To Share or Not To Share Storage in Mesh Networks: A System Throughput Perspective*

Shahram Ghandeharizadeh, Shahin Shayandeh, Tooraj Helmi
Computer Science Department
University of Southern California
Los Angeles, CA 90089
shahram@usc.edu, shayande@usc.edu, thelmi@usc.edu

Abstract

While the cost per megabyte of storage is economical, sharing storage might be expensive because delivery of a clip occupying the shared storage requires bandwidth. This is especially true for mesh networks where devices are constrained by the radio-range and bandwidth of their wireless networking card. Assuming a device, termed a peer, is configured with a mass storage device, it may share either all, a fraction, or none of its storage. When a peer does not shares its storage, it stores clips with the objective to enhance a local criterion such as number of clip requests serviced using its mass storage device. When a peer shares its storage, it may store clips to optimize a global metric such as the total number of devices that may display their clips simultaneously, termed system throughput. Using this global metric, we show the following surprising result: the throughput of certain mesh networks is enhanced when peers do not share storage. By understanding these tradeoffs, one may design adaptable software to enable a peer to monitor its environment and decide to share or not to share its storage. We demonstrate the feasibility of such a design using a simulation study.

1 Introduction

In multi-hop wireless networks, two peers may communicate by employing a number of intermediate peers that collaborate to relay information. An interesting subclass, termed mesh networks, is more constrained by bandwidth and storage limitation and less constrained by peer mobility and power consumption. An example includes Canada's SuperNet [3] which delivers educational material to remote areas of Alberta using wireless peers as extensions of its wired infrastructure. Another might be an office setting [10] or a home entertainment system [6] where multiple peers collaborate to provide on-demand access to continuous media, audio and video clips[1]. With the later, a household may store its personal library on a peer for retrieval everywhere, e.g., when visiting a friend's home. A peer may encrypt its content to either protect it from un-authorized access, i.e., authentication, or implement a business model for generating revenues. These mesh networks raise a host of privacy and security issues [2] that constitute an active area of research by several communities.

Assuming a peer is configured with a mass storage device, it may share either all, a fraction, or none of its storage. In a self-organizing environment where network connectivity is dictated by peers being in radio-range of one another, a peer must decide whether it shares its storage. This study investigates this topic by focusing on one performance metric, system throughput. Throughput is defined as the number of simultaneous peers able to display their referenced clips. In order to display a clip, a peer may stream the clip from either its mass storage device or its neighboring peers.

Maximizing system throughput is challenging when a peer, say P_c, references a clip that does not reside in its local storage. In this case, P_c must discover neighboring peers, say $\{P_1, P_2, ..., P_p\}$, with a replica of the referenced clip. Next, P_c admits itself into the network and streams the clip from one or more candidate servers by reserving bandwidth along those paths that connect them to P_c. Streaming overlaps display of the clip at P_c with delivery of its remainder, minimizing startup latency. Startup latency is defined as the time elapsed from when the request is issued at P_c until P_c initiates display of the referenced clip.

Streaming requires an admission control component to prevent a new request from interfering with those active requests that are streaming their referenced clips. If net-

*This research was supported in part by NSF grant IIS-0307908.

[1]The focus of this study is on delayed mode of communication using pre-recorded clips. Real-time mode of communication benefits marginally from sharing storage.

978-0-7695-3096-3/08 $25.00 © 2008 IEEE
DOI 10.1109/WAINA.2008.84

IEEE computer society

Authorized licensed use limited to: University of Southern California. Downloaded on November 29,2020 at 06:22:53 UTC from IEEE Xplore. Restrictions apply.

work bandwidth is not managed intelligently, a new request may cause many active displays to starve for data, causing their display to suffer from frequent disruptions and delays, termed hiccups.

Ideally, a client should find its referenced clip in its local storage. This would avoid the discovery process, use of limited network bandwidth and admission control. This is possible when the continuous media repository size is smaller than the size of each peer's mass storage device. Even with today's 750 Gigabyte disk drives, there is an explosion of content and it is unrealistic to assume the entire repository fits on the mass storage device of a peer.

When the repository size exceeds a peer's storage capacity, a peer may store clips to enhance either a local or a global criterion. A local criterion might be to maximize the number of references serviced using a peer's local storage. This criterion motivates a peer not to share its storage. Simple is a greedy data placement strategy to realize this criterion. With a global criterion, a peer may collaborate with other peers to enhance the overall system throughput. This motivates a peer to share its storage. Halo-clip is a data placement strategy for this mode of operation. Of course, one may consider a hybrid of these two possibilities where a peer decides to share or not to share its storage. By understanding the tradeoffs associated with Simple and Halo-clip, this paper presents Hybrid which enables a peer to choose whether it employs Simple or Halo-Clip. Obtained results show Hybrid outperforms a technique that requires all peers either to share storage (Halo-clip) or not to share storage (Simple). In this study we assume a centralized admission control in order to focus on the comparison of different data placement strategies.

Placement of continuous media in ad-hoc networks is an emerging area of research. Several studies assume storage of a peer is shared [11, 4, 1]. Simple [8] is a technique that does not share storage. To the best of our knowledge, no prior study quantifies the tradeoffs associated with sharing storage. This is the key novelty of our study.

The rest of this paper is organized as follows. Sections 2 and 3 outline two alternative data placement strategies designed not to share and to share storage, respectively. We quantify their tradeoffs in Section 4. Obtained results lead to development of Hybrid in Section 5 which is shown to be superior to both Simple and Halo-Clip. Brief conclusions and our future research directions are contained in Section 6.

2 Not to share storage: Simple

A Simple data placement strategy assigns clips to each peer with the objective to enhance a local criterion such as the local hit ratio of each peer. In [8], we study two alternative policies for Simple: Frequency-based and Byte-hit.

Parameter	Definition
A	Area of geographical region covered by the peers
R	Radio range of a peer
C	Number of clips in the repository, $1 \leq i \leq C$
$\mathcal{N}$	Number of peers
f_i	Frequency of access to clip i
$S_{C,i}$	Size of clip i
S_{DB}	Size of the database, $S_{DB} = \sum_{i=1}^{C} S_{C,i}$
S_N	Storage capacity of a peer
S_T	Total storage capacity of the peers in the network
$S_{P,i}$	Size of clip i's prefetch portion
$B_{Display,i}$	Bandwidth required to display clip i
$\overline{B_{Display}}$	Average display bandwidth of clips in the database
$B_{Link}(i,j)$	Bandwidth of a link connecting two peers i and j
$\overline{B_{Link}}$	Average bandwidth of links in the mesh network
$\overline{B_{Link,j}}$	Average network bandwidth of node j to its neighbors
r_i	Number of replicas for clip i
H_i	Number of hops for clip i
$\overline{H_i}$	Average number of hops for clip i

Table 1. Parameters and their definitions

Frequency-based sorts clips based on their frequency of access (f_i) and assigns those with highest value to a peer until the storage capacity of the peer is exhausted. Byte-hit is similar with one difference: It sorts clips based on their frequency of access to each byte, $\frac{f_i}{S_{C,i}}$. We show Byte-hit is superior to Frequency-based because it maximizes system throughput and is more robust to errors in frequency of access to clips.

We use Byte-hit as the representative of Simple for comparison with Halo-clip in Section 4.

3 To share storage: Halo-Clip

When sharing storage, an algorithm places data to enhance a global metric such as the number of peers that may display their referenced clips simultaneously. In addition to specifying a framework to quantify this metric, the algorithm must address the following questions: First, what is the granularity of data (a block or a clip) occupying the shared storage? Second, how many replicas of a granule should be constructed in the mesh network? Third, how should these replicas be placed across devices? In this section, we present Halo-Clip and its decentralized implementation to address these questions.

Halo-clip controls placement of data at the granularity of a clip. To minimize startup latency, a peer may prefetch the first few blocks of each clip. When a user references a clip using a peer, if the system admits the request into the network then the peer displays its prefetch portion while streaming its remainder from one or more neighboring peers. The size of clip i's prefetch portion is dictated by: 1) the estimated available bandwidth of the path between a client and the closest replica(s) of clip i, termed B_{Path}, and

Authorized licensed use limited to: University of Southern California. Downloaded on November 29,2020 at 06:22:53 UTC from IEEE Xplore. Restrictions apply.

2) clip size. If B_{Path} exceeds $B_{Display}$ then startup latency is the time required to deliver the first block, which dictates the size of the prefetch portion. Otherwise, prefetch size is [7]: $S_{P,i} = S_{C,i} - \lfloor \frac{B_{Path}}{B_{Display,i}} \times S_{C,i} \rfloor$, where $S_{C,i}$ denotes the clip size.

To maximize the number of simultaneous displays, we minimize the average amount of bandwidth required to transmit the clips across the network. This is defined as:

$$\overline{B(H_i)} = \sum_{i=1}^{C} f_i B_{Display,i} \overline{H_i} \tag{1}$$

where f_i is the frequency of access to clip i, and $\overline{H_i}$ denotes the average number of hops required to deliver clip i to a client. Average startup latency is dictated by the time required to deliver the clips' prefetch portion.

$$\overline{L(H_i)} = \sum_{i=1}^{C} f_i \frac{S_{P,i} - \overline{S_{P,i}}}{B_{Display,i}} \overline{H_i} \tag{2}$$

where $\overline{S_{P,i}}$ denotes the average amount of prefetch data on each peer. It is possible for the prefetch portion of a clip to be present on some peers and absent from others. In this case, $\overline{S_{P,i}}$ is the total size of the replicated prefetch portions divided by $\mathcal{N}$. When $\overline{S_{P,i}}$ and $S_{P,i}$ are equal, $\overline{L(H_i)}$ becomes zero, providing the best startup latency. Note that both Equations 1 and 2 use $\overline{H_i}$ as the average number of hops to deliver either the clip itself or its prefetch portion.

The amount of storage on each peer is finite. Thus, the placement of data must satisfy the following constraint:

$$\sum_{i=1}^{C} r_i S_{C,i} \leq S_V \tag{3}$$

where S_V is the total storage of peers minus the amount of prefetch data, i.e., $S_V = S_T - \mathcal{N} \sum_{i=1}^{C} \overline{S_{P,i}}$

Assuming a graph topology, we relate the number of replicas r_i to $\overline{H_i}$ as $r_i = \lceil \frac{Q}{\overline{H_i}^2} \rceil$, where Q is a constant describing the area covered by the nodes and the density of the nodes. For example, with a general graph structure, $Q = \frac{2A}{3\sqrt{3}R^2}$ [9]. Using the Lagrangian multipliers method, the optimization problem can be expressed as:

$$Min \left\{ F(\overline{H_i}) + \varphi \left[\sum_{i=1}^{C} \frac{Q S_{C,i}}{\overline{H_i}^2} - S_V \right] \right\} \tag{4}$$

Where $F(\overline{H_i})$ is the target function:

$$\begin{aligned} F(\overline{H_i}) &= \alpha \overline{B(H_i)} + (1-\alpha) \overline{L(H_i)} \\ &= \sum_{i=1}^{C} f_i \overline{H_i} \left(\alpha B_{Display,i} + (1-\alpha) \frac{S_{P,i} - \overline{S_{P,i}}}{B_{Display,i}} \right) \\ &= \sum_{i=1}^{C} f_i \overline{H_i} \Omega_i \end{aligned}$$

Where $\Omega_i = \alpha B_{Display,i} + (1-\alpha) \frac{S_{P,i} - \overline{S_{P,i}}}{B_{Display,i}}$ and α is a tuning parameter that controls the weight (importance) given to $\overline{B(H_i)}$ and $\overline{L(H_i)}$, $0 \leq \alpha \leq 1$.

We solve for $\overline{H_i}$ to obtain:

$$\overline{H_i} = \sqrt[3]{\frac{2\varphi Q S_{C,i}}{f_i \Omega_i}} \tag{5}$$

Substituting $\overline{H_i}$ in the constraint, we compute the Lagrangian multipliers as:

$$\varphi = \frac{1}{2} \sqrt{\frac{Q \left(\sum_{i=1}^{C} \sqrt[3]{f_i^2 \Omega_i^2 S_{C,i}} \right)^3}{S_V^3}} \tag{6}$$

Substituting φ back into Equation 5, we obtain:

$$\overline{H_i} = \sqrt{\frac{Q \sum_{k=1}^{C} \sqrt[3]{f_k^2 \Omega_k^2 S_{C,k}}}{S_V}} \times \sqrt[3]{\frac{S_{C,i}}{f_i \Omega_i}} \tag{7}$$

Now, the average number of replicas for each clip i is:

$$r_i = \lceil \frac{S_V \sqrt[3]{f_i^2 \Omega_i^2}}{\sqrt[3]{S_{C,i}^2} \sum_{k=1}^{C} \sqrt[3]{S_{C,k} f_k^2 \Omega_k^2}} \rceil \tag{8}$$

3.1 Decentralized implementation

One may implement Halo-clip using a randomized algorithm. In this paper, we use a variation of the three stage technique described in [9]. Its three stages are Snooze, Create, and Challenge. When a peer publishes a new clip, say clip X, it broadcasts a publish message to all peers in the mesh network. A receiving peer P_i spawns a thread to invoke the three phases in turn. During Snooze, the thread sleeps for a random period of time. Upon expiration of this time out, the thread invokes Create. During this phase, if available storage of peer P_i exceeds the size of clip X then it selects itself as a candidate server for this clip. Otherwise, it determines if other clips should be swapped out in favor of X. This might be performed by comparing the byte-hit value of X with those clips occupying its storage. If sufficient victim objects can be identified to store X, P_i marks these as victim objects. Otherwise, Halo-clip terminates without storing X.

Assuming P_i decides to store clip X, it generates a suppress message to prevent those neighboring peers that are H_i hops away from becoming candidate servers for X. Each peer P_j that receives this message might be in either Snooze, Create or Challenge phases for Halo-clip. If in Snooze phase then P_j increases its sleep time. If in Create phase then P_j challenges P_i as the candidate server for X. At this point both P_j and P_i enter the Challenge phase of Halo-clip. Halo-clip may use a variety of criteria to

Authorized licensed use limited to: University of Southern California. Downloaded on November 29,2020 at 06:22:53 UTC from IEEE Xplore. Restrictions apply.

decide the winner. These might include each peer's free storage, number of neighbors, and available network bandwidth. The winner, say P_i, is the candidate server for X.

If P_i's suppress message encounters a peer P_j executing challenge phase of Halo-clip with another peer P_k then P_j waits[2] until the outcome of its current challenge is resolved prior to challenging P_i. If P_j is the winner then P_j challenges P_i. Otherwise, P_j terminates its invocation of Halo-clip.

The candidate server for X generates a message to inform all those neighbors H_i hops away that it is the server for clip X. At this point all peers in the Snooze phase terminate their invocation of Halo-clip.

4 A Comparison

In this section, we compare Simple with Halo-clip. Key insights are as follows. When access frequencies are either unknown or expected to vary significantly in a short span of time, Halo-clip is superior to Simple because it enhances the likelihood of a peer acting as a server for its neighbors. Otherwise, simple is superior to Halo-clip when a deployment is storage challenged and has limited network bandwidth. Below, we elaborate on these insights.

We start with an overview of our experimental methodology. Next, we present the obtained results.

Experimental design: We have developed a flexible experimental framework that can emulate alternative network topologies, peer specifications, and repository characteristics. This framework consists of two distinct components: data placement, and workload generator. The first component consumes configuration files that specify system and repository characteristics along with a choice of data placement strategy. It computes a placement of clips across the peers. The second component consumes this data placement and a target access distribution to clips in order to compute the system throughput.

Both components support a variety of network topologies such as String, Grid, and arbitrary Graphs. With all topologies, we assume the presence of a base station that provides access to the wired infrastructure and remote servers. It is located at one end of the String topology and one corner of the Grid and Graph topologies. We examined different peer specifications, manipulating the storage capacity of each peer and its network bandwidth. We considered repositories consisting of both a single media type and a mix of media types. With each, we manipulated the number of clips for each media type, and frequency of access to the different clips.

The workload generator computes system throughput as follows. It averages the number of simultaneous displays supported by the following process for a fixed number of iterations. In each iteration, it picks peers in a round-robin manner. The chosen peer references a clip for display per specified Zipfian distribution. If the referenced clip, say clip i, resides in local storage of the peer then the number of simultaneous displays of this iteration is incremented by one. Otherwise, it invokes a centralized[3] admission control component that employs the Ford-Fulkerson algorithm [5] to stream the referenced clip from neighboring peers or the base station. If the request can be admitted into the system then the number of simultaneous displays is increased by one. Otherwise, the request is rejected. Since the request is either admitted or rejected instantaneously, the tolerable startup latency in this case is zero. This might be different in a real system that would show advertisements while waiting for the request to be admitted. An iteration ends once all $\mathcal{N}$ peers have invoked this procedure. Note that the maximum throughput for an iteration is $\mathcal{N}$.

Robustness to error in frequency of access: We analyze the robustness of Halo-clip and Simple by manipulating the frequency of access to clips inputted to the workload generator. The Zipfian distribution of access is a property of the clip repository used by the data placement component to assign clips to peers. An input to the workload generator is a distribution that shifts the original one with parameter g. The value of g ranges from 0 to $C-1$. This means the frequency of access for clip i assumed when placing data is assigned to clip $(i+g)\ mod\ C$ when generating the workload. Both components employs the same distribution when $g=0$.

Obtained results: The insights presented at the beginning of this section are based on analyzing numerous results obtained from different experimental settings. This section shows the key insights using the simplest experiments. Its observations hold true for more complex experiments, as summarized at the end of this section.

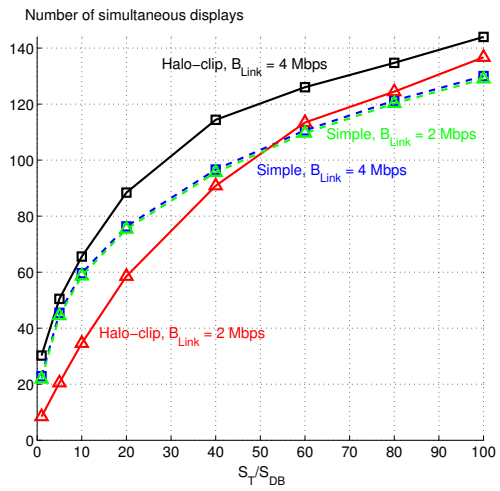
Figure 1 shows the average number of simultaneous displays with Simple and Halo-clip for a variety of $\frac{S_T}{S_{DB}}$ ratios when $g=0$, i.e., the same distribution of access is used by both data placement and workload generator. Target repository consists of one media type with a display bandwidth requirement of 4 Mbps. There are 576 equi-sized clips. Our target mesh topology is a Grid consisting of 144 peers with a link bandwidth of 2, 4, or 12 Mbps. The maximum possible system throughput is 144.

We increase $\frac{S_T}{S_{DB}}$ by increasing the storage of each peer while maintaining the same repository of clips. This causes both Simple and Halo-clip to support a higher system throughput because both techniques assign a larger fraction

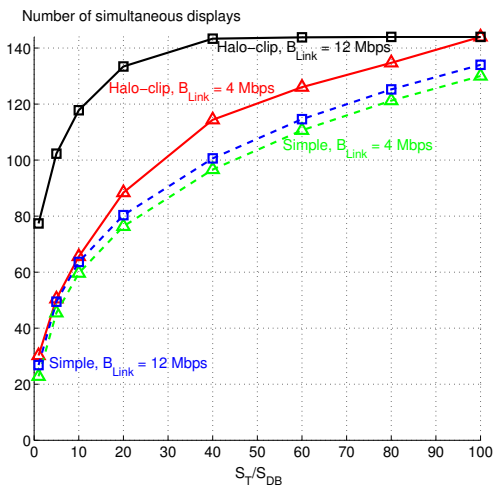
[2] P_j waits for the outcome of its challenge because it is the recipient of the suppress message and the algorithm is distributed.

[3] We assume a centralized admission control in order to minimize the number of research topics and maintain our focus on evaluating alternative data placement strategies. An investigation of alternative admission control policies including distributed ones is a future research topic.

Authorized licensed use limited to: University of Southern California. Downloaded on November 29,2020 at 06:22:53 UTC from IEEE Xplore. Restrictions apply.



1.a) $B_{Link}(i,j)$ bandwidths of 2 and 4 Mbps.



1.b) $B_{Link}(i,j)$ bandwidths of 4 and 12 Mbps.

Figure 1. Simple and Halo-clip with different $B_{Link}(i,j)$ bandwidths

of the repository to each node, reducing demand for network bandwidth to stream clips from either a neighboring peer with Halo-clip or the base station with Simple.

When the link bandwidth is comparable to the display requirement of a clip ($B_{Link}(i,j)$=2 Mbps ¡ $B_{Display,i}$=4 Mbps) and $\frac{S_T}{S_{DB}}$ is less than 60, Simple outperforms Halo-clip. This is because Simple enables a peer to service many more requests from its local storage when compared with Halo-clip. To illustrate, when $\frac{S_T}{S_{DB}}$=10, the average frequency of access to clips assigned to a peer i ($\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} f_{i,j}$ for the k clips assigned to peer i) is 39.52% with Simple. It is 18.91% with Halo-clip; almost one half of that with Simple. Halo-clip requires a peer to stream clips from the shared storage of its neighboring peers. It exhausts the available network bandwidth quickly, preventing the system from admitting requests. This results in a lower system throughput. As we increase the amount of storage per peer ($\frac{S_T}{S_{DB}}$ values beyond 60), Halo-clip constructs additional replicas of each clip. This enhances the number of requests serviced using local storage of a peer and reducing the average number of hops required to stream a clip. To illustrate, when $\frac{S_T}{S_{DB}}$ is 80, average frequency of access to clips for each peer is 79.43% with Halo-clip and 82.68% with Simple. This minimizes the demand for network bandwidth and enables Halo-clip to outperform Simple.

When we change the link bandwidth to 12 Mbps, Halo-clip outperforms simple by a wide margin for all $\frac{S_T}{S_{DB}}$ values. The additional network bandwidth enables peers to stream clips from their neighboring devices. With Simple, every time a peer fails to find a referenced clip in its local storage, it must stream the clip from the base station which is located in one corner of the Grid topology. Once the link bandwidth of the peers (in this case 2) neighboring the base station are exhausted, the base station may not stream additional clips. This explains the modest gains with Simple when link bandwidth is increased from 4 to 12 Mbps. Halo-clip realizes the maximum possible system throughput when $\frac{S_T}{S_{DB}}$ is 40. Beyond this point, increasing the amount of storage per peer does not enhance system throughput further (unless the access frequencies are erroneous, as discussed in the following paragraphs).

In passing, the placement of the base station has a dramatic impact on system throughput with Simple. If it was centrally located in the Grid topology, it would have more neighbors, enabling Simple to support a higher throughput as we increase the link bandwidth.

We compared Simple and Halo-clip using a heterogeneous repository and a variety of network topologies. The insights presented in this section continue to hold true and repeat themselves. Also we ran simulations to compare robustness of Halo-clip and Simple to error in frequency of access. Obtained results show that Halo-clip provides either a comparable or a superior throughput to Simple when $g > 0$ which is translated to error in frequency of access.

Due to lack of space, we presented a sketchy outline of our key observations and results in above.

5 Hybrid

One may design a hybrid technique where a peer decides whether it employs Simple or Halo-clip to manage its storage. Our experimental results show this hybrid technique performs the same as Simple when Simple is superior and

Authorized licensed use limited to: University of Southern California. Downloaded on November 29,2020 at 06:22:53 UTC from IEEE Xplore. Restrictions apply.

Halo-clip otherwise.

The details of this technique, termed Hybrid, are as follows. Periodically, a peer j monitors its average network bandwidth to its neighbors ($\overline{B_{Link,j}}$) and the average bandwidth required to display clips ($\overline{B_{Display}}$) that constitute the repository. As long as this ratio ($\frac{\overline{B_{Link,j}}}{\overline{B_{Display}}}$) is greater than one, this peer shares its storage using Halo-clip. As soon as this ratio becomes less than one, it switches to Simple.

Thus, with Hybrid, peers with abundant network bandwidth share their storage while those with insufficient network bandwidth do not share their storage. Hybrid does not consider storage capacity of the network relative to repository size, i.e., $\frac{S_T}{S_{DB}}$ ratios, because Halo-clip is superior to Simple only when storage is constrained and network bandwidth is abundant. Hybrid's criterion is a generalization of this.

We considered different network topologies to compare Hybrid with Simple and Halo-clip. In investigated topologies some nodes are configured to have scarce bandwidth (their $\overline{B_{Link,j}}$ is less than the $\overline{B_{Display}}$) and the rest have abundant available bandwidth.

Figure 2 shows the number of simultaneous displays supported by Hybrid, Halo-clip, and Simple. Target topology is a 100 node Grid divided into four zones. Each zone consists of 25 nodes and occupies one corner of the grid. Two diagonal zones have scarce bandwidth while the other two have abundant bandwidth. The average link bandwidth of this target topology is 3.5 Mbps. As a function of $\frac{S_T}{S_{DB}}$ ratios, Hybrid enables a peer to adjust dynamically to either share or not to share its storage. With Hybrid, when a peer invokes Halo-clip, it assumes the storage capacity of the entire network as shared when placing clips across peers. A peer selectively discards Halo-clip's proposed placement by not sharing its storage (and employing Simple).

6 Future Research

One of our long term research objectives is to investigate the interaction of system throughput with other performance metrics including clip availability and startup latency. Clip availability quantifies what fraction of a repository is available to a peer when it becomes disconnected from the mesh network. Startup latency is the delay from when a peer issues a request for a clip to the onset of its display. Halo-clip is flexible enough to enable a peer to prestage a prefetch portion of a clip in anticipation of future references. This enhances the peer's startup latency by reducing its amount of shared storage. An understanding of these interactions enables a peer to adjust parameter settings of Halo-clip in response to the requirements of its target application.

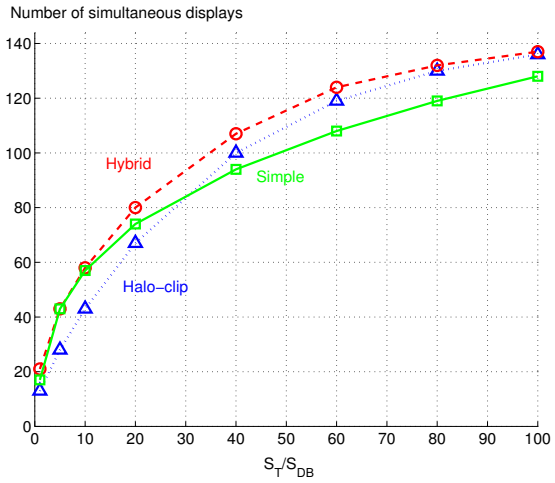


Figure 2. Hybrid versus Simple and Halo-clip

References

[1] A. Aazami, S. Ghandeharizadeh, and T. Helmi. An Optimal Continuous Media Replication Strategy for Ad-hoc Networks of Wireless Devices. In *Tenth International Workshop on Multimedia Information Systems (MIS)*, August 2004.

[2] W. Cheng, L. Golubchik, and D. Kay. Total Recall: Are Privacy Changes Inevitable? a position paper. In *Proceedings of the First ACM Workshop on Continuous Archival and Retrieval of Personal Experiences*, New York, NY, October 2004.

[3] S. M. Cherry. Across the great divide. *Spectrum IEEE*, 41(1):36–39, January 2004.

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

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

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

[7] 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.

[8] 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.

[9] S. Ghandeharizadeh, B. Krishnamachari, and S. Song. Placement of Continuous Media in Wireless Peer-to-Peer Networks. *IEEE Transactions on Multimedia*, April 2004.

[10] K. Jain, J. Padhye, V. Padmanabhan, and L. Qui. Impact of Interference on Multi-hop Wireless Network Performance. In *ACM Mobicom*, Sept 2003.

[11] S. Jin and L. Wang. Conten and Service Replication Strategies in Multi-hop Wireless Mesh Networks. In *Proceedings of the IEEE/ACM MSWiM*, October 2005.

Authorized licensed use limited to: University of Southern California. Downloaded on November 29,2020 at 06:22:53 UTC from IEEE Xplore. Restrictions apply.