.

The rest of this paper is organized as follows. In Section II, we describe our proposed technique. Section III analyzes both worst and average cases for deciding how frequently a block should be replicated, along with the optimal block size. Next, Section IV outlines a distributed technique for controlling the placement of data. We conclude with brief conclusions and future research directions in Section V.

II. Data Placement and Replication

When a H2O device is deployed in a household, it registers itself with a base station (similar to how cellular phones register with their base stations). This process might flood the network with a register command. The base station requests different H2O devices to stage certain blocks of each clip with the objective to ensure that all H2O devices in its coverage area contain all blocks of different clips. This placement of data also implements our proposed replication technique.

Parameter	Definition
S_C	Size of a clip.
S_P	Amount of data prefetched to minimize startup latency.
z	Number of blocks for a video clip.
$S_{C,R}$	Total space occupied by a clip with our proposed placement technique.
S_b	Size of a block.
$B_{Display}$	Bandwidth required to support a continuous display.
D	Display time of a block, D=$\frac{S_b}{B_{Display}}$.
b_i	block i of a clip.
r_i	Number of copies of a block b_i.
h	Time to retrieve a block from a H2O device that is one hop away.
R	Radio range measured in meters.
H_i	Number of hops tolerated by block b_i, $\lfloor H_i = \frac{(i-1)D}{h} \rfloor$.
$\mathcal{N}$	Number of H2O devices.

TABLE II
PARAMETERS AND THEIR DEFINITIONS

In order to present our replication scheme, we assume a cycle-based display technique [20], [25], [4], [3], [18] for each H2O device. This technique assumes all blocks of a video clip are equi-sized. It displays one block with size S_b in one cycle. The number of cycles is dictated by the number of blocks that constitute the clip. The duration of a cycle is dictated by the display time of a block which is fixed at D seconds assuming constant bit rate (CBR) continuous media, i.e., D=$\frac{S_b}{B_{Display}}$. This technique requires block b_i to be memory resident before the display of block b_{i-1} ends. Thus, if the time to retrieve a block from a H2O device that is one hop away is h then block i may tolerate H_i hops where $H_i = \frac{(i-1)D}{h}$ relative to start of a display. To simplify discussion, assume h is a fixed constant; see Section V for a discussion of h. Moreover, it is acceptable for a H2O device to receive block b_i in fewer than H_i hops. This is because it has sufficient disk bandwidth and storage to store these blocks for future use.

The core replication and placement strategy is as follows. For each video clip X:

- Divide X into z equi-sized blocks, each S_b in size.
- Place the first block of X, b_1, on all nodes in the network. To simplify discussion, assume each H2O device is configured with sufficient storage capacity to store the first block of all clips. Due to space constraints, we do not discuss those scenarios when this assumption is violated.
- For each block b_i of X, $1 \leq i \leq z$, compute its delay tolerance H_i. This is the farthest number of hops that the block can be located from the H2O device that displays clip X.
- Based on H_i, compute r_i, the total number of replicas for block b_i. The value of r_i decreases monotonically with i until it reaches 1.
- Construct r_i replicas of block b_i and place each copy of the block in the network in such a way as to ensure that for all nodes there exists at least one copy of the block b_i that is no more than H_i hops away.

Fig. 2. A linear traversal of H2O devices

The value of r_i is a topology dependent computation. In Section III, we consider different topologies and their impact on the amount of storage required by our proposed technique. Section IV describes a distributed implementation of our technique.

III. Modeling and Performance Analysis

We analyze the value of r_i with three different topologies. The first is based on a worst-case scenario. This model is intentionally pessimistic (biased against our proposed approach) to show our approach provides savings even with these assumptions. The second computes the average case based on a grid topology of $\mathcal{N}$ H2O devices with $\sqrt{\mathcal{N}}$ devices along each of the x and y-axis. The last analyzes a graph topology based on a fixed area A and radio range R of the H2O devices. We quantify the storage requirements of our technique with each of these topologies for a clip with size S_C. With the grid and graph topologies, we derive an optimal block size, S_b, to minimize total required storage. Next, in Section III-D, we compare the impact of these topologies when compared with one another and a full replication scheme.

A. Worst-case linear topology

The simple topology of Figure 2 assumes the $\mathcal{N}$ H2O devices are connected to one another sequentially. With r_i copies of block b_i, when a H2O device requests b_i, in the worst case scenario, this request must visit $\mathcal{N} - r_i - 1$ H2O devices to retrieve b_i. In order to observe a zero startup latency, block i should be replicated r_i times where $r_i = \mathcal{N} - H_i$. The value of r_i for the last blocks of a video clip might be a negative number. In these cases, we reset the value of r_i to one to ensure the availability of at least one block. The system stops replicating those blocks whose index exceeds U_r where $U_r = \lceil \frac{(\mathcal{N}-1)h}{D} + 1 \rceil$. This means the total storage space occupied by a clip with z blocks once replicated, $S_{C,R}$, is:

$$S_{C,R} = \begin{cases} S_b \times (z\mathcal{N} - (\frac{D}{h} \times (\frac{(z+1)z}{2} - z))) & if\ z \leq U_r \\ S_b \times (z - U_r + U_r\mathcal{N} - (\frac{D}{h} \times (\frac{(U_r-1)U_r}{2}))) & if\ z > U_r \end{cases} \quad (1)$$

With full replication, total storage required by a clip increases as a linear function of $\mathcal{N}$, i.e., $\mathcal{N}\ S_C$.

To illustrate the benefits of our replication technique, Figure 3 shows the percentage savings with our technique when compared with full replication. The x-axis of this figure is the display time of a clip which varies from 2 to 120 minutes with increments of 2 minutes. The y-axis is the percentage saving defined as $100 \times (1 - \frac{S_{C,R}}{S_C \times \mathcal{N}})$. This figure assumes: a) an environment with 1000 H2O devices, $\mathcal{N}$=1000, b) one hop delivery of a block requires 0.5 seconds (h=0.5), and c) the bandwidth required for each clip is constant and fixed at 4

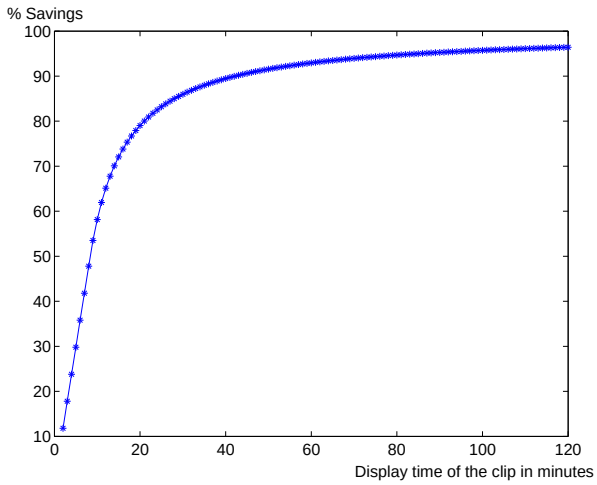


Fig. 3. Percentage savings with our technique when compared with full replication with linear topology.

Mbps, $B_{Display} = 4\ Mbps$. Thus, the size of a clip, S_C, varies from 60 to 3600 MB. The size of a block is fixed at 1 MB, S_b=1 MB, resulting in a display time of 2 seconds for each block, D=2.

As the display time of a clip is increased from 2 to 120 minutes, the percentage savings provided by our technique increases. This is because z exceeds U_r, which means that some blocks consists of only one copy. The percentage of these blocks dominates as the display time of a clip increases to two hours, providing grater savings when compared with full replication.

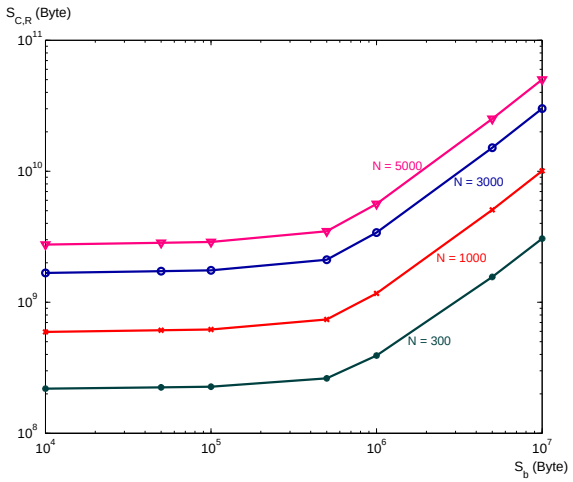
To illustrate, a 2 minute clip consists of 60 blocks (z=60): b_1, b_2, ..., b_{60}. Hence, block b_2 should be replicated 996 times while the last block b_{60} should be replicated 764 times. Total storage required by a clip is now 52 Gigabytes while full replication requires 60 Gigabytes. With a 20 minute clip, the number of blocks that constitute this clip increases to 600, and the algorithm constructs one copy of each block with an index equal to or greater than 251, U_r=251. Now, total storage required by a clip is 122 Gigabytes (instead of 600 with full replication). U_r remains fixed at 251 with longer clips, providing greater savings when compared with full replication.

Section III-D analyzes the behavior of our algorithm as a function of $\mathcal{N}$.

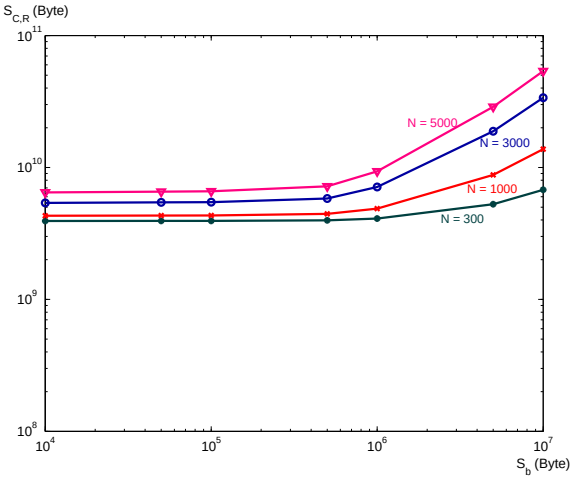
B. Grid topology

With a topology that organizes $\mathcal{N}$ nodes in a square grid of fixed area, each node (with the exception of those nodes on the edge) neighbors only the four nodes in each cardinal direction. At least one copy of block b_i must be placed within H_i hops of every node (where H_i is as defined in table II). For the grid topology, there are $2H_i^2 + 2H_i + 1$ nodes within H_i hops of any given node. Therefore, the number of replicas r_i required for block b_i is given as:

$$r_i = \lceil \frac{\mathcal{N}}{2H_i^2 + 2H_i + 1} \rceil \qquad (2)$$



4.a. Two minute clip, S_C=60 MB



4.b. Two hour clip, S_C=3600 MB

Fig. 4. Total storage space required as a function of block size with grid topology, h=0.75 Seconds.

The expected total storage required in the network for a clip with z blocks of size S_b is therefore:

$$S_{C,R} = \sum_{i=1}^{z} r_i S_b = S_b \sum_{i=1}^{z} \lceil \frac{\mathcal{N}}{2H_i^2 + 2H_i + 1} \rceil \qquad (3)$$

We numerically studied the behavior of this model for clip sizes ranging from 2 minutes to 2 hours (S_C ranges from 60 MB to 3600 MB), various per-hop delays (h varies from 0.125 seconds to 1.5 seconds), and network sizes ($\mathcal{N}$ ranges from 300 to 5,000 H2O devices). Figures 4.a and 4.b show the total required storage space as a function of block size for the two clip sizes.

These figures show that the total storage requirement increases with the block size (in other words, it decreases with increasing number of blocks). While this suggests it is beneficial to maximize the number of blocks that constitute a clip, a system designer must consider the complexity of the replica placement algorithm. Section IV shows this complexity to increase as a function of the number of blocks. Thus it is advisable to choose a block size that balances these conflicting

factors. Figures 4.a and 4.b suggest that a good solution would be to choose a value of the block size that matches the knee of the curve (i.e. the critical point at which the total storage requirement curve starts to rise sharply). From our experiments we find that the choice of this critical block size S_b depends critically on the per-hop latency h and is independent of the network size $\mathcal{N}$ and the clip-size S_c. Finally, we note that as shown in Section III-D the percentage savings in storage obtained by partial replication (when compared with full replication) is greater than 98%. This holds true for all our experiments.

C. Average Case Graph topology

With the graph topology, the network connectivity depends on the radio range R of individual devices. Assuming $\mathcal{N}$ nodes are scattered in a fixed area A, each node communicates with those nodes in its radio range. As before, at least one copy of block b_i must be placed within H_i hops of every node (where H_i is as defined in table II). For the graph topology, it can be shown that there are on average between $\frac{(2\gamma(H_iR)^2)\mathcal{N}}{A}$ and $\frac{(\pi\gamma(H_iR)^2)\mathcal{N}}{A}$ nodes within H_i hops of any given node (for $H_i \geq 1$). Here γ is a density-dependent correction factor between 0 and 1, it can be approximated by 1 when the network is very dense and there are many nodes distributed evenly across the region. Using the upper bound, the number of replicas r_i required for block b_i is:

$$r_i \approx \begin{cases} \mathcal{N} & if\ H_i < 1 \\ \lceil \frac{A}{\pi\gamma(H_iR)^2} \rceil & if\ H_i \geq 1 \end{cases} \quad (4)$$

The expected total storage required in the graph-topology network for a clip with z blocks of size S_b is again:

$$S_{C,R} = \sum_{i=1}^{z} r_i S_b \quad (5)$$

We analyzed this model with different parameter settings. When invoked with the parameter settings of the Grid topology, we obtained results and trends similar to those of Figure 4. Hence, we refer the reader to the discussions of Section III-B. Note that a choice of γ value changes the storage requirements of this topology, i.e., scales the graphs of Figure 4 vertically. Section IV validates the accuracy of these analytical models by comparing them with results obtained from a simulation study of our replication strategy.

D. A comparison of alternative topologies

We compared the percentage savings in storage space offered by each assumed topology when compared with full replication as a function of $\mathcal{N}$, see Figure 5. (See Section III-A for a definition of percentage savings.) Our proposed technique with both the average grid and graph topologies provide several orders of magnitude savings in storage space when compared with full replication. Their percentage savings are greater than 97%.

With the worst-case linear topology, we analyzed different clip sizes to compute $S_{C,R}$. The block size is 1 MB in all experiments. With short clips, e.g., 2 minutes long, our proposed scheme starts to degenerate into full replication with large values of $\mathcal{N}$. Its percentage saving is maximized with approximately 250 H2O devices. Note that with the worst case assumptions of linear, our technique continues to provide more than 80% savings with a two hour clip.

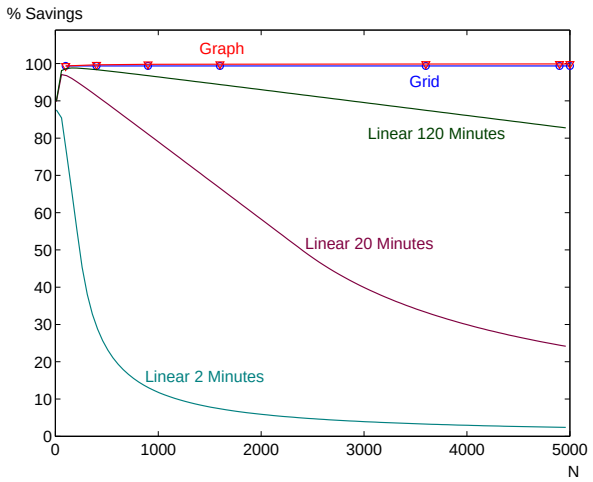


Fig. 5. Percentage savings with worst case linear (3 different clip sizes), and the grid and graph topologies when compared with full replication.

IV. Distributed Implementation

We now discuss two distributed implementations of our proposed replication technique. Both control the placement of r_i copies of each block b_i with the following objective: each node in the network is within H_i hops of at least one copy of b_i. Assuming a user employs a $H2O_p$ to publish a clip X, the general framework of both implementations is as following. First, $H2O_p$ computes the block size S_b, the number of blocks z, and the required hop-bound H_i for each block, using expressions of Section III. Next, it floods the network with a message containing this information, querying which H2O device will host a copy of which block of X. Each recipient of this message, say $H2O_j$, computes a binary array, A_j, that consists of z elements. For each element i of A_j that is a one, $H2O_j$ contacts $H2O_p$ for a copy of block b_i. $H2O_p$ may employ either a multicast or an unicast protocol to publish those blocks with many copies.

The two alternative implementations are differentiated in how $H2O_j$ computes binary array A_j, i.e., identity of those blocks of X resident on $H2O_j$. The first implementation, termed TIMER, employs a distributed timer-suppress algorithm. The second, termed ZONE, assumes existence of nodes with geopositioning information that makes them aware of both their (x,y) coordinates and the extent of the service area. In the following, we detail each technique. This section concludes with a comparison of these techniques using a simulation study.

1. TIMER: When $H2O_j$ receives the flooded query message, it performs z rounds of "elections", one for each block of clip X. During each election i, $H2O_j$ determines if it will maintain a copy of block b_i. For block b_i, each node picks

a random timer value from 1 to M (to avoid unnecessary duplicates, M should be much greater then N, the number of nodes in the network) and starts to count down. When the timer at a given node counts down to zero, if the node is not already "suppressed", it elects to store a copy of block b_i and sends a suppress message to all nodes within H_i hops of itself (via controlled-flooding). At the end of round i, it is easy to see that every node will be in one of two states: either it has elected to hold a copy of block b_i or it is suppressed. In either case, every node in the network is guaranteed to be within H_i hops of a node that has elected to hold block b_i. When the timer of multiple nodes within H_i hops expires at the same time, this technique generates more copies of a block than necessary. One may extend TIMER to detect and compensate for these scenarios.

2. ZONE: This distributed algorithm assumes each node is aware of its (x,y) coordinate and the extent of the service area. It uses this information to space-out copies of a given block. This is accomplished by placing each copy in a separate square zone whose size is such that all nodes can be reached within H_i hops from a node in the zone. For ease of exposition let us consider that all nodes in the network fit within a square of $S \times S$ (its generalization to an arbitrary rectangle is trivial). Then for block b_i, the size of each square zone of size $s_i \times s_i$ must be such that it fits within a circle of radius $H_i R$. It can be shown that the side of this zone should be $s_i = \gamma H_i R\sqrt{2}$, where $\gamma \leq 1$ is a correction factor that should depend on the node density. For lower node densities, it is best to have a smaller value of γ which results in more copies. If the whole area is broken into zones of size s_i, then it is advisable to place a copy near the center of each zone. Assuming the area spans coordinates $(0,0)$ to (S,S), the center of each zone i occurs at $((l+0.5)s_i, (k+0.5)s_i)$ with both l and k ranging from 0 to $\lceil \frac{S}{s_i} \rceil - 1$ in value.

In this distributed algorithm therefore, when each node receives the flooded query, there are again z rounds of elections, one for each block of the clip. In each round, all nodes first determine which zone they belong to, based on the knowledge of the block number b_i and the hop bound H_i. Then in each zone corresponding to block b_i, all nodes participate in a distributed leader election protocol (a simple wave algorithm such as FloodMax [17] would suffice), whereby the node that is closest to the zone center is elected to hold a copy of the block b_i. This node then sends a suppress message to all nodes within the zone that are H_i hops away. If any nodes within a zone are still unsuppressed (which should happen rarely if the value of γ is chosen carefully by the system designer), they may revert to a timer-suppress scheme that is restricted to the zone. This guarantees that all nodes within each zone are within H_i hops of a node that has elected to hold a copy of block b_i.

We simulated these two algorithms using the C# programming language. We experimented with a variety of parameter settings such as node to area densities, radio ranges, clip sizes, etc. Figure 6 shows one experimental result simulating a 1km by 1km square area with 300 randomly distributed H2O devices and a clip consisting of 60 blocks (z=60). The x-axis of this figure is the block-id, ranging from 1 to 60. The y-axis

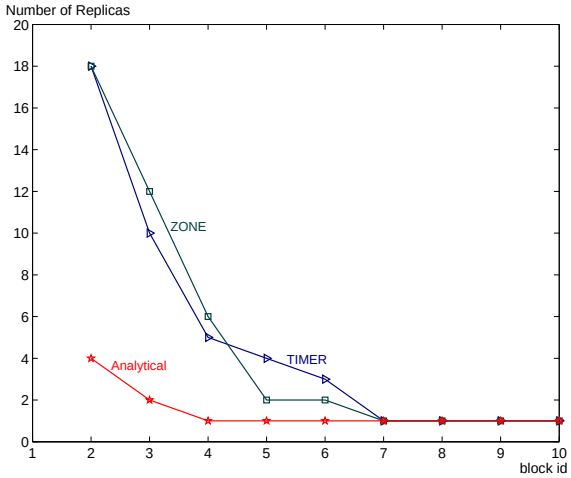


Fig. 6. Performance of TIMER and ZONE with a graph topology ($\mathcal{N} = 300$, R = 100 meters, A = 1 million square meters). This figure does not show 300 replicas of block 1 with all techniques.

denotes the number of replicas for each block with the two alternative techniques and the analytical model of Section III-C configured with γ=1. The radio range of each node is 100 meters. With TIMER, we used a single random placement of nodes and report the average number of replicas for each block across 100 instantiations of the randomized timers with the maximum value set at 1000 $\mathcal{N}$. With ZONE, we average across 100 random placement of nodes. The obtained results show the following. First, both distributed implementations require a few more replicas per block than predicted by the idealized analytical expressions. This is due to the randomly placed H2O devices at the edges of our square area that incur more than H_i hops for the first few (seven) blocks. The analytical model of Section III-C ignores the impact of these nodes. TIMER and ZONE are forced to construct additional copies of the first few blocks in order to satisfy their H_i requirement. The percentage savings offered by these strategies is still several orders of magnitude superior to full replication. Table III shows the total number of blocks with each technique and percentage savings relative to full replication. Figure 7 shows the percentage difference between the analytical models (with γ=1) and TIMER with different number of nodes, percentage difference = $100 \times \frac{TIMER - Analytical}{Analytical}$. The x-axis of this figure is the block size. The clip size is fixed at 60 MBytes. Thus, a larger block size (S_b) results in fewer blocks (z). The results show the accurracy of the graph analytical model with a large number of nodes. With 300 nodes, a smaller γ value improves the estimations provided by the analytical models, compare $\mathcal{N}$=300 with γ=1.0 and γ=0.5.

TIMER realizes a uniform distribution of blocks across the H2O devices. Figure 8 shows the percentage of nodes with a specific block count with three different clip sizes: 2, 20, and 120 minutes of display time (60, 600, and 3600 MB in size, respectively). This figure shows the average number of blocks per device for a given clip length. With the 2 minute clip, each node should have 1.32 blocks. The results show that more than 70% of the nodes have one block. Longer clips result in a more uniform distribution of blocks. With

$\mathcal{N}$	Analytical		TIMER		ZONE	
	$S_{C,R}$	Savings	$S_{C,R}$	Savings	$S_{C,R}$	Savings
300	363	97.98%	394	97.81%	394	97.98%
1000	1063	98.23%	1079	98.20%	1075	98.23%
3000	3063	98.30%	3076	98.29%	3078	98.30%
5000	5063	98.31%	5074	98.31%	5079	98.31%

TABLE III

A COMPARISON OF ANALYTICAL MODELS FOR THE GRAPH TOPOLOGY WITH THE TWO DISTRIBUTED IMPLEMENTATIONS (Z = 60, R = 100 METERS, A = 1 MILLION SQUARE METERS).

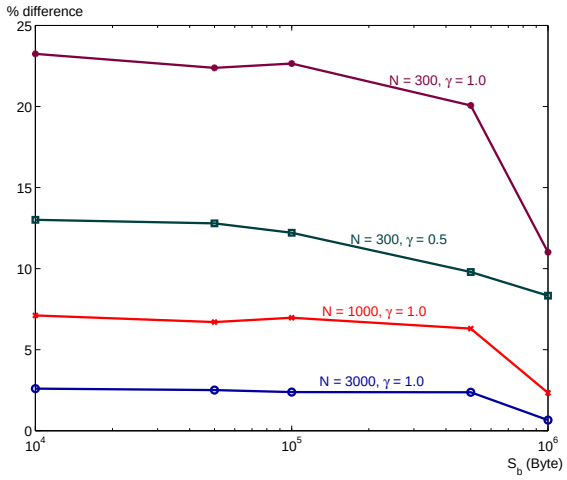


Fig. 7. Percentage difference between the analytical model and TIMER ($S_C = 60$ MBytes, R = 100 meters, A = 1 million square meters).

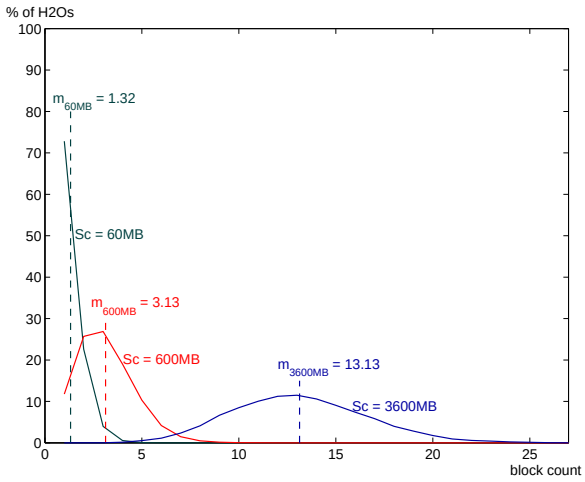


Fig. 8. Percentage of H2O devices with a specific number of blocks using TIMER ($\mathcal{N} = 300$, R = 100 meters, A = 1 million square meters). Expected average number of blocks per node for a clip with size S_C is marked as a dashed line and denoted m_{S_C}.

the 120 minutes clip, each node should have approximately 13.13 blocks. The obtained results show a normal distribution with a mean that approximates 13.13 and a reasonable standard deviation, e.g., approximately 80% of nodes contain between 9 to 17 blocks. Note that with multiple clips of varying sizes, as per the law of large numbers, the standard deviation around the mean decreases even further, resulting in a balanced utilization of storage across the entire network.

When compared with TIMER, ZONE cannot distribute blocks as uniformly across the nodes. This is because ZONE favors placement of those blocks with a large H_i value towards the center of a zone. While one may extend ZONE to enhance its storage utilization, our results demonstrate the superiority of the simple TIMER technique.

V. CONCLUSION

This paper explored a novel architecture that consists of collaborating H2O devices to provide on-demand delivery of a clip, i.e., minimal startup latency. Our primary contribution is a replication technique that replicates the first few blocks of a clip more frequently because they are needed more urgently.

This study assumed a fixed value for h, the time to retrieve a block from a H2O device that is one hop away. With wireless ad hoc networks of H2O devices, h is a function of the number of transmitting H2O devices in the same radio range [15]. The system may utilize the worst expected h value when computing the number of replicas for each block, diminishing the percentage savings offered by our replication technique. However, we believe each H2O device must be extended with an admission control module and a dynamic data delivery scheduling technique to address variability of h. With admission control, a new request to display a clip at H2O_j is not initiated if it disrupts a currently active display on H2O_k. At the same time, H2O_k may employ a different set of nodes to route its required data in order to free up resources to enable H2O_j to be admitted (by satisfying its h requirement). Design of admission control, dynamic route scheduling, and their impact on h is a short term research direction.

Another research direction is to extend our designs to adjust the placement of data when a user requests a clip, similar in spirit to path replication [16]. This would enable the H2O cloud to respond to user actions such as shutdown of a H2O device, removal of a H2O device from a household, etc. The basic idea is as follows. When a H2O device displays a clip, the network is flooded with requests for the blocks of that clip. As a request walks from one H2O device to another, it increases a counter W. When a H2O device replies with a block b_i to a request with W_i walks, it includes W_i and its identity as a part of its reply. If W_i exceeds H_i, as the block is routed back to the requesting H2O device, the node that is H_i blocks away from this device stores a copy of b_i. Otherwise, no additional copies are constructed. This must be extended with an approach that deletes extra copies of a block and prevents a ping-pong behavior where different nodes construct and delete blocks repeatedly.

Finally, it is important to extend this study with models that control the number of block replicas based on the bandwidth available between two H2O devices. These models construct additional replicas of a block when bandwidth is low, degenerating into full clip replication in extreme situations.

In this paper, we have quantified the impact of different H2O topologies on the storage space requirement of our proposed technique. For the typical cases, graph and grid, we showed that it provides significant savings (several orders of magnitude) in storage space when compared with full

replication. We proposed two distributed implementation of our technique, TIMER and ZONE. A simulation study of these techniques validates the estimations provided by the analytical model. Thus our partial replication technique is a crucial data placement solution for efficient storage, low startup latency and hiccup-free reception of continuous media in wireless peer-to-peer networks.

ACKNOWLEDGMENT

This research was supported in part by an unrestricted cash gift from Microsoft research. We wish to thank the anonymous referees for their valuable comments.

REFERENCES

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems. In *ICMCS*, pages 118–126, 1996.

[2] J. Almeida, D. Eager, and M. Vernon. A Hybrid Caching Strategy for Streaming Media Files. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking*, 2001.

[3] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Straggered striping in multimedia information systems. In *Proceedings of the ACM SIGMOD International Conference on Database Management*, 1994.

[4] H.J. Chen and T. Little. Physical Storage Organizations for Time-Dependent Multimedia Data. In *Proceedings of the Foundations of Data Organization and Algorithms (FODO) Conference*, October 1993.

[5] E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. In *Proceedings of the ACM SIGCOMM*, August 2002.

[6] D. Eager, M. Ferris, and M. Vernon. Optimized Regional Caching for On-Demand Data Delivery. In *Proceedings of MMCN*, 1999.

[7] J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe. Multimedia Storage Servers: A Tutorial. *IEEE Computer*, 28(5):40–49, 1995.

[8] S. Ghandeharizadeh. Cable Networks and Distributed Video Repositories. In *Proceedings of the Cable 1997*, March 1997.

[9] S. Ghandeharizadeh. H2O Clouds: Issues, Challenges and Solutions. In *Fourth IEEE Pacific-Rim Conference on Multimedia*, December 2003.

[10] S. Ghandeharizadeh, A. Dashti, and C. Shahabi. Pipelining Mechanism to Minimize the Latency Time in Hierarchical Multimedia Storage Managers. *Computer Communications*, 18(3):38–45, March 1995.

[11] S. Ghandeharizadeh and T. Helmi. An Evaluation of Alternative Continuous Media Replication Techniques in Wireless Peer-to-Peer Networks. In *Third International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE, in conjunction with MobiCom'03)*, September 2003.

[12] S. Ghandeharizadeh and R. Muntz. Design and Implementation of Scalable Continuous Media Servers. *Parallel Computing*, 24(1):91–122, May 1998.

[13] Y. Guo, S. Sen, and D. Towsley. Prefix Caching Assisted Periodic Broadcast: Framework and Techniques to Support Streaming for Popular Videos. In *Proceedings of ICC*, 2002.

[14] K. A. Hua and S. Sheu. Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems. In *SIGCOMM*, pages 89–100, 1997.

[15] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of Ad Hoc wireless networks. In *Mobile Computing and Networking*, pages 61–69, 2001.

[16] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of the 16th annual ACM International Conference on Supercomputing*, 2002.

[17] N. Lynch. *Distributed Algorithms*, chapter 4. Morgan Kaufmann Publishers, San Mateo, CA, 1996.

[18] R. Ng and J. Yang. Maximizing Buffer and Disk Utilization for News On-Demand. In *Proceedings of Proceedings of the International Conference on Very Large Databases*, 1994.

[19] J. Nussbaumer, B. V. Patel, F. Schaffa, and J. P. G. Sterbenz. Networking Requirements for Interactive Video on Demand. *IEEE Journal of Selected Areas in Communications*, 13(5):779–787, 1995.

[20] V.G. Polimenis. The Design of a File System that Supports Multimedia. Technical Report TR-91-020, ICSI, 1991.

[21] P. Rodriguez and E. W. Biersack. Dynamic Parallel Access to Replicated Content in the Internet. *IEEE/ACM Transactions on Networking*, 10(4):455–465, August 2002.

[22] S. Sen, J. Rexford, and D.Towsley. Proxy Prefix Caching for Multimedia Streams. In *Proceedings of IEEE INFOCOM*, 1999.

[23] C. Shahabi and F. Banaei-Kashani. Decentralized Resource Management for a Distributed Continuous Media Server. *IEEE Transactions on Parallel and Distributed Systems*, 13(6):455–465, June 2002.

[24] R. Tewari, H. N. Vin, A. Dan, and D. Sitaram. Resource Based Caching for Web Servers. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking*, 1998.

[25] F.A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID-A Disk Array Management System for Video Files. In *First ACM Conference on Multimedia*, August 1993.

[26] S. Viswanathan and T. Imielinski. Pyramid Broadcasting for Video on Demand Service. In *SPIE Multimedia Computing and Networking Conference, Volume 2417*, pages 66–77, February 1995.

[27] B. Wang, S. Sen, M. Adler, and D. Towsley. Optimal Proxy Cache Allocation for Efficient Sreaming Media Distribution. In *Proceedings of IEEE INFOCOM*, 2002.

[28] Y. Wang, Z. Zhang, D. Du, and D. Su. A Network Conscious Approach to End-to-End Video Delivery over Wide Area Networks Using Proxy Servers. In *Proceedings of IEEE INFOCOM*, 1998.

Shahram Ghandeharizadeh received his Ph.D. degree in Computer Science from the University of Wisconsin, Madison, in 1990. Since then, he has been on the faculty at the University of Southern California. During the 1990s, he led a team of graduate students to design and implement Mitra, a scalable continuous media server. Mitra was a pioneering system that supported the display of clips with both a low and a high bandwidth requirement. In 1997, Panasonic licensed Mitra for research and development purposes. H2O clouds are a natural extension of Mitra to a peer-to-peer network of wireless devices.

Bhaskar Krishnamachari obtained his B.E. in Electrical Engineering from The Cooper Union for the Advancement of Science and Art with a four-year scholarship in 1998, and his M.S. and Ph.D. in Electrical Engineering from Cornell University in 1999 and 2002 respectively with a full graduate fellowship. Dr. Krishnamachari is currently an Assistant Professor in the Department of Electrical Engineering-Systems at the University of Southern California, with a joint appointment in Computer Science. His ongoing research is focused on developing mathematical models and algorithms for self-configuration and data acquisition/delivery in distributed wireless networks. He is a member of the IEEE and ACM and the Tau Beta Pi and Eta Kappa Nu honor societies.

Shanshan Song received her BS degree in Computer Science from Special Class for Gifted Young in University of Science and Technology of China in 2001, and MS degree in Computer Science from University of Southern California (USC) in 2003. She is currently pursuing Ph.D. degree in Department of Computer Science at USC. Her research insterest lies primarily in the area of network security and dynamic resource allocation for computional Grids. She can be reached at shanshas@usc.edu.