CD-PAN: A Protocol for Peer-to-Peer Content Distribution in a Weakly Connected and Heterogeneous Personal Area Network

Shiva Chaitanya

Dept. of CSE
Pennsylvania State University
State College, PA 16801

Prasenjit Sarkar

Storage Systems
IBM Almaden Research
San Jose, CA 95120

Abstract

This paper presents *CD-PAN*[1], a mechanism to automatically distribute content objects to weakly connected heterogeneous content devices in a personal area network without a global namespace. The content devices under consideration range from cell phones to personal computers, each of which is capable of downloading content objects on its own. The proposed mechanism alleviates the need to manually synchronize content that is downloaded to each of these content devices. A simulation study shows that *CD-PAN* outperforms other prefetching schemes in all our workload experiments. The performance improvement tended to increase with increase in popularity distribution skew, temporal locality and frequency of content creation/updates. The performance of *CD-PAN* increases when pair-wise communication capabilities are higher, and also adapts well to increasing power and metadata constraints.

1 Introduction

Electronic devices are becoming increasingly popular as content sharing gadgets due to the recent advances in storage capacity and network connectivity. With the increasing number of convergent applications, there is a need to efficiently share content across devices. Mobile devices are capable of autonomously downloading content via wireless broadband links into embedded storage cards – in the past, such devices had to rely on personal computers with wired broadband links. Furthermore, with the advent of technologies like Wi-fi, Bluetooth and mobile ad-hoc networks (MANETs), mobile devices can communicate on-the-fly with peer mobile devices as well as larger static devices such as personal computers and digital video recorders. It is envisioned that in the future, more electronic devices such as cameras, camcorders and music players will evolve into content devices.

The problem being addressed in this paper is the distribution of content across the vast spectrum of devices from personal computers to cell phones in order to ensure availability of frequently used data on any device at any given time. In a personal area network (PAN) comprising of devices belonging to a single user or a small set of users, the global workload across devices is determined by the users' inherent profiles and changing interests. With mobile devices increasingly supporting convergent applications, a global snapshot of content shows considerable similarity across personal devices and evolves with time in an organized fashion [31, 44]. At present, the only way to synchronize content is explicit, manual, and tedious in most cases. There are utilities that allow for automatic content synchronization [47,42,54] but they are typically personal computer based and specific to a content device. Personal computer-based synchronization is ill-suited for decentralized content sharing networks provided by wireless devices. For example, getting an audio file manually from a Digital Video Recorder (DVR) to a cell phone is challenging for the average home user.

Our work addresses the unique challenges of a personalized environment through a peer-to-peer content exchange protocol. A personal area network is different in size from well-known distributed systems that share data. In a personal area network the numbers of devices and users are very small and the number of active users on a device is typically one. Also, the resources present on mobile devices like storage space, CPU and battery power are limited. Furthermore, the network connectivity between the devices in such an environment is intermittent. Finally, the connectivity, if available, between any two peer devices is weak compared to the content sizes available for distribution.

Our work fundamentally differs from the previous research in the following ways: (a) It differs from prior works on Content Distribution Networks [7] in that they were primarily designed for strongly connected networks. (b) It differs from prior works dealing with consistency [49, 51, 5] and disconnected operations [28, 41] in that it uses a completely decentralized peer to peer model for information exchange. (c) It differs from peer to peer and ad hoc networks [53] based information dissemination approaches in that it considers the unique properties of the personal area network domain. The prior works exploit the scalability of several users coexisting and sharing information whereas we exploit the strong similarity of access profiles on personalized devices for content distribution.

The key contribution of this paper is the design of an automatic peer-to-peer content distribution mechanism based on an integrated content prefetch and eviction protocol called *CD-PAN*. Since storage capacities on devices are limited compared to rapidly growing user content, data needs to be moved between devices so that different user access patterns are satisfied. Consequently, we model the problem as one of exchanging content objects to match user access profiles in a decentralized fashion between content devices. We also present techniques to alleviate the meta-data processing required by our mechanism and suggest ways by which the power consumed due to device interactions can be reduced.

We designed and implemented a simulation model for *CD-PAN* and used synthetic and derived workloads to compare *CD-PAN* to alternative schemes. The simulation results show that *CD-PAN* outperforms a raw prefetching scheme in all workload experiments. The performance improvement tended to increase with increase in popularity distribution skew, temporal locality and frequency of content creation/updates. The performance of *CD-PAN* increases when pair-wise communication capabilities

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

are higher, and also adapts well to increasing power and metadata constraints.

The rest of the paper is organized as follows: Section 2 presents the operating environment for *CD-PAN*. Section 3 presents the architecture for *CD-PAN*, with an emphasis on the content exchange protocol between two peer devices. It also describes the simulation model for evaluating *CD-PAN*, as well as the workloads and comparable systems used in the evaluation. Section 4 presents the results of our simulation study that show the effectiveness of the content distribution system. Finally, related work and conclusions are presented in Sections 5 and 6 respectively.

2 Operating Environment

The environment in which *CD-PAN* operates consists of a network of interconnected personal devices i.e. a personal area network (PAN). Although these devices are increasingly running convergent applications and sharing content, they have different characteristics in terms of CPU power, communication bandwidth, storage capacity, power consumption and cost. Some of the devices and their characteristics are seen in Figure 1.

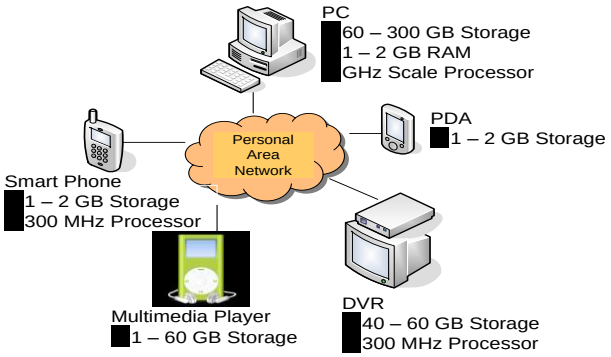


Figure 1: Personal Area Network

Emerging applications require that the devices in a PAN communicate with each other to synchronize user content, either automatically or explicitly. Because of the diversity of devices in a PAN, any content distribution mechanism must take into account the resource constraints of each device.

3 *CD-PAN* Protocol

The detailed presentation of *CD-PAN* first describes the organization of the content objects in a device, which facilitates efficient context exchange between two devices. We then describe how devices running *CD-PAN* exchange content objects to maximize the probability of usage.

3.1 Content Object Metadata

Every device is assumed to have a content organization format that is optimized for the function of the device. For instance, most devices use a file system to organize content in the device – the sophistication of the file system can vary depending on the capabilities of the device. For content distribution, we define additional metadata on the content objects in each device in order to optimize the exchange of content between two devices. We define a *content object* as a single file or a collection of files that is necessary to use the content. For example, an audio object can be represented by a single audio format file while a movie object can be made up of a collection of files.

We also assume that it is possible to generate a globally unique identifier for each content object in a device. The identifier is used for the purpose of identifying content objects to the content distribution mechanism and does not include other namespace identifiers in the devices. A global identifier can be generated based on the content, using standard hashing mechanisms such as MD5. A hash collision may degrade the effectiveness of *CD-PAN* but not cause any errors in correctness. A more sophisticated approach is to use an algorithm that can generate the same hash key for content that is different but perceived as identical by a human observer.

To estimate the popularity of content objects in the device, *CD-PAN* associates an access counter with each object. The access counter is the number of hits seen per unit size of an object *o* and is represented as *AC(o)*. In this paper, we assume the unit size to be a kilobyte, albeit other unit sizes can be used without loss of generality. Consequently, a small object with a reasonable number of hits can get precedence over a large popular object in both fetching and eviction. This works well for small mobile devices where size constraints place restrictions on the number of large objects that can be accommodated.

Each device contains two data structures, $TopObjects^{n}(A)$ and $BottomObjects^{n}(A)$. The former contains the globally unique identifiers of the *n* most popular content objects currently present in device A while the latter contains the identifiers of the least popular *n* content objects in device A. We have an independent reference model here by assuming that the popularity of a content object is directly related to its access counter value. The popularity of content object O1 is greater than that of content object O2 if the access counter value of O1 is greater than that of O2. To address the temporal effects in object popularity, we periodically weigh the access counters by multiplying each of them by an *aging fraction*.

CD-PAN also keeps track of the misses for content objects not present in a device. To that end, the metadata has a *MissList(A)* for device A that is defined to be a list of objects that are absent in the device but for which requests were issued. Requests for missing objects in a device can either be issued because they were previously present and evicted afterwards or they were referenced through links from other objects such as a hyperlink from one web object to other. The miss list is used to determine the content objects that need to be exchanged during when two devices come into contact. Finally, the counter *N(A)* indicates the number of content objects in device A. The size of the miss list is bounded by *N(A)* as that is the maximum number of content objects that can be retrieved into device A. This bound assumes that object sizes in content devices are equivalent, though in reality object sizes may differ greatly between content devices.

Note that we do not take the number of misses to an object into account here, because the number of misses does not truly indicate the future access probability of the object. For example, an application can detect quite early that a very popular object is absent on the device and not try to access it further, resulting in very few number of misses recorded for the object. However in the calculation of access counters for an object, the total number of requests seen to objects also includes the number of misses to the object.

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

3.2 Content Exchange Protocol

We now present the design of the *CD-PAN* content exchange protocol. Our design is motivated by the fact that devices in a personal area network are typically not connected to each other for long periods. We therefore do not assume on-demand fetches (i.e. servicing object misses immediately by contacting a central server or a peer device) and query forwarding between peer devices to service object misses. Also, the battery costs required for continuous network communication and setting up an ad-hoc networking infrastructure are unacceptable. Hence, objects are prefetched by a device A from a peer device B when the two devices A and B are in proximity.

In the following description of the protocol, we assume that device A is trying to retrieve objects from device B. The reverse case of device B obtaining content objects from device A flows in a symmetrical manner. The protocol proceeds in five distinct phases (Figure 2):

1. In the *Miss initiated prefetch determination* phase, device A obtains the necessary object metadata information from device B and determines the content objects that need to be fetched from device B to satisfy the future accesses based on A's miss-list.
2. In the *Aggressive prefetch determination* phase, device A uses the metadata information obtained from device B to determine the content objects that need to be optimistically fetched from device B to satisfy the future accesses due to temporal variations in workload.
3. In the *Power constraint determination* phase, device A imposes any power constraints on the number of objects that device A fetches from device B.
4. In the *Integrated prefetch and eviction* phase, device A actually fetches and evicts the objects that meet the evaluation criterion from the earlier phases.
5. Finally, in the *Parameter tuning* phase, device A recalculates tuning parameters that are used the next time device A comes into proximity with device B.

3.2.1 Miss Initiated Prefetch

In the first phase the device trying to prefetch objects looks to obtain the metadata information for the objects that have incurred misses in the device. Device A first sends the miss-list of device to device B which responds back with a data structure:

$$LocatedList^n(A,B) = MissList(A) \cap CurrentObjects(B) \quad n = l_{AB} * \min(N(A), N(B)), 0 \le l_{AB} \le 1 \qquad (1)$$

$LocatedList^n(A,B)$ contains the set of object identifiers in the miss-list of device A present in device B. The data structure is sorted by the access counter values of objects as calculated in device B and the first n object identifiers are sent to A. The value n of the queried $LocatedList^n(A,B)$ represents a fraction of the minimum of content object capacities for devices A and B. The term $min((N(A), N(B))$ represents the bound for the number of items in the $LocatedList^n(A,B)$ as we never replace more than the minimum of the content object capacities for devices A and B. This bound is based on the assumption of object size equality stated in Section 3.1 and we validate this assumption in our experimental section (Section 4.1). The fraction l_{AB} is initially set to 1 and reset at every parameter tuning phase after evaluating content objects in device B.

Device A then makes an initial valuation of the content objects present in the $LocatedList^n(A,B)$. This is done by recalculating the access counters of the content objects in $LocatedList^n(A,B)$ based on the relative access profiles on the two devices. We assume a simple linear model to relate the access profiles of common objects between a pair of devices. If o is a content object in $LocatedList^n(A,B)$, then the access counter for the object is recalculated as:

$$AC(o) = k^{miss}_{AB} * AC(o) \qquad (2)$$

The term k^{miss}_{AB} represents the linear factor that relates the access profile of device A to that in device B with respect to the content objects in $LocatedList^n(A,B)$. The term k^{miss}_{AB} is initially set to 1 and adjusted after every evaluation in the parameter tuning phase.

3.2.2 Aggressive Prefetch

This phase is used to fetch the metadata information for objects that are optimistically predicted as ones that will be accessed in the near future. Device A does not contain any information about these objects and purely bases the assumption on the object profiles of device B. This is an aggressive approach based on the premise that the global workload in a personal area network varies with time due to a combination of changing user profiles, object updates and other factors. For example, a change in schedule might have forced a user to stop using a particular device for a long time. But when he returns back to it, he would still expect to find his recently accessed items from other devices. Therefore, even though the devices are heterogeneous and differ in the workloads, there is a need for them to optimistically fetch unknown objects from one another.

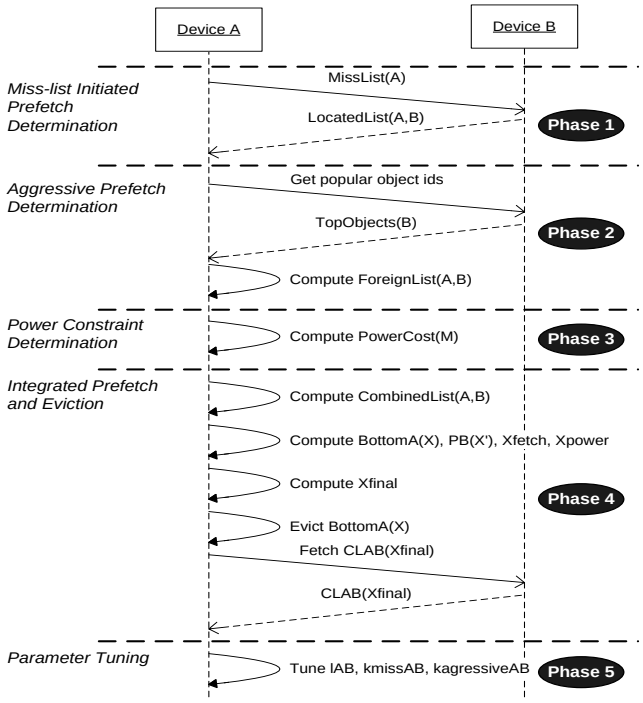


Figure 2: Content Exchange Protocol sequence diagram

Device A first queries device B for the data structure $TopObjects^n(B)$ where n is defined as in Section 3.2.1. Following this, device A first calculates the top objects

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

unknown to it from the $TopObjects^n(B)$, and stores it in the list *ForeignList(A,B)* that is defined as:

$$ForeignList(A,B) = TopObjects^n(B) - CurrentObjects(A) \quad (3)$$

Similar to the previous phase, device A makes an initial valuation of the content objects present in the *ForeignList(A,B)*. The access counters of the content objects in *ForeignList(A,B)* are re-calculated as:

$$AC(o) = k^{aggressive}_{AB} * AC(o) \quad (4)$$

The term $k^{aggressive}_{AB}$ represents the linear factor that relates the access profile of device A to that in device B with respect to the content objects in *ForeignList(A,B)*. The term $k^{aggressive}_{AB}$ is initially set to 1 and adjusted after every evaluation in the parameter tuning phase.

Note that we use two distinct parameters k^{miss}_{AB} and $k^{aggressive}_{AB}$ for the two phases described above, instead of just one. The rationale is that they represent two different content sets. The first parameter represents content objects seen by a device, while the second parameter represents content objects never seen by a device but potentially desirable. Our experimental evaluation shows different behaviors for the two sets, and validates our assumption.

3.2.3 Power Constraint Determination

In the power constraint determination phase, we project power considerations into the content exchange protocol. For devices with power constraints, the benefit of fetching too many objects will be offset by the power consumed during the exchange operation. For larger devices with an independent power source, such constraints may not apply. To take these issues into consideration, we define the power constraint terms p_A, ct_A and C_A for a particular device A in Table 1. For a device connected to an external power supply, p_A and ct_A are considered zero.

Power consumption in device A per byte transferred	p_A
Constant power dissipated per message in device A	ct_A
Fraction Battery Charge available in A	C_A

Table 1: Power characteristics of a content device

We estimate the power cost of the content distribution protocol in terms of power dissipated if a set of M objects were fetched from device B. The power dissipated is the total amount expended on both the devices in transferring the set M objects from device B to A. We use the linear power consumption model [35] for network usage. It is further normalized by the product of fractional battery charges available on the devices to ensure judicious usage of remaining battery charge. This is represented by the equation:

$$PowerCost(M) = \left[size(M) * (p_A + p_B) + |M| * (ct_A + ct_B)\right] / C_A * C_B \quad (5)$$

where size(M) is the sum of the sizes of the objects in M and |M| is the cardinality of the set. Note that we ignore the power cost incurred during the transfer of object metadata ias the metadata size is a couple of orders of magnitude less than that of the exchanged objects.

We assume that there is a user-imposed constraint P that defines the maximum amount of network power that can be consumed during the protocol for transferring objects from device B to device A. This parameter is used to throttle the number of objects actually fetched from device B in the protocol. The next phase integrates the information obtained till this point and uses that to actually determine the objects that will be prefetched from device B to device A.

3.2.4 Integrated Prefetch and Eviction

This is the core phase of the proposed protocol. In this phase, the goal is to predict the benefits of prefetching the candidate objects from the first two phases, and using that to determine the optimal number of objects that need to prefetched from device B, and evicted from device A to obtain the maximum relative benefit. The number of objects that result in the maximum predicted benefit is prefetched, and the power threshold P defined in Section 3.2.3 is used to throttle this value.

In the first step of this phase, we sort-merge the two lists *LocatedList(A,B)* and *ForeignList(A,B)* based on their access counters and call this list *CombinedList(A, B)*. We denote the set of the first X' objects in the sorted *CombinedList(A,B)* by $CL_{AB}(X')$ and obtain the least popular X objects in device A from the data structure *BottomObjects(A)* and denote it by $Bottom_A(X)$. We calculate the benefit as the increase in fraction of unit size accesses occurring to the fetched X' content objects as compared to that of the X content objects evicted. Assuming an independent reference model for the analysis,

$$PB(X') = f_B(CL_{AB}(X')) - f_A(Bottom_A(X)),$$
$$size(CL_{AB}(X')) - freespace(A) - size(Bottom_A(X)) \quad (6)$$

The term PB(X') defines the predicted benefit of evicting X content objects from device A and fetching X' content objects from device B. The value of the above function at any point X' can be calculated using the metadata information that device A has or obtained from device B. The function $f_A(obj)$ denotes the ratio of unit size accesses occurring on an object *obj* in device A to the total number of unit size accesses seen after its creation in device A. The function $f_A(O)$ for a set of objects *O* is extended as the sum of $f_A(obj)$ for each *obj* in the set *O*. The free disk space available on the device A is denoted by *freespace(A)*. We choose the value X' to be the lowest X' that maximizes the function PB(X'). We vary X' from 0 to $|CL_{AB}(X')|$ and observer PB(X') to compute the optimal in $O(|CL_{AB}(X')|)$ time. The optimal X' is denoted by X_{fetch}.

Next, we determine the maximum value of X'' such that the fetching of first X'' objects in the *CombinedList(A,B)* does not violate the power constraint equation below

$$PowerCost(CL_{AB}(X'')) - P \quad (7)$$

We call this value X_{power} as the number of content objects that can be fetched from device B under the given power constraints. We now compute the minimum of the two parameters X_{power} and X_{fetch} as the desired number of content objects to fetch from device B and denote this as X_{final}:

$$X_{final} = min(X_{power}, X_{fetch}) \quad (8)$$

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

Note that the user of a device can pin objects that will not be evicted in this protocol. This is especially relevant in cases where the device generates new content or updates existing content that has not yet been propagated elsewhere.

3.2.5 Parameter Tuning Phase

In the last parameter tuning phase, we set the values of the parameters l_{AB}, k^{miss}_{AB} and $k^{aggressive}_{AB}$ based on the evaluations done in the previous phases. These values are then used the next time content objects are fetched from device B to device A.

The parameter l_{AB} represents the factor we apply to the number of content objects we fetch from device B to device A. As mentioned before, initially, this value is 1 and we fetch metadata about l_{AB} *min(N(A), N(B))* objects from device B to device A. Subsequently, we optimize this by fetching metadata of a lesser number of content objects by setting a new value of l_{AB}:

$$l_{AB} = \frac{Xf_{etch}}{N(A)} \quad (9)$$

The rationale behind this tuning is that we are optimistically assuming that the number of content objects is going to be similar every time device A and B come into network proximity. We define a *run* of the protocol as a complete execution of the five phases of the context exchange protocol, when two devices A and B come into network proximity If the number of content objects fetched from device B drops for any reason between two protocol runs, we will get information about a greater number of content objects than is required but that will not affect the results. However, if the number of content objects fetched from device B increases for any reason between two protocol runs, we will get lesser information than what is required and this may affect the quality of the decision. Therefore we employ an empirically observed factor Δ to make sure that we are able to adjust to an increasing trend in the number of content objects fetched from device B. In either case, the parameter l_{AB} will adjust to the trend in the next protocol run.

The parameters k^{miss}_{AB} and $k^{aggressive}_{AB}$ represent the closeness of the object profiles between A and B for the purpose of the *Miss Initiated Prefetch* and *Aggressive Prefetch* determination phases, and can be tuned by employing simple linear regression methods. This tuning for the parameters k^{miss}_{AB} and $k^{aggressive}_{AB}$ is performed at a later stage than that for the parameter l_{AB}. If we use particular values for the two parameters k^{miss}_{AB} and $k^{aggressive}_{AB}$ to obtain content objects from device B, the parameters are tuned after we analyze the actual benefit of obtaining the content objects from device B.

More formally, the merged set $CL_{AB}(X)$ is partitioned into two sets $CL^{miss}_{AB}(X)$ and $CL^{aggressive}_{AB}(X)$ based on whether the content objects were initially present in the *LocatedList(A,B)* or the *ForeignList(A,B)*. We define an ordering on the protocol runs between the two devices A and B. The two parameters k^{miss}_{AB} and $k^{aggressive}_{AB}$ for the protocol run $r+1$ are obtained by tuning the values used in run r based on the fraction of unit size accesses to the prefetched objects corresponding to $CL^{miss}_{AB}(X)$ and $CL^{aggressive}_{AB}(X)$ respectively. The value of k^{miss}_{AB} is calculated as:

$$k^{miss}_{AB}(r+1) = p \times k^{miss}_{AB}(r) + (1-p) \times \frac{f_A(CL^{miss}_{AB}(X))}{f_B(CL^{miss}_{AB}(X))}$$

The term p is the fixed learning rate for the linear regression. The value for $k^{aggressive}_{AB}$ is calculated similarly. By modeling the linear regression, we implicitly assume that the first order variation between access profiles of a shared object on any two devices remains constant and subsequent orders can be ignored. This is a reasonable assumption because of the strong similarity of object profiles between devices in a personalized environment.

3.3 Metadata Management

This section deals with the issue of metadata management on a content device in terms of space and time costs. If metadata management takes up too much space, or too much of device CPU time, it could potentially adversely affect device operation. The content exchange protocol assumes the presence of the $TopObjects^N(A)$ and $BottomObjects^N(A)$ lists that indicate the most and least popular n devices in terms of access count, as well as a miss list to keep track of missed content objects. The miss list is stored as a simple linear hash structure indexed by the object identifer. The maintentance of the $TopObjects^N(A)$ and $BottomObjects^N(A)$ lists indicates the need to sort the object identifiers in these data structures. A first approach to maintain the sorted order would be to use a B-tree like data structure, but this ignores the fact that the typical size of the $TopObjects^N(A)$) and $BottomObjects^N(A)$ lists in our experiments is a small fraction of the total number of objects. Therefore, we use three data structures to sort the object identifiers in terms of access count. The first data structure is a small B-tree that stores the top most objects in terms of access count, the second is a hash list that stores the middle tier of objects in terms of access count, and the third is another small B-tree that stores the bottom most objects in terms of access count. The sizes of the B-trees are tuned to the typical size of content object metadata exchanged between two content devices (as represented by n in Section 3.2.1). Objects get moved from one data structure to another only when an increase in the access count of an object violates the ordering requirements of the three data structures. . Our architecture focused more on the content distribution protocol with the assumption that consistency is handled by a mixture of manual or above mentioned automated mechanisms.

3.4 Simulation Model

The evaluation of the proposed peer to peer content distribution consisted of running a global workload on simulated devices. The characteristics of the user behavior like relative popularities of objects and temporal variations in the global object dataset are reflected on the local device workloads. In a single user scenario, it is assumed that only one device is actively processing the workloads. This is a reasonable assumption considering the fact that a user is normally engaged in interactive processing only with one device.

The simulation environment makes a set of assumptions about the interaction between the content devices. First, peer devices go through connection and disconnection periods. The connection periods are utilized for fetching content objects from one content device to another. Note that some devices may connect to external networks for creating new content. Second, we assume that a device A fetches objects from B using *CD-PAN* only at instances where a user is transitioning from the device B to device A. This is a reasonable assumption as the chances of proximity of devices A and B are higher at transition points.

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

3.5 Workload Characteristics

CD-PAN is evaluated with both synthetic workloads, as well as workloads derived from real-world activity on content devices. The former are useful as they provide us with a characterization of the algorithm behavior with respect to various environmental parameters. Derived workloads, on the other hand, provide us with an expected behavior of the algorithm in an actual deployment environment.

3.5.1 Synthetic

For synthetic workloads, we generate multiple sets of documents, each set representing a particular type. For mobile devices in a personalized environment, web pages and multimedia files dominate the document types [31, 44**]**. We vary the proportion of accesses occurring to the document types for a particular combination of content device and user because different devices are optimized for different specialized applications [18]. The portion of the global workload corresponding to a particular document type is characterized by the following parameters: number of objects, popularity skew,size distribution, content update frequency, temporal Locality.

3.5.2 Derived

In addition to the synthetic workloads, we ran experiments on workloads whose characteristics were obtained from survey studies on real household data. The percent values in Table 3 represent the relative proportion of instances that the times spent on a particular device happen in the lives of those people in North America who own the listed devices and use them quite frequently [31]. From the information provided in [1, 32, 46, 26, 11] for web access patterns and [37, 38] for multimedia content, we obtain the parameter values for the two types of document benchmarks presented in Table 2. All the four devices are Wi-Fi enabled and are capable of communicating with each other.

	Web Browsing	**Multimedia**
Zipf	0.75	0.4
Mean size	10 KB	500 KB
Size Distribution	Lognormal	Lognormal + Pareto for long tail
Request Rate per usage hr	100	30
New/update frequency per day	25%	10%

Table 2: Characteristics of usage patterns for content activity

3.6 Comparable Schemes

We compare *CD-PAN* with a *Raw Prefetch* scheme (*RP)* where a device A does not take into account the popularity of the content objects in B while deciding which content objects to fetch from B to A. It is assumed that the device A uses only the miss-list to determine the objects to fetch from B. The objects to be evicted are chosen from the least popular set of objects as described in *CD-PAN*. *RP* allows us to evaluate the design decision to base our fetch policy on the popularity of content objects in a peer device.

Home PC (Device A) Capacity – 60GB		**PDA/Smartphone (Device B) , Capacity 128 MB**		**iPOD/MP3 player (Device C) Capacity 10 GB**		**TV (DVR) (Device D) Capacity (20 GB)**	
Daily usage time distribution		**Daily usage time distribution**		**Daily usage time distribution**		**Daily usage time distribution**	
< 30 mins	8%	< 15 mins	33%	< 10 mins	8%	< 30 mins	6%
30 – 1 hrs	13%	15- 30 mins	24%	10 – 30 mins	27%	30 – 1hr	13%
1 – 2 hrs	27%	31 – 1 hr	18%	30 - 1 hr	33%	1 – 2 hrs	31%
2 - 3 hrs	18%	1 – 2 hrs	15%	1 – 2 hr	24%	2 – 3 hrs	24%
3 – 4 hrs	14%	2 – 3 hrs	5%	2 -3 hr	6%	3 – 4 hrs	13%
Over 4	18%	3 – 4 hrs	2.5%	Over 3 hrs	2%	Over 4 hrs	12%
		Over 4	2.5%				

Table 3: Characteristics of content devices in a PAN

Another scheme that we compare against is the *On-demand Fetch* scheme (*ODF*) that assumes the capability of fetching a content object from an external source on a miss. While this may not be practical in terms of latency and monetary cost, ODF gives an idea of how traditional cache replacement techniques would have performed for the workloads. To make the comparison fair, the popularity of objects in a device is computed only at transition periods and the eviction costs are based on this information. LFU is used as the replacement algorithm.

Finally, we compare *CD-PAN* to an *Optimal* offline prefetching scheme *(OPT)*. OPT assumes the same model as the CD-PAN and the RP schemes with respect to prefetching of objects at only transition points. Since the actual benefit of prefetching content objects (as opposed to the predicted benefit in *CD-PAN)* can be obtained by looking at the future accesses, we use that to find the optimal number of objects to be prefetched and evicted. Note that OPT is an upper bound on the size normalized hit percentage performance of CD-PAN and RP. ODF assumes a different architecture and could perform better than OPT in some cases.

4 Experimental Results

This section presents experimental results for both synthetic and derived workloads. In the former set of experiments, we vary the parameters, and in the latter we use the workload characteristics listed in section 3.5.2. We have conducted extensive experiments, and the interesting results are presented for brevity. We use two metrics for comparison, *size normalized hit percentage* which is the the percentage of requests per kilo byte that hit in the host device and *power normalized hit ratio*, the ratio of the *size normalized hit percentage* to the average power cost of prefetching.

4.1 Synthetic Workloads

These experiments are conducted for a set of 4 devices: one large device (with 100GB storage), and three smaller homogenous devices (each 100MB storage). The average user session time in the large device is slightly longer than in the smaller devices. A default aging fraction value of 0.6 was chosen. The aging fraction is the fractional value that is used to

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

weigh the access counters of objects between two consecutive user sessions on a device.

The power threshold P (defined in Section 3.2.3) chosen for a device is the amount of power expended if 40% of the device capacity were to be transmitted. We set this bound after many trials of experiments that showed that this bound does not degrade performance while providing a reasonable bound on power dissipation. The linear regression parameter p (Section 3.2.5) was set the value of 0.8 by default in our experiments. We observed that the results did not vary a lot by changing this value because the statistics were gathered after the devices reached steady states with respect to the observed hit ratios.

4.1.1 Variation in document popularity

Web workloads are known to exhibit sharp popularity distributions with majority of accesses going to a small minority of documents [32] whereas multimedia workloads follow much more uniform distribution [37, 38]. One of the standard techniques to vary the popularity skew is by using different values of the zipf parameter α.

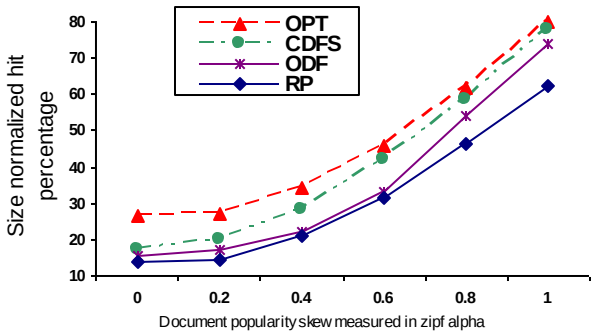


Figure 3: Results for different values of zipf parameter

Figure 3 shows the performance of the different content distribution schemes for varying values of Zipf parameter α. As α increases from 0 to 1, the popularity distribution of documents changes from uniform to skewed. In tandem, the observed size normalized hit percentages increase because of the reduction in working set sizes. We observe that the size normalized hit percentages for *CD-PAN* are consistently better than those for *RP* and *ODF*, and close to optimal in almost all cases. The improvement is small for smaller values of α because *CD-PAN* cannot effectively distinguish between popular and unpopular objects but it increases with α. For example, *CD-PAN* shows an improvement of 11% and 25% over *RP* for values of α = 0.0 and 1.0 respectively. As the popularity distribution becomes more skewed, the accuracy of predicting the frequently accessed documents to be prefetched and rarely accessed documents to be evicted improves. We found that this was true by observing the ratios of the predicted benefit in *CD-PAN* to the actual benefit of prefetching observed at the end of the user session. The ratio was much larger for skewed popularity distributions (zipf α = 1) at about 0.95, than for relatively uniform distributions (zipf α = 0.2) at about 0.82.

4.1.2 Variation in temporal locality

We study the effect of temporal locality in the workloads on the performance of the schemes. We characterize the temporal locality of a workload by the average distance between consecutive requests to an object [45] when all the requests are placed on a LRU stack. As the distance decreases, the requests exhibit greater temporal effects.

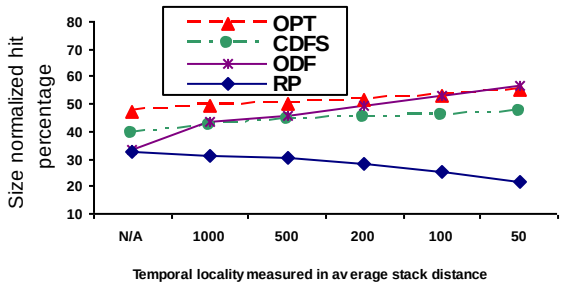


Figure 4: Size normalized hit percentages for different temporal locality behaviors

Figure 4 shows the size normalized hit percentages of the different schemes for an independent reference workload and when the Average Stack Distance (ASD) is varied from 50 to 1000. We used the probabilistic sliding window algorithm in the Surge Benchmark [45] to synthesize the workloads with temporal locality. From Figure 4 we observe that *CD-PAN* performs considerably better than *RP*. The improvement is more dramatic than observed in Section 4.1.1 for an independent reference. For workloads with good temporal locality there is a greater need for a content distribution scheme to consider the recent popularities of documents in peer devices. *CD-PAN* addresses this requirement in the aggressive prefetch phase (Section 3.2.2) while *RP* does not. We also notice that the performance of *ODF* increases with locality because of better cache utilization.

4.1.3 Variation in content updates

In this section we study the effect of dynamically added content to the current set of documents in the global workload. Web workloads especially show significant amount of updates to existing documents and newly created documents with time[64]. In the current study, we treat an update to a document as the creation of a new one that needs to be fetched in its entirety by a device looking to replace a stale copy. The addition of new documents and their subsequent rate of accesses are controlled such that the overall workload characteristics like popularity distribution of documents and temporal locality are preserved.

Figure 5 shows the performance of the content distribution schemes with variance in the frequency of new content added from 1% to 20%. The large device was chosen as the content creation device. The size normalized hit percentages are a function of the working set which is kept constant by using an independent reference model and the same Zipf parameter α = 0.7. *CD-PAN* outperforms *RP* by about 25% for 1% content creation. The performance gap increased with frequency of new content with a percentage improvement of nearly 90% when the content creation frequency is 20%. The performance of *ODF* is independent of the amount of new data created.

We however observe the declining trend for *CD-PAN* with increase in new content. This is to be expected because new content objects take time to evict existing old content objects in a device. However, we noticed that the performance of CD-PAN for high percentages of content creation can be improved by using a lower *aging fraction*. An aging fraction of around 0.8 performed the best for content creation frequency of 1% whereas a value around 0.5 proved to be the best aging fraction for 20% content creation. With a lower aging fraction the

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

access counter values of newly created objects increase at a relatively faster rate with every access and quickly dominate the existing objects that are not being accessed at the same rate. This works well for workloads that have high content updates.

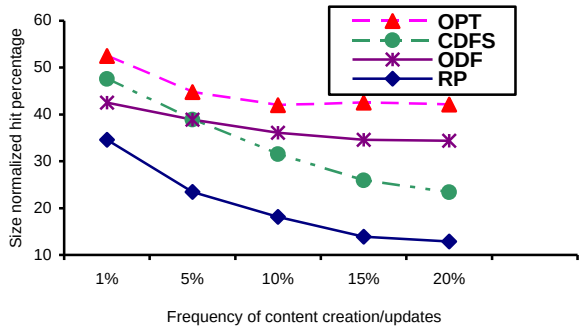


Figure 5: Size normalized hit percentages with frequency of content creation/updates

4.1.4 Varying content profiles

The popularity distribution of shared documents can vary a lot amongst heterogeneous devices [31, 44]. A content device can show specific application characteristics that ignore document types popular in other content devices.

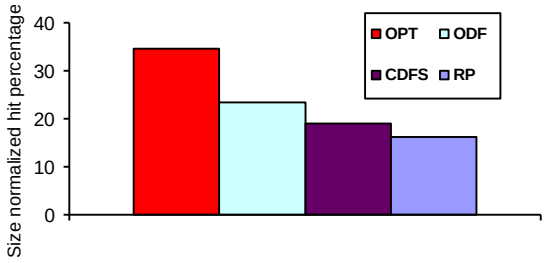
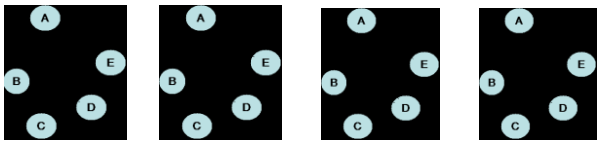


Figure 7 shows the size normalized hit percentages for the different schemes in device A

To illustrate this, we ran experiments with varying popularity distributions of the three document types X, Y and Z on devices A,B and C. Device A accessed a document of type X with probability 0.8 and Y with 0.2. Devices B and C accessed the document sets (Y,Z) and (Z,X) with similar probabilities respectively. We observe from Figure 7 that *CD-PAN* outperforms *RP* by about 17%. The mismatch in relative popularities of the document types between any two devices causes the aggressive phase to perform poorly. *CD-PAN* adapts to this behavior by reducing the number of candidate documents from the aggressive prefetch phase. This is highlighted by steady-state values for $k^{aggressive}_{AB}$ at about 0.27 compared to k^{miss}_{AB} at about 3.3.

4.1.5 Communication variation

In this section, we study the performance of the schemes with varying capabilities of the devices to communicate with each other and hence, to prefetch data for disconnected operations. Figure 8 shows an increasing degree of connectivity between device nodes as the cut increases from 1 to 4. An edge denotes that the two devices can communicate with each other. Note that a transition of user work can still directly occur between two devices that are not connected in the graph in figure 8. Device A is the large device that introduces new content at 10% frequency.



Cut = 1 Cut=2 Cut = 3 Cut 4

Figure 8: Different communication capability scenarios

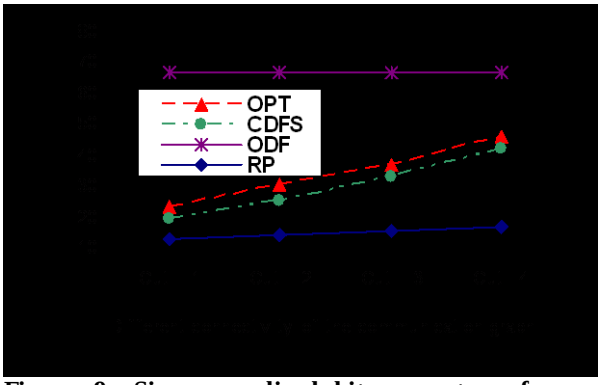


Figure 9: Size normalized hit percentage for varying communication capabilities

We observe the size normalized hit percentages of the schemes on device C for the four communication capability cases in Figure 9. We observe that *CD-PAN* performs better than *RP* in all the cases and the improvement is more when the devices have better communication capabilities. In the first three cases, C is not connected to the content generator and has to rely on other devices B, D and E to retrieve the data it needs. We observe that when the connectivity with the source device is low (Cut =1), the size normalized hit percentage of the *CD-PAN* scheme on device C is low (19%) in spite of a single hop connectivity with B which can communicate directly with A. However, when a single edge is added to the graph between A and E, the size normalized hit percentage jumps to 27% and reaches 42% for a fully connected graph. This shows that two devices tend to correlate their object profiles faster when the number of common devices that they can communicate with is higher. Since the access counter of an object is reset to zero in device B when an object is transmitted from device A to B in our protocol, it takes longer for an object to be transmitted between two devices that cannot communicate directly in the communication graph.

4.1.6 Varying power thresholds

Different devices have different tolerance values for power dissipation [39, 10] depending on the strength of the power source. In this section, we study the sensitivity of performance of our scheme when the power thresholds applied in the *Power Constraint* phase were varied.

If the power constraint is P, then the fraction of the capacity of the device that can be transmitted with the given constraint is defined to be the *fractional power threshold*. Figure 10 shows the size normalized hit percentages when the fractional power threshold is varied from 1 to 0.02. We observe that *CD-PAN* performs steadily till a threshold of around 0.1 and degrades thereafter. With stringent power thresholds, the restriction on the number of objects exchanged becomes larger and reduces the effectiveness of *CD-PAN*. The rapid degradation after a threshold of 0.1 indicates that the average number of objects prefetched in the experiment shown was around 10% of the

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

total number of objects residing in the small simulated device. We found a similar behavior for different varying workload parameters studied in Sections 4.1.1 to 4.1.5 as well. The metric *power normalized hit* in our scheme shows an increasing trend with stricter power thresholds as the most popular objects have a higher access benefit while the power costs are uniform over all content objects.

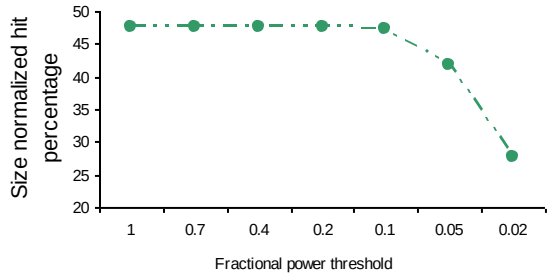


Figure 10: Size normalized hit percentage for the CD-PAN scheme with varying power thresholds

4.2 Derived Workloads

We now present the results of using the real-world derived usage patterns documented in Table 3 (section 3.5.2). We consider two devices: a SmartPhone and a iPOD/MP3 player that synchronize with a home PC content generator.

Figure 11 shows the size normalized hit percentages for the two devices. For the SmartPhone, the size normalized hit percentage for *CD-PAN* is around 30%, which is about 3 times better than that for *RP*. We found that the good performance of *CD-PAN* is because of the dominant contribution from the aggressive prefetch phase which speculatively fetches dynamically created offline web content and multimedia files from the home PC.

CD-PAN performed much better than *ODF* because the limited storage in a SmartPhone is not adequate to act as a cache for the working set of the web browsing and multimedia files. Therefore the LFU eviction algorithm used by ODF resulted in excessive fetching and eviction similar to the thrashing phenomenon seen in virtual memory. The CD-PAN scheme does not suffer from inadequate storage size because it makes the best use of available space by storing the frequently accessed objects and evicting them only when they are to be replaced with more popular ones. For the iPOD/MP3 Player, the size normalized hit percentages were higher for all the schemes. *CD-PAN* outperformed *RP* by about 25% for this metric. The improvement was limited by the fact that the primary music files workload in the iPOD/MP3 player does not show much dynamic variation and the relatively large storage space of the music device is sufficient to store the working set of frequently accessed music files.

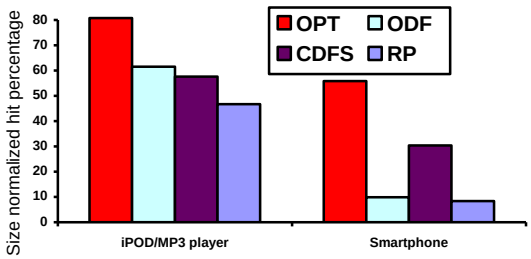


Figure 11: Size normalized hit percentage for the two mobile devices, iPOD/MP3 player and Smartphone

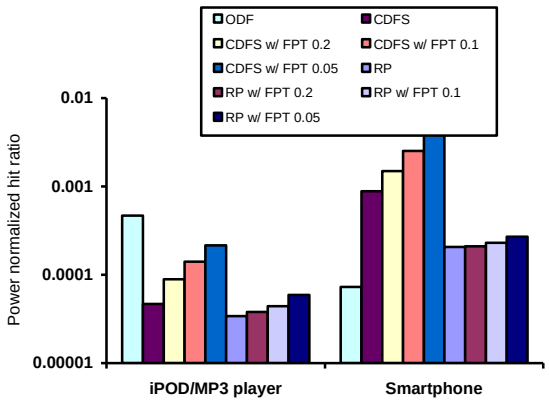


Figure 12: Power normalized hit ratios for the mobile devices, iPOD/MP3 player and Smartphone

We also ran experiments to study the effect of power constraints on these devices. We found that in general the performance of *CD-PAN* degraded with stricter thresholds on power consumption. But, as Figure12 shows, with stricter power thresholds indicated by decreasing fractional power threshold (FPT) values, there is a sharp increase in the power normalized hit ratios and the increase in ratios are much higher for *CD-PAN* than for *RP*.

5 Related Work

Prefetching of objects is a very common technique used in web domains and file systems to improve latency of requests. The reference patterns of clients at the proxies and the servers have been used as an effective source of information to drive prefetching [65, 3, 8, 34]. Most of the studies use a collection of web client workloads and study how effectively future web accesses can be predicted from past accesses.

A number of studies have investigated prefetching between web servers and proxies as well [58, 14]. Peer-to-peer caching of data has been used as a cooperative web caching technique where caches in peers share internet objects among themselves [4,51, 6,54]. Cooperative web caching is found to be useful in improving hit ratio in a group of small organizations. Outside the web contexts, prefetching as a latency-reducing technique has been explored in file and memory systems. Several studies have investigated application-controlled prefetching in the file system area [46, 23, 22].

In the area of distributed file systems with frequent disconnected operations, several studies have proposed mechanisms to automate the process of file hoarding. Griffioen et al [27] proposed a file prefetching scheme based on graph-based relationships that track frequency of accesses within a look-ahead window size. An analytical approach based on the cost-benefits of read-ahead buffering and prefetching was proposed in [16]. The use of the last successor model for file prediction and more elaborate techniques based on pattern-matching and context modeling were proposed in [22,59,61, 20]. There is also a significant body of work on using transparent compiler-directed approaches [62] and application-level hints for improved prefetching [50].

In the context of wireless data dissemination networks for mobile devices, previous works [29, 56, 34, 19, 36, 52] have investigated the problems of cache replacement and prefetching

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

individually but few efforts have considered the aspects together to enhance data availability in mobile devices [24]. Prefetching targeted for mobile users in a wide area wireless network assume some sort of infrastructure deployment [63,39] which is used by the prefetching algorithm based location, route and speed. Like CD-PAN, [15] takes power considerations into account when deciding to cache to fetch data. It differs in that it is a distributed file system with a global namespace and thus requires a central repository. It is similar to CD-PAN in that it takes space and energy considerations into account while sharing content data but differs in that the ensembles operate only in disconnected mode and requires a coordinator to direct data sharing.

6 Conclusion

The key contribution of this paper is a protocol for the automatic organization and transfer of content objects across heterogeneous and weakly connected devices in order to maximize the availability of frequently accessed content objects in every device. In order to avoid misses, a content device uses *CD-PAN* to fetch content objects in the background from peer devices without user intervention. Experiments show that *CD-PAN* outperforms *RP* in all synthetic and derived workload experiments. The performance improvement tended to increase with increase in popularity distribution skew temporal locality and frequency of content creation/updates. We observed that *CD-PAN* performs really well when the pair-wise device communication capabilities are higher. We also found that *CD-PAN* adapts well to the decreasing tolerance levels for power and metadata.

References

[1] Atul Adya, Paramvir Bahl and Lili Qiu. Analyzing the browse patterns of mobile clients. In *Proceedings of the ACM SIGCOMM Workshop on Internet Measurement. 20*01

[2] Anand Balachandran, Goeffrey M. Volker, Paramvir Bahl and P. Venkat Rangan. Characterizing user behavior and network performance in a public wireless LAN. In *Proceedings of the ACM SIGMETERICS International Conference on Measurement and Modeling of computer systems,* 2002.

[3] Azer Bestavros and Carlos Cunha. Server-initiated document dissemination for the www. *IEEE Data Engineering Bulletin*, September 1996

[4] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz, Kurt J. Worrell. A hierarchical internet object cache. In *Proceedings of the 1996 USENIX Annual Technical Conference,* 1996

[5] Antony Rowstron and Peter Drushel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the ACM Symposium on Operating Systems Principles, 2001*

[6] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative web caching. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles,* 1999

[7] Balachander Krishnamurthy, Craig Wills and Yin Zhang. On the use and performance of content distribution networks. In *Proceedings of the ACM Workshop on Internet Workshop,* 2001

[8] Carlos Cunha and Carlos F. B Jaccoud. Determining www user's next access and its application to prefetching. In Proceedings of ISCC' 97: *The Second IEEE Symposium on Computers and Communications*, July 1997

[9] Carlos Cunha, Azer Bestavros and Mark Crovella. Charecteristics of WWW client-based traces. Technical Report TR-95-010, Boston University, 1995

[10] Carla Schlatter Ellis. The case for higher-level power management, In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems,* 1999.

[11] Dennis Fetterly, Mark Manasse, Mark Najork and Janet L. Wiener. A large-scale study of the evolution of web pages, , Software : Practice and Experience 2004

[12] Dahlia Malkhi and Doug Terry. Concise version vectors in WinFS. In *Symposium on Distributed Computing,* 2005

[13] Daniel Peek and Jason Flinn, EnsemBlue : Integrating distributed storage and consumer electronics. *To appear in the USENIX Symposium on Operating Systems Design and Implementation,* 2006

[14] Evangelos P. Markatos and Catherine E. Chronaki. A top-10 approach to prefetching on the web. Technical report No. 173, ICS-FORTH, August 1996

[15] Edmund B. Nightingale and Jason Flinn, Energy-efficiency and storage flexibility in the blue file system. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation,* 2004

[16] Elizabeth Shriver, Christopher Small and Keith A. Smith. Why does file system prefetching work?. In *Proceedings of the 1999 USENIX Annual Technical Conference,* 2001

[17] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris and Ion Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the ACM Symposium of Operating Systems Principles,* 2001

[18] Guanling Chen and David Kotz. A Survey of context-aware mobile computing research. Dartmouth Computer sScience Technical Report TR2000-381

[19] Guohong Cao. Proactive power-aware cache management for mobile computing systems. *IEEE Transactions on Computers*. 2002

[20] Geoffrey H. Kuenning and Gerald J. Popek. Automated hoarding for mobile computers. In *Proceedings of the 16th ACM Symposium on Operating systems Principles,* 1997

[21] Gene T. J Wuu and Arthur J. Bernstein. Efficient solutions to the replicated log and dictionary problem. In *Proceedings of the ACM Symposium on Principles of distributed computing,* 1984

[22] Hui Lei and Dan Duchamp. An analytical approach to file prefetching. In *Proceedings of the 1997 USENIX Annual Technical Conference,* 1997

[23] Hugo Patterson, Garth A. Gibson, Eka Ginting, Daniel Stodolsky, and Jim Zelenka. Informed prefetching and caching. In *Proceedings of 15th ACM Symposium on Operating System Principles,* December 1995.

[24] Huaping Shen, Mohan Kumar, Sajal K. Das and Zhijun Wang. Energy-efficient caching and prefetching with data consistency in mobile distributed systems. *In Proceedings of the International Parallel and Distributed Processing Symposium*. 2004

[25] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu and Jorjeta Jetchava. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the International Conference on Mobile Computing and Networking,* 1998

[26] Junghoo Cho and Hector Garcia-Molina. The Evolution of the Web and Implications for an incremental crawler, In *Proceedings of the International Conference on Very Large Data Base,* 2000

[27] James Griffioen and Randy Appleton. Reducing file system latency using a predictive approach. In *USENIX Summer Technical Conference,* 1994

[28] James J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. In *Proceedings of the ACM Transactions on Computer Systems,* 1992

[29] Jianliang Xu, Qinglong Hu, Wang-Chien Lee and Dik Lun Lee. An Optimal Cache Replacement Policy for Wireless Data Dissemination under Cache Consistency. *In Proceedings of the 30th International Conference on Parallel Processing.* 2001

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825

[30] Karin Petersen, Mike J. Spreitzer, Douglas B. Terry Marvin M. Theimer and Alan J. Demers. Flexible update propagation for weakly consistent replication. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, 1997
[31]http://www.us.31g.com/RutUS_prod/Documents/12/Consumer%20Convergence%20study.pdf
[32] Lee Breslau, Pei Cao, Li Fan, Graham Phillips and Scott Shenkar. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of IEEE INFOCOM*, 1999
[33] Lei Guo, Songging Chen, Zhen Xiao, Xiaodong Zhang. Analysis of multimedia workloads with implications for internet streaming. In *Proceedings of the International Conference on World Wide Web*, 2005.
[34] Li Fan, Pei Cao, Wei Lin and Quinn Jacobson. Web Prefetching between low-bandwidth clients and proxies: potential and performance. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 1997
[35] Laura Marie Feeney, Martin Nilsson. Investigating the energy consumption of a wireless network interface in an Ad hoc networking environment. *In IEEE Proceedings of INFOCOM*, 2001
[36] Liangzhong Yin, Guohong Cao, Chita Das, and Ajeesh Ashraf. Power-aware prefetch in mobile environments. *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002
[37] Maureen Chesire, Alec Wolman, Goeffrey M. Voelker and Henry M. Levy. Measurement and Analysis of a Streaming-media Workload. In *Proceedings of the USENIX Conference on Internet Technologies and Systems*. *20* 01
[38] Minaxi Gupta and Mostafa Ammar. A Novel Multicast Scheduling for multimedia servers with variable access patterns. In *Proceedings of the IEEE International conference on Communications*. 2003
[39] Maria Papadopouli and Henning Schulzrinne. Design and Implementation of a Peer-to-Peer Data Dissemination and Prefetching Tool for Mobile Users. *First NY Metro Area Networking Workshop*, 2001
[40] Marvin McNett and Geoffrey M. Voelker. Access and Mobility of wireless PDA users. In *ACM SIGMOBILE Mobile Computing and Communications Review*, 2005
[41] Michael N. Nelson, Brent B. Welch, John K. Ousterhout. Caching in the Sprite network file system. *ACM Transactions on Computer Systems*, 1988
[42] MagicSync: http://www.pdatopsoft.com/SmartPhones/MagicSync-Lite-(SmartPhone)
[43] Nalini Belaramani, Mike Dahlin, Lei Gao, Amol Nayate, Arun Venkataramani, Praveen Yalagandula, and Jiandan Zheng. PRACTI Replication. In *Proceedings of the Symposium on Networked Systems Design and Implementation*, 2006
[44] http://www.44ab.tkk.fi/~hverkasa/verkasalo_hmr_2006.pdf
[45] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of SIGMETRICS 1998*
[46] Paul Barford, Azer Bestavros, Adam Bradley, Mark Crovella. Changes in Web Client Access Patterns. World Wide Web Journal 1999
[46] Pei Cao, Edward W. Felton, Anna R. Karlin, and Kai Li. Implementation and performance of integrated application-controlled file caching, prefetching and disk scheduling. In *TOCS*, November 1996
[47] PocketMirror : http://www.chapura.com/
[48] Richard G. Guy, John S. Heidemann, Wai Mak, Thomas W. Page Jr., Gerald J. Popek, Dieter Rothmeier, Implementation of the Ficus replicated file system. In *USENIX Summer Conference*, 1990
[49] Rivka Ladin, Barbara Liskov, Liuba Shrira and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Transactions on Computer Systems*. 1992
[50] R. Hugo Patterson, Garth A. Gibson, Eka Ginting, Daniel Stodolsky and Jim Zelenka. Informed prefetching and caching. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP)*, 1995.
[51] Renu Tewari, Michael Dahlin, Harrick M. Vin and Jonathan S. Kay. Design considerations for distributed caching on the internet. Technical Report CS98-04, University of Texas at Austin, 1998
[52] Swarup Acharya. Broadcast disks: Dissemination-based data management for asymmetric communication environments. *In PHD Dissertation, Brown University*, 1998
[53] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 2004
[54] Syncexpress : www.syncdata.it/syncexpress.html
[54] Sitaram Iyer, Antony Rowstron and Peter Druschel. Squirrel : A decentralized peer-to-peer web cache. In *Proceedings of the Principles of Distributed Computing*, 2002
[55] Shudong Jin and Azer Bestavros. Temporal locality in web request streams: sources, characteristics and caching implications. In *Proceedings of the ACM SIGMETRICS*, 2000
[56] Savvas Gitzenis and Nicholas Bambos. Power Controlled Data Prefetching/Caching in Wireless Packet Networks. In *IEEE Proceedings of INFOCOM 2002*
[57] Sumeet Sobti, Nitin Garg, Chi Zhang, Xiang Yu, Arvind Krishnamurthy and Randolph Y. Wang. PersonalRAID : Mobile Storage for Distributed and Disconnected Computers. *In Proceedings of the Conference on File and Storage Technologies*, 2002
[58] Thomas M. Kroeger, Darrell D. E. Long, and Jeffrey C. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. In *Proceedings of USENIX Symposium on Internet Technology and Systems*, December 1997
[59] Thomas M. Kroeger and Darrell DE Long. The case for efficient file access pattern modeling. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*,1999
[60] Thomas Kunz, Thomas Berry, James P. Black and Hugh M. Mahoney. WAP Traffic: Description and Comparision to WWW Traffic. In *Proceedings of International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2000
[61] Thomas M. Kroeger and Darrell DE Long. Design and implementation of a predictive file prefetching algorithm. In *Proceedings of the 2001 USENIX Annual Technical Conference*, 2001.
[62] Todd C. Mowry, Angela K. Demke, Orran Krieger. Automatic compiler-inserted I/O prefetching for out-of-core applications. In *Proceedings of the 1996 Symposium on Operating Systems Design and Implementation*, 1996
[63] Tao Ye, H.-Arno Jacobsen and Randy Katz. Mobile Awareness in a wide area wireless network of info-stations. In *Proceedings of Mobicom*, 1998
[64] Venkata N. Padmanabhan and L. Qiu. The content and access dynamics of a busy web site: Findings and Implications. In *Proceedings of the ACM SIGCOMM*, 2000
[65] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using Predictive prefetching to improve world wide web latency. *ACM SIGCOMM Computer Communication Review*, July 1996

Digital Object Identifier: 10.4108/ICST.WICON2008.4825
http://dx.doi.org/10.4108/ICST.WICON2008.4825