Cooperative Caching Techniques for Continuous Media in Wireless Home Networks*

Shahram Ghandeharizadeh and Shahin Shayandeh
Computer Science Department
University of Southern California
Los Angeles, California 90089-0781
shahram@usc.edu, shayande@usc.edu

November 13, 2008

Abstract

With wide spread deployment of wireless home networks, management of data across devices is becoming increasingly important. This is especially true for continuous media (audio and video clips) because they are large in size and are streamed at a pre-specified rate to support a display free from disruptions and delays. Caching of clips across devices is an effective way to improve key quality of service (QoS) metrics including the fraction of requests serviced successfully when the home's connection to the outside infrastructure is lost (data availability), number of devices that may stream and display their referenced clips simultaneously (throughput), and the average delay incurred from when a user references a clip to the onset of its display (average startup latency). In this paper, we focus on home networks consisting of a handful of devices and present a novel cooperative caching technique named Cont-Coop. Cont-Coop controls the content of participating caches based on the asymmetric bandwidth of wireless connections between devices. We compare this technique with an alternative that does not control the content of cooperative caches, showing Cont-Coop is superior when the access pattern to clips is skewed. In addition, we show cooperative techniques enhance all the aforementioned QoS metrics when compared with a greedy caching technique.

A Introduction

Wireless home networks have become pervasive and widely deployed due to their low cost and ease of installation. Today, a typical home network consists of an access point, several PCs, and one or more consumer electronic devices [27]. It is anticipated that future home networks provide multimedia streaming. An example might be a household with several PCs and TVs. Each TV might include a set-top box, e.g., Apple TV, that synchronizes with a PC using its wireless networking card to stream audio and video clips. Devices might be configured with a mass storage device[1] and set aside a fraction of their storage to cache content.

While different members of a household may exhibit different preferences for different clips, the system may build a profile of those clips referenced frequently [2]. Ideally, these clips should occupy the caches in order to minimize the number of references to remote servers. A device may

*This research was supported in part by NSF grant IIS-0307908. A shorter version of this paper appeared in the First ACM International Conference on Ambient Media and Systems, Quebec City, Canada, February 11-14, 2008.

[1]Apple TV is configured with 40 gigabytes (GB) of disk storage.

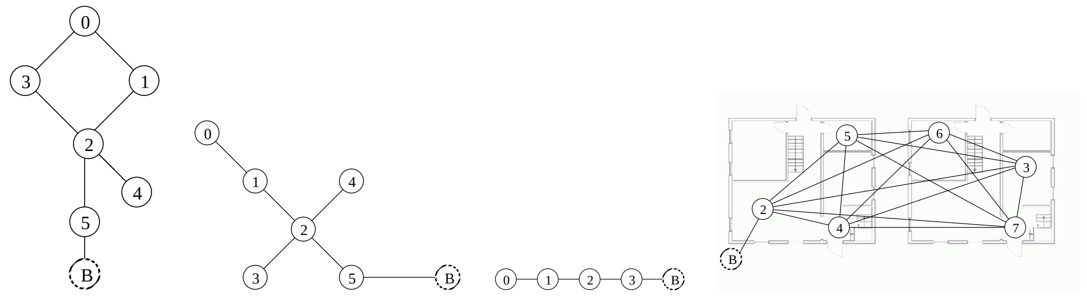

1.a Kyte 1.b Dagger 1.c String 1.d Real deployment

Figure 1: Alternative topologies.

observe two kinds of cache hits. First, a local cache hit where a device finds its referenced clip in its local storage. Second, a cooperative cache hit where a device locates its referenced clip in the storage of another device in the household. Both minimize round-trip delays of messages to remote servers. Moreover, they improve availability of clips when the home network is disconnected from the infrastructure outside the home. In this study, we assume a base station or a gateway [2] as intermediary between the wireless home network and the outside infrastructure. This is denoted as B in topologies of Figure 1.

We assume the link bandwidth of wireless connections between devices is asymmetric and devices exchange data using ad hoc communication. This is based on a recent study [27] that analyzed deployment of six wireless devices in different homes in United States and England, demonstrating asymmetric bandwidth of wireless connections. For example, with the British household of Figure 1.d where each device is configured with an 802.11a networking card, the bandwidth from device 3 to device 7 is 9 Mbps while the reverse bandwidth is 1 Mbps. Ad hoc communication provides a higher bandwidth when compared with a deployment that employs an access point because it avoids the use of low bandwidth connections [27]. For example, if node 7 is the access point, all communications from other devices to device 3 would observe a bandwidth of 1 Mbps[2]. With ad hoc communication, a device may communicate with those devices in its radio range directly. Moreover, a device may send its data to a destination device by using other devices as routers. In our example, this enables device 7 to stream a clip to device 3 at bandwidths higher than 1 Mbps by using device 6 (or 5) as an intermediary to route its data.

A key question in this environment is how to cache clips across devices. In [15], we showed greedy caching techniques such as GreedyDual [5] and Dynamic Simple [14] provide a higher cache hit rate than LRU because they consider the size of clips. (See Table 1 for definition of cache hit rate.) In this study, we consider cooperative caching techniques where multiple (say $\mathcal{N}$) devices form a group and synchronize the state of their caches [21, 19, 2] to enhance a global metric such as the average startup latency. Assuming the middleware of [19] which enables a device to join and leave a cooperative group, the alternative caching techniques can be taxonomized along two dimensions. Each dimension corresponds to a design question that produces a hierarchy of techniques. The two design questions are:

1. Does a technique control the identity of clips cached across the participating devices based on the asymmetric bandwidth of their wireless connections?

[2]One may place the access point strategically so that it provides a high bandwidth to each device. Changes in the environment, e.g., new furniture, may reduce the bandwidth of a connection from the access point to one or more devices. Ad hoc communication reacts to such changes dynamically.

Term	Definition
Core node	A node whose cache state is independent of other nodes.
Startup latency (δ)	Delay from when a user references a clip to the onset of display.
Throughput	Number of devices displaying their referenced clips simultaneously.
Cache hit rate	The percentage of clip requests satisfied using the cache.
Byte hit rate	The number of bytes satisfied from the cache as a fraction of the total bytes referenced by the client.

Table 1: Terms and their definitions

2. When a device displays a clip, does it cache the referenced clip?

Below, we consider each question in turn.

In response to the first question, we present two alternative techniques: Random and Controlled, named Rand-Coop and Cont-Coop, respectively. Rand-Coop does not control the identity of clips occupying caches. Cont-Coop controls the content of caches by choosing one device, named the core node, to contain the frequently accessed clips. This device is chosen strategically to minimize the possibility of bottleneck links when multiple devices observe cache misses and stream their referenced clips, enhancing startup latency. The choice of a core node is not obvious. For example, assuming the same bandwidth for all links and considering the Dagger and Kyte topologies of Figure 1, one may anticipate device 2 to be the core node for each topology. In reality, devices 1 and 0 should be the core node for the Dagger and Kyte topologies, respectively. The algorithm to choose the core node is one novel aspect of our study.

Cont-Coop may result in the mistreatment[3] phenomena of [21]. With this phenomena, the core node may appear to have 'hijacked' the cache of other devices participating in a cooperative group by reducing their cache hit rate while maximizing its own cache hit rate. At the same time Cont-Coop chooses the core node strategically in order to enhance the average startup latency of the entire cooperative group.

In response to the second design question, Rand-Coop and Cont-Coop might be configured with one of the following two policies when a device displays a clip: Either transient copy (T-Copy) or one copy (1-Copy). With T-Copy, when a device N_i streams a clip X from another device (or a remote server outside of the home), N_i caches X in anticipation of future correlated references for X. With 1-Copy, if a copy of X resides in the cache of a device in the cooperative group, N_i streams X and does not cache X. When compared with one another, T-Copy provides a lower availability of data while enhancing startup latency when a few clips are popular and accessed frequently, see Section D.

Contributions of this study are two folds. First, with Cont-Coop, one device is elected as a core node to cache the popular clips and minimize link bandwidth contention caused by cache misses of other devices. This novel design decision improves average startup latency of the cooperative group significantly. Second, we compare cooperative caching with greedy caching, demonstrating that it enhances startup latency of a streaming environment with link bandwidths higher than the bandwidth required to display a clip. Greedy caching is inferior even though it observes a higher cache and byte hit rates (30% higher in some instances) because its cache misses observe a huge startup latency.

The rest of this paper is organized as follows. Section B surveys the related work. Section C presents two cooperative caching techniques (Rand-Coop, Cont-Coop) and policies for managing the

[3]This phenomena may also occur with Rand-Coop. However, it is more likely with Cont-Coop.

cache when a device references a clip (1-Copy and T-Copy). Section D compares these techniques with one another. Brief conclusions and future research directions are presented in Section E.

B Related work

Cooperative caching in home networks is based on established techniques used for improving the performance of distributed networked systems. Our focus is on online algorithms that assume no advance knowledge of future requests. Such techniques have been shown to improve the performance of: 1) home networks [19] and a neighborhood of home gateways [2], 2) multiprocessor systems with uniform and non-uniform access to shared memory [22, 25], 3) file and virtual memory systems in a high speed, local area network [13], 4) proxy cache servers deployed across a few organizations [36], 5) Content Distribution Networks (CDNs) for streaming media [28, 1, 6, 17], 6) peer-to-peer (P2P) networks of devices [8, 33], and 7) ad hoc networks of devices that consider either mobility [18, 4], energy constraints [31], or both [26]. Due to vastness of each topic and strict page limits on this submission, listed references are samples and not intended to be exhaustive. In a nut shell, Cont-Coop is novel for two reasons. First, it is designed to enhance average startup latency of a wireless home network. Second, it elects one device as a core node to minimize bandwidth contention caused by cache misses of other devices, enhancing average startup latency significantly. Below, we describe the key differences with prior studies in turn.

Techniques similar to Rand-Coop configured with T-Copy are presented in [19, 2]. While [19] presents cooperative caching for home networks, [2] focuses on cooperative caching for home gateways (termed base stations in Figure 1) in a neighborhood. Both studies assumes LRU as the caching technique employed by a node. The general purpose framework of [19] presents general techniques for nodes to join and leave a cooperative group, stream clips, and prefetch data. In [2], Rand-Coop configured with T-Copy is compared with Internet Cache Protocol (ICP) and the Cache Array Routing Protocol (CARP) to show it provides a lower cost. Neither study presents a caching technique that considers the asymmetric bandwidth of network connections between devices. Moreover, the QoS metrics considered by these earlier studies are simpler than those presented here.

Cooperative caching for multiprocessors with shared memory and virtual file systems must address coherence of data blocks with updates. In our target environment, clips are updated rarely. Most often, these updates modify metadata of a clip such as its DRM data. To support such periodical updates, one may utilize designs similar to those employed by proxy cache servers: when a cache adds a new clip, it calculates a Time-to-Live (TTL) for the clip and releases the clip when its TTL expires [7]. Another possible design is to employ *If-Modified-Since* [35] requests to the remote server that published the clip to determine if the cached clip should be refreshed.

Cooperative caching techniques for proxy web caches strive to maximize the number of requests serviced using web caches. Two key metrics are cache hit rate and byte hit rate, see Table 1. Our techniques are different because they enhance startup latency of streaming media. At the same time, we employ the search techniques proposed for proxy caches to locate a cached clip in the cooperative group. These include hierarchical [7, 30, 35], directory-based [29, 12, 32], hash-based [20, 34], and multicast-based schemes [24, 23].

Content Distribution Networks (CDNs) for streaming media [28, 1, 6, 17] cache partial or entire clips at proxies deployed close to clients to reduce network and server load and enhance system performance. They propose novel placement and replacement strategies for proxy cache servers that are independent of both the original servers that publish clips and the clients that reference them. Streaming CDNs are a key component of the infrastructure deployed outside a home, providing service to many homes in a neighborhood. The techniques employed by CDNs are not appropriate for each individual nodes in the home network because the cache of each node is autonomous. Clip references issued by a node dictate the content of its cache. Moreover, the topology of the wireless network, link bandwidth and location of the base station relative to nodes of the home network dictate how Cont-Coop chooses the core node to minimize formation of bottlenecks links in the home network.

For P2P network of devices, cooperative caching has been used to both search [8] and to enhance download time [33]. Search is complementary to our study because our focus is on delivery of a streaming clip. Hence, we focus on [33] which describes a replication technique that considers the client and server dilemma in a P2P network: While the client wants to obtain its referenced data item as quickly as possible, the server wants to service no more than its fair share of the total workload. This study shows if the workload is distributed fairly then one also minimizes the average download time when the delay at the server is convex in the utilization factor of the server. It also shows replicating files proportional to their request rate ensures fairness and enhances download time. Key assumptions of [33] include: 1) each node provides the same bandwidth, and 2) all files are equi-sized. Our assumed environment violates both assumptions. Another key difference is our focus on bandwidth contention instead of the client and server dilemma of P2P networks.

Finally, devices in our environment are neither mobile nor energy constrained as assumed in [18, 4, 31]. This differentiates design decisions of these studies from Cont-Coop[4].

C Cooperative caching

This section describes two techniques that employ message passing to synchronize the state of $\mathcal{N}$ nodes that participate in a cooperative group. Below, we describe how a node synchronizes the state of its cache with $\mathcal{N}-1$ other nodes while preserving its autonomy. Subsequently, we introduce Rand-Coop, detail Cont-Coop and its design decisions, and 1-Copy and T-Copy policies.

With both Rand-Coop and Cont-Coop, different events may cause a node to synchronize its cache state with those of other nodes: upon caching a clip, when choosing a victim, or both. In this paper, we focus on '*choosing a victim clip*' as the event to synchronize a cache state. With this design, when a node N_i references a clip X that does not reside in the cooperative group, N_i may cache X. If N_i has insufficient space for X, N_i chooses a victim clip by performing the following steps. First, it contacts other nodes in the cooperative group for a list of clips occupying their caches. Using these, it computes a list of those clips in its cache with one or more replicas in the cooperative group. It sorts these clips in descending order of their $\frac{f_i}{S_i}$ values (where S_i and f_i are the size and frequency of access to clip i, respectively), and swaps out as many clips as necessary to accommodate incoming clip X. If there is still insufficient free space then it invokes its greedy cache management technique to free its cache space to accommodate X. A greedy cache management technique might be either GreedyDual [5] or DYNSimple [14] designed to support variable sized clips [15].

One may improve upon the above protocol in a variety of ways. For example, N_i may cache the list of clips stored at other nodes and refresh this list after some elapsed time v [2]. Another possible improvement is for N_i to request only changes to the state of N_j's cache since the last time N_j provided its list of cached clips. This minimizes the amount of data exchanged to synchronize the cache state of participating nodes. Similar techniques are studied in [2]. Key tradeoffs are between cache hit rates and the overhead of exchanging metadata among the nodes. A future research direction is to characterize the impact of this overhead on the startup latency and throughput of a home network.

[4]While Section C.2.3 considers possible movement of nodes, they are assumed to be infrequent.

The above framework preserves autonomy of a node N_i because, when no cooperating device is in N_i's radio range, N_i employs a greedy caching technique and its functionality is not impaired. N_i may join and leave a cooperative group using the middleware proposed in [19]. Note that we assume nodes in a home network are cooperative. An adversarial node would degrade the performance of both the home network and itself because bandwidth contentions impact all nodes.

C.1 Rand-Coop

The design of Rand-Coop is simple: Each node synchronizes its cache with every other node, causing the $\mathcal{N}$ caches to behave as if they are one. The placement of clips across devices is controlled by how different devices issue requests for different clips.

C.2 Cont-Coop

Cont-Coop selects one node as the core node. The core node manages its cache state using a greedy strategy and does not coordinate with the other nodes. Other nodes synchronize the state of their caches with one another and the core node. As detailed in Section C.2.1, the core node is chosen strategically to maximize utilization of all wireless connections that constitute the home network. We assume this will reduce likelihood of bottleneck links which in turn enhances both the throughput and startup latency of the system. This hypothesis is validated in Section D.2.

Below, we describe how Cont-Coop chooses a core node. Subsequently, Section C.2.2 extends this algorithm to a distributed environment. Finally, Section C.2.3 discusses evolving network topologies due to movement of devices. These sections use topologies of Figure 1 for illustration purposes. In this figure, vertices represent nodes. The vertex denoted B represents the base defined as the interface between the wireless home network and the infrastructure outside of the home. An edge from N_i to node N_j represents two wireless links. One link from N_i to N_j, (i,j), and a second link from N_j to N_i, (j,i). Each link has a bandwidth which is not shown in the figures.

C.2.1 Selection of core node

While the bandwidth of individual links in a network may exceed the bandwidth required to stream and display a clip, when all devices stream different clips, they may exhaust the bandwidth of one or more links in the network. These links are called bottleneck links. Cont-Coop chooses a core node strategically to prevent formation of bottleneck links.

Figure 2 shows the pseudo-code to select a core node. It minimizes the bandwidth contention caused by a cache miss that must utilize the base. This is realized by choosing a node N_i that minimizes the standard deviation in the weight imposed on the different links of the network. Weight of a link is the product of 1) the amount of traffic imposed on that link, and 2) the ratio of the link bandwidth and the maximum link bandwidth in the network. The first parameter quantifies the number of possible paths to N_i and the base. The second parameters captures networks where different links provide different bandwidths. Below, we elaborate on the pseudo-code and use the network topology of Figure 1.a to illustrate its execution. We start by assuming all network links have the same bandwidth. Subsequently, we extend the discussion to consider scenarios where links offer varying bandwidths.

The pseudo-code of Figure 2 considers the possibility of each node as the core exhaustively. It consists of several key components. First, for each node N_i as a candidate core, it considers every other node N_j and enumerates the number of *shortest* paths from N_i to N_j and N_j to the base. The total number of such paths is denoted P. Second, it sums the number of paths that utilize a link L_i to compute S_i. Next, it assigns a weight of $\frac{S_i}{P}$ to each link L_i assuming link bandwidth is

```
Core node selection( )
{
 Let B denote the base;
 CSL = Empty list;
 $L_{Max}$ = Link with the highest bandwidth;
 Pick a node $N_i$ as the core node
 {
   Initialize the weight of all links in the network to zero;
   For each remaining node $N_j$ do
   {
     i) Enumerate all possible shortest paths from $N_j$ to $N_i$ that does not involve the base station;
     ii) Enumerate all possible shortest paths from $N_j$ to $B$;
     iii) Let P be the set of paths identified in Steps i and ii;
     iv) For each link $L_i$ in the network do
     {
          Let $S_i$ denote the number of paths in P that include $L_i$;
          Increment link weight of $L_i$ by $\frac{S_i}{P} \times \frac{L_{max}}{Bandwidth\ of\ L_i}$;
     }
   }
   Std = standard deviation in the link weight;
   Insert $(N_i, Std)$ in CSL;
 }
 $N_{Core}$ = Node corresponding to the CSL entry with the minimum standard deviation value;
}
```

Figure 2: Pseudo-code to select a core node.

the same for all nodes (below, we remove this assumption). It chooses the node which minimizes the standard deviation in link weights as the core node.

To illustrate, consider the Kyte topology of Figure 1.a. The core selection starts by choosing N_0 as the core. Next, it picks N_1 as a client and enumerates the possible paths to N_0. They include a direct path (0,1) and a multi-hop path {(0,3), (3,2), (2,1)}. There is also one path from the base station B to N_1. The pseudo-code of Figure 2 chooses the shortest path from N_0, eliminating the multi-hop path. With two paths (P=2), we increment the weight of each participating link by 0.5. The impacted links are (5,2), (2,1), and (0,1). We repeat this process with nodes N_3, N_2, N_4, and N_5.

If one assumes the bandwidth from the base station to N_5 is very high (fiber to the home) then N_5 would be serviced from the base station at all times, ignoring its paths to N_0. In this case the value of P is 1 for N_5 and weight of link (b,5) is also one.

Assuming identical bandwidth for different links, the pseudo-code of Figure 2 selects N_0 as the core node because it minimizes the likelihood of bandwidth congestion (balanced number of paths across links).

When the link bandwidths are asymmetric, we normalize a link L_i's weight by the ratio of 1) the link with the highest bandwidth (L_{max}), and 2) L_i's bandwidth. This enables Cont-Coop to utilize those links with the highest bandwidth. This is realized by increasing the weight of links with a lower bandwidth, resulting in a higher standard deviation across the link weights. This is desirable because it prevents selection of a core node that increases the usage of low bandwidth links. To illustrate, consider the String topology of Figure 1.c. When the bandwidth for all links is identical, Cont-Coop will choose N_1 as the core node. If the bandwidth from N_2 to its neighbors is asymmetric then Cont-Coop may select N_2 as the core node. For example, assume the bandwidth between all nodes except for N_2 and N_1 is symmetric and set to 8 Mbps. When the bandwidth from N_1 to N_2 is 1 Mbps and from N_2 to N_1 is 14 Mbps, it is better to choose N_2 as the core node. Intuitively N_2 is a better choice because its link bandwidth enables it to minimize startup latency when servicing N_0 and N_1. If N_1 was the core node then its out-going link bandwidth (1 Mbps) would stream a clip to node N_2 by incurring a high startup latency. With the pseudo-code of Figure 2, the standard deviation when choosing N_2 as the core node is 0.97. It is 2.73 with N_1 as the core node. Thus, N_2 is selected as the core node.

C.2.2 Decentralized selection of core node

With a network consisting of a handful of devices, a decentralized implementation of the core node selection technique might be as follows. Periodically, a node exchanges information about its link bandwidth with every other node in the network. Each node builds a topology of the network and invokes the pseudo-code of Figure 2 independently to select a core node. Subsequently, each node casts a vote for its computed core node by broadcasting a message to every other node. The node that receives the most number of votes is elected as the core node. In case of ties, Cont-Coop might be configured with a variety of tie breaker techniques such as choosing the node with either most neighbors, highest amount of bandwidth, or highest networking card (MAC) address.

There may exist two extreme transient states: 1) every node elects itself as the core node, and 2) there exists no core node. With the first, each node operates independent of the other nodes using its local greedy caching technique. With the second, Cont-Coop becomes Rand-Coop. Periodic invocation of the decentralized technique by the different nodes will resolve these transient states.

C.2.3 Evolving network topologies

Members of a household may physically move nodes of the wireless network and change its topology. A node may detect such changes by monitoring the identity of its neighboring nodes and its wireless network bandwidth to these nodes. Moreover, nodes may build a profile of how frequently such changes occur and the average duration of a stable network topology. Once a node detects a change in network topology, it may initiate either the centralized or distributed implementation of core node selection. This may either re-affirm the existing core node or elect a new node as the core node. If the network topology changes too frequently, the network may not stabilize. In such cases, it is best to not use Cont-Coop. A poor choice of node as the core node impacts startup latency adversely and may cause Cont-Coop to perform worse that Rand-Coop.

C.3 One copy, 1-Copy

With the 1-Copy cache state management policy, when node N_i references clip X, it does not cache X if a copy of it exists in the cooperative group. It only streams X and frees sufficient space to maintain a portion of X in support of a display free from disruptions and delays.

C.4 Transient copy, T-Copy

T-Copy requires a node N_i that references clip X to materialize X in its cache always. This enables N_i to service future correlated references for clip X using its local copy. When there exist a few popular clips that are referenced by all nodes frequently then either most or all nodes may cache these clips with T-Copy. These nodes observe cache hits for these clips as long as they are cache resident. These clips might be deleted when N_i decides to cache a new clip and has insufficient free space. This is a rare event when the distribution of access to clips is skewed. Thus, it does not impact average startup latency. However, with a more uniform distribution of access, T-Copy may result in a higher startup latency when compared with 1-Copy.

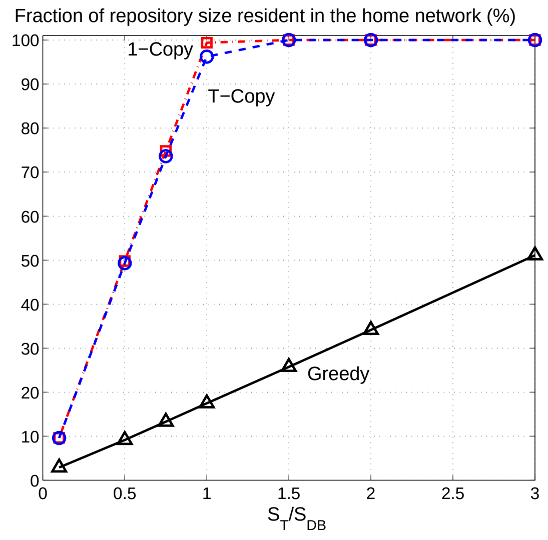
In general, T-Copy results in lower availability of clips when compared with 1-Copy for skewed distribution of access across clips, see Section D.1. This is because T-Copy replicates popular clips across multiple nodes, reducing the number of unique clips cached in the cooperative group.

D Comparison

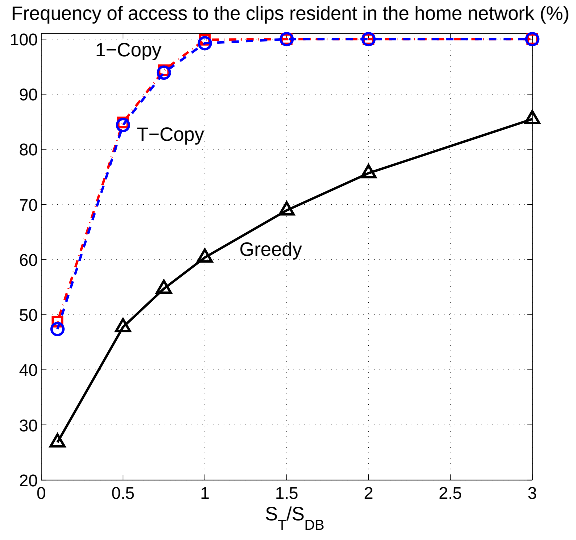
We use a simulation study to a) evaluate Rand-Coop and Cont-Coop when configured with 1-Copy and T-Copy, and b) compare cooperative caching with a greedy caching strategy. The simulator represents each device as a node. A node is configured with a fixed amount of storage and sets aside a portion of it as cache. While S_T denotes the sum of the size of caches contributed by $\mathcal{N}$ nodes in a cooperative group, S_{DB} is the size of the clip repository. A node may choose amongst different replacement techniques to control which clips occupy its cache. In this study, we use Dynamic Simple (DYNSimple) because it is superior [15] to alternatives such as GreedyDual [5] and LRU. With DYNSimple, a node chooses a victim by estimating the frequency of access to each clip i (f_i), and replacing the clip with the smallest value for $\frac{f_i}{S_i}$ where S_i is the size of clip i.

An edge from node N_i to node N_j means N_j is in the radio range of N_i to receive data from it. There must exist an edge from N_j to N_i in order for N_i to receive data from N_j. Each edge is assigned a pre-specified bandwidth, providing the asymmetric transmission rates between nodes as described in [27].

Node N_i may have g edges with different bandwidths to g different nodes in the system. We consider two different transmission models. The first, termed kp-card, assumes the bandwidths



3.a) %DB



3.b) %FreqDB

Figure 3: Availability of data in home network as a function of $\frac{S_T}{S_{DB}}$, topology is Dagger, $\mu = 0.73$.

correspond to the transmission rates observed by N_i's k cards using their p channels to communicate with the g nodes. In essence, N_i's total out-going bandwidth is the sum of the bandwidths specified on its out-going edges. The second, termed Shared-BW, assumes the bandwidths are shared and N_i may not transmit at the full rates specified on each of its out-going edges simultaneously. As an example, assume N_i has two out-going edges to nodes N_j and N_k with bandwidths of 10 and 12 Mbps, respectively. With the kp-card model, N_i may transmit to N_j at a rate of 10 Mbps while transmitting to N_k at a rate of 12 Mbps simultaneously. With the Shared-BW model, if N_i transmits data to N_j at a rate of 10 Mbps then it may transmit data to N_k at a rate of 2 Mbps only. With both models, N_i's minimum data rate is 10 Mbps. However, N_i's maximum data rate is 22 Mbps and 12 Mbps with kp-card and Shared-BW, respectively. This study assumes Shared-BW transmission model.

We assume a repository of video clips where each clip is encoded using a constant bit rate encoding technique. The display bandwidth requirement of each clip is 4 Mbps. We assume the repository consists of 864 clips grouped into 3 categories with display times of 2 hours, 60 minutes, and 30 minutes. The size of these clips are 3.5 Gigabytes (GB), 1.8 GB, and 0.9 GB, respectively. S_{DB} is 1.85 Terabytes.

We use a Zipf-like distribution [3] to generate requests for different clips. To elaborate, Zipf's law [37] defines the relative popularity of a request for the i'th most popular clip is proportional to $\frac{\chi}{i}$ where χ is a normalizing constant. Different studies provide different definitions for this law. Assuming C clips are rank ordered based on their popularity (1, 2, ..., C), a general definition named Zipf-like distribution is as follows. The probability of access for clip i is: $\frac{\frac{1}{i^\mu}}{\chi}$. The exponent μ $(0 \leq \mu \leq 1)$ controls the mean of the distribution and $\chi = \frac{1}{\sum_{i=1}^{C} \frac{1}{i^\mu}}$. This law has been used to model the distribution of web page requests [16, 3, 36], and sale of movie tickets[5] in the United States [10].

[5]In [10], a Zipf-like distribution is defined as $\frac{\chi}{i^{(1-\omega)}}$ where ω is 0.27. In this paper, μ equals $1-\omega$. To be consistent with [10], we analyze 0.73 as a possible value for μ in Section D.

With a Zipf-like distribution, a larger value for exponent μ makes the distribution more skewed by increasing the frequency of access to a few popular clips. On the other hand, a smaller value of μ makes the distribution more uniform.

One node in the system is designated to admit requests in the network by reserving link bandwidth on behalf of a stream. This node, denoted N_{admit}, implements the Ford-Fulkerson algorithm [9] to reserve link bandwidths. It is configured with the policy to minimize startup latency.

The simulator conducts rounds that are repeated tens of thousands of times. A round selects nodes one at a time. The order in which nodes are selected is shifted by one in each iteration, ensuring that every node has a chance to be the first to stream a clip in the network. A node (say N_1) references a clip using a random number generator conditioned by the assumed Zipf-like distribution. If this clip resides in N_i's local storage then its display incurs a zero startup latency. Otherwise, N_i identifies those nodes containing its referenced clips, termed candidate servers. Next, it contacts N_{admit} to reserve a path from one of the candidate servers. N_{admit} provides N_1 with the amount of reserved bandwidth, the paths it must utilize, and how long it must wait prior to streaming the clip. This delay is the incurred startup latency.

In each iteration, we measure the following parameters for each node: incurred startup latency, cache hit rate, byte hit rate, average number of hops to stream the referenced clip, and the amount of data transferred across the network. Each experiment is based on ten thousand iteration of each round. At the end of an experiment, we measure the fraction of repository resident in the home network and its frequency of access. These characterize availability of data in the home network, see Section D.1.

One may introduce an arbitrary delay (termed a think time) between different iterations. However, the observed values will not change because: 1) all devices are activated in each round, and 2) we measure the average startup latency incurred in each round.

We investigated alternative topologies shown in Figure 1 with link bandwidths ranging from 8 to 16 Mbps. Presented observations hold true for all these settings. Unless stated otherwise, we employ a Dagger topology with symmetric link bandwidths of 12 Mbps in Sections D.1 to D.3. Section D.4 considers a realistic network corresponding to a deployment of six nodes employing 802.11a networking cards in a British household [27]. The link bandwidths are asymmetric, see Figure 1 for their bandwidths.

D.1 Availability of clips: Greedy versus Cooperative

One may quantify availability of data using two different metrics: 1) fraction of repository size resident in the home network, termed %DB, and 2) frequency of access to this fraction, termed %FreqDB. A higher value for the first metric (%DB) implies higher data availability because if the household loses its connection to the outside world, a larger selection of clips will be available in the home network. A higher value for the second metric (%FreqDB) implies a higher likelihood of access for clips resident in the home network. These two metrics do not necessary go hand in hand. For example, a naive technique may exhaust the available storage of participating devices with the least popular clips, maximizing the value of %DB. However, this will minimize the value of %FreqDB.

In our experiments, the cooperative techniques enhance both %DB and %FreqDB simultaneously, outperforming the greedy caching technique. Below, we present experimental results based on the Dagger topology, see Figure 1.a. Key observations hold true for other topologies.

Figure 3.a shows the behavior of different techniques as a function of the available cache size of each participating node. The x-axis of this figure represents the ratio between total cache size (S_T) and the repository size (S_{DB}), i.e., $\frac{S_T}{S_{DB}}$. When the cache size of the cooperative group is comparable with the repository size ($\frac{S_T}{S_{DB}}$=1), both Rand-Coop and Cont-Coop render almost the entire repository cache resident, achieving a value of 99% for %DB. Greedy materializes almost 18% of the repository because the most popular clips are cached on every node. This minimizes the number of unique clips materialized across different devices.

At the same time, greedy materializes those clips with the highest frequency of access resident across the caches, see Figure 3.b. This results in a higher %FreqDB for greedy, reducing its relative difference with the cooperative techniques.

Figure 3 shows 1-Copy provides a slightly higher %DB when compared with T-Copy. We observed this trend in all our experiments. This is because T-Copy replicates the popular clips that are referenced by different nodes multiple times, reducing the number of unique clips in the network. The difference between 1-Copy and T-Copy becomes smaller with a more uniform distribution of access to clips, not shown in Figure 3.

D.2 Impact of core node selection

This section shows the core node selection of Section C.2.1 enhances startup latency when compared to choosing a different node as the core node. We focus on T-Copy state management policy for two reasons. First, trends observed with T-Copy and 1-Copy are identical. Second, Section D.3 shows T-Copy enhances response time when compared with 1-Copy.

With the Dagger topology of Figure 1.b, a system designer may choose either N_1 or N_2 as the core node. The pseudo-code of Figure 2 chooses N_1 because it balances the number of paths referencing different links more evenly. Figure 4 shows the factor of improvement in startup latency when core node is N_1 relative to when core node is N_2. The x-axis of this figure denotes different $\frac{S_T}{S_{DB}}$ values. We compute the y-axis value by dividing the average startup latency observed when N_2 is the core node by the startup latency observed when N_1 is the core node. When the y-axis value is 1, it means the two configurations are providing identical startup latencies. A y-axis value greater than 1 means choosing N_2 as the core node is a poor decision as it is degrading startup latency. Figure 4 shows the factor of improvement for two different distributions of access, skewed (μ=0.73) and more uniform (μ=0.25). Generally speaking, with a uniform access pattern, choosing the right core node is not important. We elaborate on this observation in Section D.4 when comparing Cont-Coop with Rand-Coop.

With a skewed distribution of access, it is better to choose N_1 as the core node because it prevents formation of bottlenecks. To illustrate, assume nodes N_0, N_1, N_3, and N_4 reference the most popular clip. With N_2 as the core, the request for these four nodes will be directed to N_2. N_2 must support a bandwidth of 16 Mbps to stream all requests simultaneously. With a bandwidth of 12 Mbps, one of the requests must wait for the other three requests. By choosing N_1 as the core node, with this example, N_1 will service three nodes and N_5 services the other node (either N_2, N_3, or N_4).

With the realistic topology of Figure 1.d, the selection algorithm of Figure 2 chooses N_5 as the core node. Figure 5 shows the factor of improvement in startup latency with N_5 as the core node when compared with node $i = \{2, 3, 4, 6, 7\}$ as the core node. Generally speaking, choosing N_5 enhances startup latency. N_2 is a very poor choice as the core node because all references for those clips that do not exist in the cooperative group (issued by N_3 to N_7) must be directed to N_2 and streamed from the base B. With a bandwidth of 11 Mbps from N_2 to other nodes, at most 3 such streams can be supported from B. Additional stream requests by other nodes must wait, resulting in a higher startup latency.

In Figures 4 and 5, as we increase $\frac{S_T}{S_{DB}}$ from 0.1 to 0.75, the startup latency with all technique improves. However, the rate of improvement with the configuration that employs the correct core

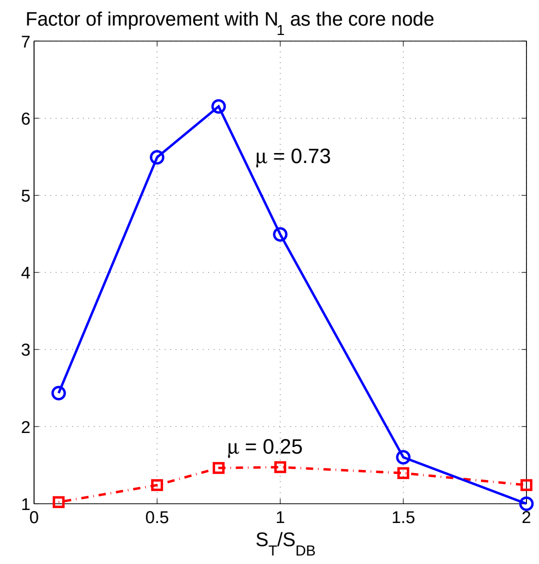


Figure 4: Factor of improvement in startup latency ($\frac{\delta(N_2)}{\delta(N_1)}$).

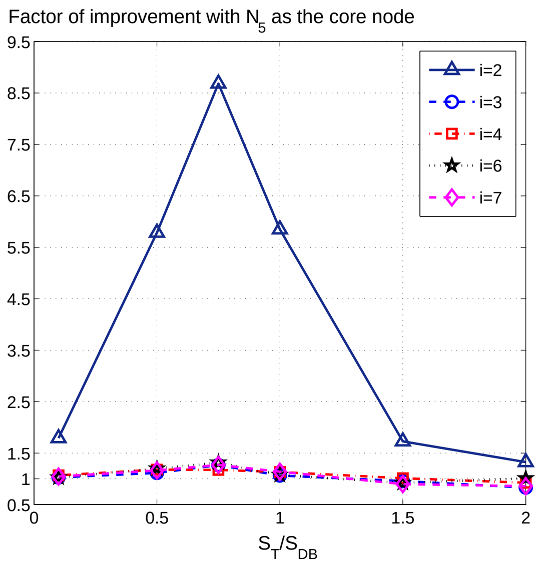


Figure 5: Factor of improvement in startup latency ($\frac{\delta(N_i)}{\delta(N_5)}$) with N_5 as the core node when compared with N_i ($i \neq 5$) as core node, μ=0.73.

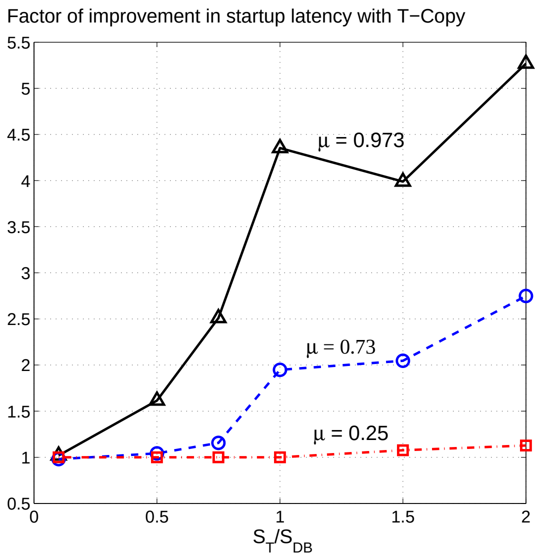


Figure 6: 1-Copy versus T-Copy ($\frac{\delta(1-Copy)}{\delta(T-Copy)}$), Dagger topology, Cont-Coop with N_1 as the core node.

node is much higher. This explains why the steep slope of the curve, peaking at $\frac{S_T}{S_{DB}}$=0.75. Beyond this $\frac{S_T}{S_{DB}}$ value, Cont-Coop with the correct core node starts to plateau while other configurations (with the wrong core node) continue to improve. This explains why the factor of improvement decreases.

D.3 1-Copy versus T-Copy

T-Copy results in a lower startup latency when compared with 1-Copy. This is particularly true with a skewed distribution of access (μ values of 0.73 and 0.973), see Figure 6. In this figure, the x-axis denotes different cache sizes, i.e, $\frac{S_T}{S_{DB}}$ values. The y-axis is computed by dividing the average startup latency observed with 1-Copy by T-Copy's startup latency, $\frac{\delta(1-Copy)}{\delta(T-Copy)}$. With y-axis values higher than 1, 1-Copy is worse than T-Copy because it is slower.

With a skewed distribution, T-Copy outperforms 1-Copy because it replicates the frequently accessed clips on all the nodes. While this results in lower availability of data (see Section D.1), it does increase the cache hit rate of each node. This is because with T-Copy, once a node references a popular clip, it replicates that clip and replaces it only when it encounters a cache miss. In between cache misses, every time it references the popular clip, it encounters a cache hit. When $\frac{S_T}{S_{DB}}$=1 and $\mu = 0.73$, T-Copy observes a cache hit rate of 37% while 1-Copy observes a cache hit rate of 20%. A higher cache hit rate translates into a higher byte hit rate. This reduces the amount of data transferred using the network, minimizing the the possibility of bottleneck links. This explains why T-Copy outperforms 1-Copy.

D.4 Comparison of greedy and Rand-Coop with Cont-Coop

In this section, we compare greedy and Rand-Coop with Cont-Coop using the topology of Figure 1.d corresponding to a deployment of six nodes in a British household [27]. With this wireless home network, Cont-Coop elects node N_5 as the core node.

$\frac{S_T}{S_{DB}}$	Greedy	Rand-Coop	Cont-Coop (N_5)
0.1	585	353	320
0.5	225	51	40
0.75	148	21	15
1.0	102	10	7
1.5	53	5	4
2.0	29	3	2

Table 2: Average startup latency for the realistic deployment of Figure 1.d using T-Copy, μ=0.73.

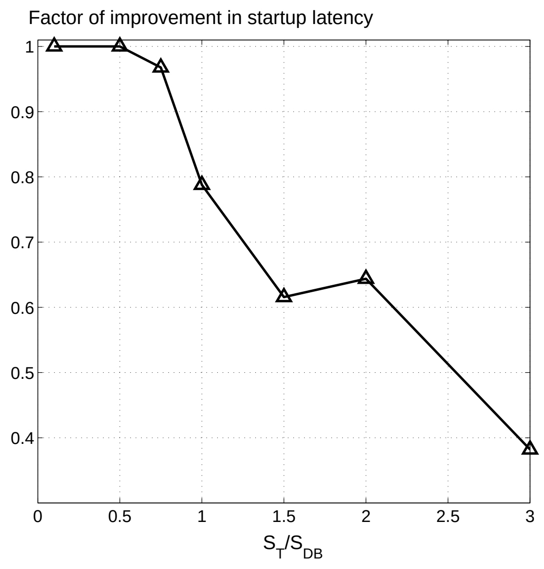
Table 2 shows the startup latency observed with greedy, Rand-Coop and Cont-Coop for different $\frac{S_T}{S_{DB}}$ values. When the total cache size is smaller than the repository size, $\frac{S_T}{S_{DB}} < 1.0$, Cont-Coop outperforms both greedy and Rand-Coop by a significant margin. With larger cache sizes, Rand-Coop provides compatible startup latencies with Cont-Coop. Both outperform greedy by a significant margin because greedy replicates the same collection of clips on each of the six nodes. When a large number of nodes observe a cache miss, they are directed to the base station, exhausting the bandwidth of wireless links from node 2 (11 Mbps), see Section D.2. With Rand-Coop and Cont-Coop, a larger selection of clips is cached in the cooperative group, enabling different nodes to stream clips from other nodes. This minimizes the formation of bottleneck links, enhancing the average startup latency.

With a more uniform distribution of access to clips ($\mu = 0.25$), greedy starts to perform almost the same as Cont-Coop. This is because most nodes observe a cache miss, exhausting the bandwidth of the links from node 2.

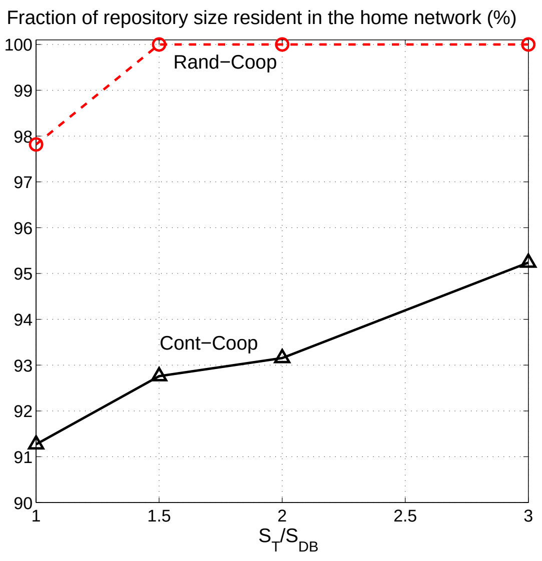
With a uniform distribution of access, Rand-Coop out performs Cont-Coop (see Figure 7.a) because it materializes the entire repository in the home network, see Figure 7.b. This enables different nodes to stream clips to one another, maximizing the utilization of the wireless links in the home network. With Cont-Coop, when multiple nodes observe a cache miss, they compete for the link bandwidth from node 2 to other nodes, increasing startup latency. Such occurrences are rare and the difference between Rand-Coop and Cont-Coop is only a few seconds. However, startup latencies are small enough that a few seconds enables Rand-Coop to outperform Cont-Coop by more than a factor of 2.

E Conclusion and future research directions

Caching techniques for streaming media in wireless home networks is an exciting area of research with many future possible extensions. In addition to topologies reported in this paper, we have applied our algorithms to topologies consisting of 4, 6, and 10 nodes. Obtained results show Cont-Coop enhances average startup latency and improves availability of data when compared with a greedy caching technique. With this key finding, a short-term research direction is as follows: What thresholds for link bandwidths should trigger nodes to form (depart from) a cooperative group? This topic requires a host of policies that consider both local and global metrics. They may construct multiple cooperative sub-groups and isolated nodes that act in a greedy manner. Obtained results apply to larger networks consisting of tens of nodes deployed in an office setting [11]. With such networks, requiring all nodes to be members of one cooperative group may degrade the overall system performance. One may utilize wireless connectivity of devices to construct multiple cooperative groups that behave independent of one another. A research topic is whether nodes at the



7.a Slow-down with Cont-Coop ($\frac{\delta(Rand-Coop)}{\delta(Cont-Coop)}$)



7.b Fraction of repository size (%DB)

Figure 7: Rand-Coop outperforms Cont-Coop with a uniform access pattern, $\mu = 0.25$, topology of Figure 1.d, T-Copy.

boundary of multiple groups should be members of core group or all groups.

We intend to explore cooperation modes that increase dependencies between devices. One possible design is to break a clip into chunks and assign different chunks to different nodes. Such a technique is considered in [6, 19]. This study constructs replicas of segments for the purposes of fault tolerance and load balancing. We plan to investigate extensions of its proposed designs to a wireless home network by exploring its availability and startup latency characteristics. A technique such as Cont-Coop which caches at the granularity of clips would serve as a yard stick to evaluate the tradeoffs associated with this alternative design.

References

[1] S. Acharya and B. Smith. MiddleMan: A Video Caching Proxy Server. In *Proceedings of NOSSDAV*, June 2000.

[2] H. Bahn. A Shared Cache Solution for the Home Internet Gateway. *IEEE Transactions on Consumer Electronics*, 50(1):168–172, February 2004.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of Infocom*, pages 126–134, 1999.

[4] G. Cao, L. Yin, and C. R. Das. Cooperative cache-based data access in ad hoc networks. *IEEE Computer*, 37(2):32–39, February 2004.

[5] P. Cao and S. Irani. Cost-Aware WWW Proxy Caching Algorithms. In *1997 Usenix Symposium on Internet Technologies and Systems*, 1997.

[6] Y. Chae, K. Guo, M. Buddhikot, S. Suri, and E. Zegora. Silo, Rainbow, and Caching Token: Schemes for Scalable, Fault Tolerant Stream Caching. *IEEE Journal on Selected Areas in Communications on Internet Proxy Services*, Sept. 2002.

[7] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell. A hierarchical internet object cache. In *USENIX Annual Technical Conference*, pages 153–164, 1996.

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

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, editors. *Introduction to Algorithms*, chapter 26.2. MIT Press, 2001.

[10] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In *2nd ACM Multimedia Conference*, October 1994.

[11] R. Draves, J. Padhye, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *MobiCom*, Sept 2004.

[12] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *Proceedings of SIGCOMM*, 1998.

[13] M. J. Feeley, W. E. Morgan, F. H. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath. Implementing Global Memory Management in a Workstation Cluster. In *Proceedings of the 15th ACM Symp. on Operating Systems Principles*, December 1995.

[14] S. Ghandeharizadeh, T. Helmi, T. Jung, S. Kapadia, and S. Shayandeh. An Evaluation of Two Policies for Simple Placement of Continuous Media in Multi-hop Wireless Networks. In *Twelfth International Conference on Distributed Multimedia Systems (DMS)*, August 2006.

[15] S. Ghandeharizadeh and S. Shayandeh. Greedy Cache Management Techniques for Mobile Devices. In *First International IEEE Workshop on Ambient Intelligence, Media, and Sensing (AIMS)*, 2007.

[16] S. Glassman. A Caching Relay for the World Wide Web. In *First International Conference on the World Wide Web*, May 1994.

[17] Y. Guo, Z. Ge, B. Urgaonkar, P. J. Shenoy, and D. F. Towsley. Dynamic Cache Reconfiguration Strategies for A Cluster-Based Streaming Proxy. *Computer Communications*, 29(10):1710–1721, 2006.

[18] T. Hara. Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility. In *INFOCOM*, pages 1568–1576, 2001.

[19] W. J. Jeon and K. Nahrstedt. QoS-aware Middleware Support for Collaborative Multimedia Streaming and Caching Service. *Microprocessors and Microsystems*, 27(2):65–72, 2003.

[20] D. R. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi. Web caching with consistent hashing. *Computer Networks*, 31(11-16):1203–1213, 1999.

[21] N. Laoutaris, G. Smaragdakis, A. Bestavros, and I. Stavrakakis. Mistreatment in Distributed Caching Groups: Causes and Implications. In *IEEE Infocom*, April 2006.

[22] D. J. Lilja. Cache coherence in large-scale shared-memory multiprocessors: issues and comparisons. *ACM Comput. Surv.*, 25(3):303–338, 1993.

[23] Jean-Marc Menaud, Valérie Issarny, and Michel Banâtre. A New Protocol for Efficient Cooperative Transversal Web Caching. In *DISC*, pages 288–302, 1998.

[24] B. S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson. Adaptive web caching: towards a new global caching architecture. *Computer Networks*, 30(22-23):2169–2177, 1998.

[25] B. Nitzberg and V. Lo. Distributed Shared Memory: A Survey of Issues and Algorithms. *Computer*, 24(8):52–60, 1991.

[26] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *MobiHoc*, pages 117–127, 2001.

[27] K. Papagiannaki, M. Yarvis, and W. S. Conner. Experimental Characterization of Home Wireless Networks and Design Implications. In *IEEE Infocom*, April 2006.

[28] R. Rejaie, H. Yu, M. Handley, and D. Estrin. Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet. In *Proceedings of IEEE INFOCOM*, 2000.

[29] A. Rousskov and D. Wessels. Cache Digests. *Computer Networks*, 30(22-23):2155–2168, 1998.

[30] M. Ségura-Devillechaise, J. Menaud, N. Loriant, R. Douence, M. Südholt, T. Fritz, and E. Wuchner. Dynamic Adaptation of the Squid Web Cache with Arachne. *IEEE Software*, 23(1):34–41, 2006.

[31] M. Shinohara, T. hara, and S. Nishio. Data Replication Considering Power Consumption in Ad Hoc Networks. In *Proceedings of the 8th International Conference on Mobile Data Management (MDM)*, May 2007.

[32] R. Tewari, M. Dahlin, H. N. Vin, and J. Kay. Design Considerations for Distributed Caching on the Internet. In *Proceedings of the 19th IEEE Conference on Distributed Computing Systems*, May 1998.

[33] S. Tewari and L. Kleinrock. On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks. In *NETWORKING*, pages 709–717, 2005.

[34] V. Valloppillil and K. W. Ross. *Cache Array Routing Protocol V1.0.* USC ISI, Feb 1998.

[35] D. Wessels and K. Claffy. ICP and the Squid Web cache. *IEEE Journal on Selected Areas in Communication*, 16(3):345–357, 1998.

[36] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the Scale and Performance of Cooperative Web Proxy Caching. *SIGOPS Oper. Syst. Rev.*, 33(5):16–31, April 1999.

[37] G. K. Zipf. Relative Frequency as a Determinant of Phonetic Change. Harvard Studies in Classified Philiology, Volume XL, 1929, 1929.