A Collaborative Replication approach for Mobile-P2P networks

Anirban Mondal[1] Sanjay Kumar Madria[2] Masaru Kitsuregawa[1]

[1] Institute of Industrial Science
University of Tokyo, Japan
{anirban,kitsure}@tkl.iis.u-tokyo.ac.jp

[2] Department of Computer Science
University of Missouri-Rolla, USA
madrias@mst.edu

Abstract

This paper proposes CADRE (Collaborative Allocation and Deallocation of Replicas with Efficiency), which is a dynamic replication scheme for improving the typically low data availability in *dedicated* and *cooperative* mobile ad-hoc peer-to-peer (M-P2P) networks. In particular, replica allocation and deallocation are collaboratively performed in tandem to facilitate effective replication. Such collaboration is facilitated by a hybrid super-peer architecture in which some of the mobile hosts act as the 'gateway nodes' (GNs) in a given region. GNs facilitate both search and replication. The main contributions of CADRE are as follows. First, it facilitates the prevention of 'thrashing' conditions due to its collaborative replica allocation and deallocation mechanism. Second, it considers the replication of images at different resolutions to optimize the usage of the generally limited memory space of the mobile hosts (MHs). Third, it addresses fair replica allocation across the MHs. Fourth, it facilitates the optimization of the limited energy resources of MHs during replication. Our performance evaluation demonstrates that CADRE is indeed effective in improving data availability in M-P2P networks with significant reduction in query response times and low communication traffic during replication as compared to a recent existing scheme as well as a baseline approach, which does not consider any replication.

Keywords: Mobile Peer-to-peer computing, dynamic replication, collaborative replica allocation, data availability, fair replication

1 Introduction

In a Mobile ad-hoc Peer-to-Peer (M-P2P) network, mobile hosts (MHs) interact with each other in a decentralized P2P fashion without using base stations. The proliferation of mobile computing technology (e.g., laptops, PDAs, mobile phones) coupled with the ever-increasing popularity of the Peer-to-Peer (P2P) paradigm (e.g., Kazaa [20]) strongly motivate M-P2P network applications. M-P2P applications are essentially aimed at facilitating mobile users in obtaining information on-the-fly from other mobile users in a P2P manner. Such P2P interactions are generally not supported by existing mobile communication infrastructures.

This work focusses on improving the performance (i.e., query response time and data availability) of *dedicated* and *cooperative* M-P2P networks by means of a collaborative approach for replica allocation and deallocation. Some M-P2P application scenarios for dedicated networks follow. Suppose a group of agriculturists[1] are performing studies of crops, crop diseases and soil fertility in a remote agricultural area, where communication infrastructures (e.g., base stations) do not exist. They need to share snapshots (pictures or video-clips) of the crops to determine the progression of diseases in the crops to decide upon the appropriate amounts of fertilizers and pesticides to administer to the crops.

Given the typically large size of such an agricultural area, suppose there is a chief agriculturist, who supervises the working of the agriculturists in a specific region of the area, to facilitate efficient data sharing. Thus, we can visualize the agricultural area as a set of regions, each region having a chief agriculturist, who is in-charge of the agriculturists working in that region. Understandably, agriculturists within the same region may wish to share data among themselves (i.e., *local querying*) or they may wish to share data with those, who are working in other regions (i.e., *remote querying*). Each chief agriculturist could keep track of information (e.g., images, video-clips) stored at MHs that are within its region, thereby facilitating local querying. Furthermore, chief agriculturists in different regions of the agricultural area could interact with each other to

[1] Agriculturists are agricultural scientists, who specialize in all aspects of crop cultivation.

facilitate remote querying.

In the same vein, a group of archaeologists, who are performing excavations in a remote area of Egypt, may need to share images (e.g., artefacts and ancient maps found), with each other by means of mobile devices because they are working towards the same collaborative goal.

Now suppose agriculturist X in region A requests an image, and obtains a replica of the image (from agriculturists of other regions). After a while, X deallocates (deletes) this replica, but now another agriculturist Y in region A requests the same image, hence the replica has to be allocated again to region A. Since images are relatively large in size, multiple allocations and deallocations of the same replica at the same region tax the generally limited bandwidths and energy resources of MHs. This may lead to undesirable *'thrashing conditions'*, where MHs spend more bandwidth and energy on allocating and deallocating replicas than on answering queries.

Notably, our target applications mainly concern slow-moving objects e.g., agriculturists and archaeologists moving in a remote area. Moreover, M-P2P ephemerality emphasizes the need for queries to be answered in a fast and timely manner, thereby necessitating query *deadlines*.

Data availability is typically low in M-P2P networks due to frequent network partitioning arising from user movement and/or users switching 'off' their mobile devices. (Data availability is less than 20% even in a wired environment [31].) Hence, several dynamic replication schemes [16, 18, 26, 28, 34] have been proposed for improving data availability in M-P2P networks. However, while these existing schemes perform *collaborative* replica allocation, replica deallocation is done only *locally* by each MH. Suppose a data item d has a high access frequency at MHs M_i and M_j, while being rarely accessed at an MH M_k. Under the existing schemes, M_k would deallocate d. But, since d is actually a 'hot' data item, d may be once again allocated to M_k after some-time, which may lead to undesirable *'thrashing'* conditions. Hence, both replica *allocation* and *deallocation* should be performed *collaboratively* in tandem with each other to prevent thrashing.

Moreover, existing schemes do not consider *fairness in replication* since they allocate replicas solely based on the read/write access probability of any given data item d without considering the origin of queries for d. Thus, these schemes would regard d as 'hot' even if only a single MH

issues a large number of (read) queries for d, thereby possibly creating several replicas of d. This runs contrary to the principle of *fairness* in serving multiple peers' requests, which is an important requirement in all P2P systems. Furthermore, MHs generally have resource constraints such as limited memory space and energy resources. Such constraints should be adequately addressed for any M-P2P replication scheme to work effectively in practice.

This paper proposes CADRE (Collaborative Allocation and Deallocation of Replicas with Efficiency), which is a dynamic replication scheme for improving the typically low data availability in a *cooperative* mobile ad-hoc peer-to-peer (M-P2P) network. In particular, replica allocation and deallocation are collaboratively performed in tandem to facilitate effective replication. Such collaboration is facilitated by a hybrid super-peer architecture in which some of the mobile hosts act as the 'gateway nodes' (GNs) in a given region. GNs facilitate both search and replication.

The main contributions of CADRE are as follows:

1. It facilitates the prevention of 'thrashing' conditions due to its collaborative replica allocation and deallocation mechanism.

2. It considers the replication of images at different resolutions to optimize the usage of the generally limited memory space of the MHs.

3. It addresses fair replica allocation across the MHs.

4. It facilitates the optimization of the limited energy resources of MHs during replication.

Our performance evaluation demonstrates that CADRE is indeed effective in improving data availability in M-P2P networks with significant reduction in query response times and low communication traffic during replication as compared to a recent existing scheme (i.e., the E-DCG+ approach [16]). We also compare the performance of CADRE with a baseline approach designated as NoRep, which does not perform any replication. The results of our experiments indicate that CADRE outperforms E-DCG+ due to several reasons such as its collaborative replica allocation and deallocation mechanism, its fair replica allocation scheme, its thrashing prevention scheme,

its consideration of MH energy and its load-conscious replica allocation scheme. Furthermore, CADRE and E-DCG+ both outperform NoRep since NoRep does not perform any replication.

Our experiments show that CADRE is indeed capable of effectively preventing thrashing by adjusting a particular threshold replication parameter. As the workload skew increases, the performance gap between CADRE and E-DCG+ also increases due to the need for replication becoming more prominent with increased load-imbalance. As the replica allocation period increases, replica allocation traffic decreases dramatically due to decreased number of replica allocation periods. CADRE also exhibits good scalability as more MHs imply better opportunities for replication.

The next section provides a brief introduction to the architecture of CADRE.

2 An overview of the architecture of CADRE

To manage replication efficiently, CADRE considers a hybrid super-peer architecture, in which some of the MHs act as the 'Gateway Nodes' (GNs). The functionality of a GN is analogous to that of a super-peer. In our M-P2P application scenario involving agriculturists, the chief agriculturist in a given region would act as the GN for the MHs in that region. GNs have high processing capacity, high available bandwidth and high energy. We assume that GNs in different regions have the capability of interacting with each other.

GNs facilitate both replication and search. Intuitively, storing replicas arbitrarily at any MH could adversely impact many MHs due to high communication overheads between MHs, unnecessary delays and querying failures. Thus, replication should be performed carefully based on MH characteristics (e.g., load, energy) as well as network topology, thereby implying that some *regional knowledge* becomes a necessity. As we shall see later, GNs have such regional knowledge due to MHs periodically sending the necessary information to GNs, hence GNs can better manage replication. GNs can also collaborate for replication across different regions. In contrast, for an architecture without any GN, each MH would have to broadcast its status to all other MHs to make each other aware of the regional status, thereby creating an undesirable broadcast storm during replica allocation. Our architecture avoids such broadcast storm due to the presence of GNs.

Incidentally, our architecture does *not* have any single 'root' GN for supervising the working of all the GNs. Neighbouring GNs periodically exchange their regional information with each other. Hence, in case of GN failures, neighbouring GNs can take over the responsibility of the failed GN. Furthermore, our architecture does not require local queries to pass via GN, thereby preserving P2P autonomy. This is possible because every MH periodically sends the list of data items/replicas hosted at itself to its GN, and GN broadcasts this information to all MHs within its region.

CADRE does not assume a priori knowledge of the movement patterns of the mobile peers because in case of our application scenarios, mobile peers move randomly i.e., they do not follow a specific pre-defined movement pattern. For example, the agriculturists and the archaeologists would move randomly within their respective regions, while taking snapshots for collecting images of their items of interest for purposes of scientific investigation. However, we do assume that the GNs have limited mobility i.e., they only move within a specific radius. For example, the chief agriculturist or the chief archaeologist of a given region would generally stay within that region.

The remainder of this paper is organized as follows. Section 3 discusses existing works, while Section 4 presents the key components of CADRE. Section 5 discusses the energy-aware query model of CADRE. Section 6 details the dynamic replication scheme of CADRE, while Section 7 presents the algorithm for replication in CADRE. Section 8 reports our performance evaluation. Finally, we conclude in Section 9 with directions for future work.

3 Related Work

This section provides an overview of existing works.

Replication schemes for traditional distributed systems: Replication strategies for distributed environments have been discussed in [22, 21, 28]. In [22], a suite of replication protocols for maintaining data consistency and transactional semantics of centralized systems have been proposed. The protocols in [21] exploit the rich semantics of group communication primitives and the relaxed isolation guarantees provided by most databases. The proposal in [28] discusses replication in distributed environments, where connectivity is partial, weak, and variant as in mobile

information systems. However, these approaches do not consider peer mobility issues.

Schemes for improving data availability in static P2P networks: Schemes for improving data availability in static P2P networks have been discussed in [27, 8, 1]. A scalable P2P framework for distributed data management applications and query routing has been presented in [27]. An update strategy, based on a hybrid push/pull Rumor spreading algorithm, for truly decentralized and self-organizing systems (e.g., pure P2P systems) has been examined in [8], the aim being to provide probabilistic guarantees as opposed to strict consistency. Replication strategies for designing highly available storage systems on highly unavailable P2P hosts are discussed in [1].

Incentive schemes have also been discussed for improving data availability in static P2P networks, the aim being to combat free-riding [12, 14, 19, 24]. These schemes involve formal game-theoretic models for incentive-based P2P file-sharing systems [12], utility functions to capture peer contributions [14], EigenTrust scores to capture participation criteria [19] and asymmetric incentives based on disparities between upload and download bandwidths [24].

Notably, these static P2P approaches are too static to be deployed in M-P2P networks since they assume peers' availability and fixed topology. As a single instance, pre-defined data access structures (e.g., distributed hash tables [33]) used in static P2P networks cannot effectively handle peer mobility and frequent network partitioning, which are characteristic of mobile environments.

Schemes for improving data availability in Mobile ad hoc networks (MANETs): The proposals in [16, 15] discuss replication in MANETs. **E-DCG+** [16] creates groups of MPs that are biconnected components in a MANET, and shares replicas in larger groups of MPs to provide high stability. An RWR (read-write ratio) value in the group of each data item is calculated as a summation of RWR of those data items at each MP in that group. Each replica is allocated at an MP, whose RWR value to the item is the highest among MPs that have free memory space to create it. The work in [15] aims at classifying different replica consistency levels in a MANET based on application requirements, and proposes protocols to realize them. Consistency maintenance is performed via quorums and it is based on local conditions such as location and time.

Incidentally, P2P replication suitable for mobile environments has been incorporated in sys-

tems such as ROAM [29], Clique [30] and Rumor [13]. The work in [11] also discusses replication issues in MANETs. Notably, the proposals in [16, 15, 29, 30, 13] do not consider M-P2P architecture, collaborative replica deallocation, fairness in replication and energy issues.

Incentive schemes for improving data availability in MANETs by combating free-riding have also been discussed [3, 4, 6, 7, 32]. The main purpose of these schemes is to stimulate node cooperation by providing incentives to nodes for relaying messages. However, these schemes do not consider M-P2P architecture and replication issues.

Schemes for improving data availability in M-P2P networks: The proposals in [36, 35] discuss incentive schemes for improving M-P2P data availability by combating free-riding. The work in [36] provides incentives to MPs for participation in the dissemination of reports about resources in M-P2P networks. Each disseminated report contains information concerning a spatio-temporal resource e.g., availability of a parking slot at a given time and location.

The work in [35] considers opportunistic resource information dissemination in transportation application scenarios. An MP transmits its resources to the MPs that it encounters, and obtains resources from them in exchange. The works in [36, 35] primarily address data dissemination with the aim of reaching as many peers as possible i.e., they focus on how every peer can get the data. In contrast, our work considers on-demand services (query-based approach) i.e., the query-issuing peer obtains only its requested data items. Replication issues are not considered in [36, 35].

Economic schemes for resource allocation: The works in [25, 37, 38] discuss economic schemes for resource allocation in wireless ad hoc networks. However, they do not consider replication. Economic schemes for resource allocation in distributed systems have also been proposed [9, 10, 23]. However, they do not address M-P2P issues such as node mobility, frequent network partitioning and mobile resource constraints.

4 Key components of CADRE

This section discusses the key components of CADRE.

User-specified size of the query result image

When a user issues a query for an image, he knows his available memory space status. Hence, he knows the *maximum* amount of memory space, which he can expend for storing the query result image. Thus, when querying, users specify the *maximum size*, designated as $maxsize$, for the query result image and CADRE answers queries while considering this $maxsize$ constraint. This is especially important for M-P2P networks due to the generally limited memory space at individual MHs and the significant differences in available memory space across the MHs. Since image size increases with increasingly finer granularity, a user specifying larger value of $maxsize$ would obtain finer image granularity (i.e., better image quality). Notably, this is in contrast with static P2P systems, where memory space constraints of the query issuing peer are not considered when answering queries due to static peers generally having significant available memory space.

Suppose MH M_I issues a query Q for an image img, and MH M_S serves the query request. M_S could either be the owner of img or it could store a replica of img. Let the size of img at M_S be $size_{img}$. The following cases arise:

1. Query Result $maxsize < size_{img}$
2. Query Result $maxsize \geq size_{img}$

In Case 1 above, img should be *compressed* to satisfy the $maxsize$ query constraint. Such compression should be performed by M_S (and not M_I) due to two reasons. First, it ensures smaller-sized images being transmitted across the network, thereby optimizing bandwidth consumption. Second, a one-time compression of img by M_S is likely to enable M_S to serve multiple user requests, which optimizes energy consumption. Furthermore, performing the image compression at M_I would require every query-issuing MH to compress img individually, which would increase individual MH energy consumption significantly. Notably, image compression algorithms [5, 17] can be used in conjunction with CADRE. For Case 2, image decompression is not necessary since users specifying larger $maxsize$ values can be directed to either the original owner of img or any MH that stores a relatively larger-sized replica of img.

Given that different users can specify different $maxsize$ values for their queries on img, M_S needs to determine the size of the replica (of img) that it should store at itself to reduce its image compression-related energy consumption. For this purpose, M_S keeps track of queries issued to itself by maintaining a list $RepSize$ of the form (img_{id}, MH_{id}, $maxsize$), where img_{id} is the unique identifier of the queried image img, MH_{id} is the identifier of the MH that issued the query and $maxsize$ is the maximum query result size specified by MH_{id}. If an MH M_I accesses img multiple times, the value of $maxsize$ for the most *recent* access is used to populate $RepSize$ since recent $maxsize$ value specified by M_I better reflects M_I's current memory space status. Thus, given img, M_S determines img's replica size by providing equal weight to accesses made by each query-issuing MH since it considers one entry of $maxsize$ for each of these MHs, thereby ensuring *fairness* across multiple user requests for deciding the replica size.

The owner of an image stores the original image. Given the original image size S_o, we consider n different ranges of granularity for replica size based on the extent of image compression relative to S_o. Results of our preliminary performance study revealed that n = 4 is a reasonable value for our application scenarios. Hence, we consider the following four ranges of granularity: *low*, *medium*, *high*, *original*. Let the replica size be S_r. For *low*, $(0.25 \times S_o) \leq S_r < (0.5 \times S_o)$. In case of *medium*, $(0.5 \times S_o) \leq S_r < (0.75 \times S_o)$. Similarly, for *high* $(0.75 \times S_o) \leq S_r < S_o$. Finally, for *original*, $S_r = S_o$. Thus, when different MHs issue queries to MH M_S with different $maxsize$ values, M_S maps each query to any one of these four mutually exclusive ranges and keeps a count of the number of queries for each range. M_S determines the replica size to be in the range that corresponds to the maximum number of queries. Finally, M_S decides the exact size of the replica by averaging the $maxsize$ values of the queries within the selected range.

Fairness in replication

To ensure fairness in replication, each MH M assigns a score σ to each data item d. σ essentially quantifies the *importance* of d to the network as a whole. Hence, σ should increase as d serves more MHs. Given d, M computes σ as follows. First, M sorts the MHs, which recently requested

d, in *descending* order of their access frequencies for d i.e., the first MH in this order made the maximum number of accesses to d. Given this order, M computes σ of d as follows:

$$\sigma \ = \ w \ \times \ (\ \sum_{i=1}^{N_{MH}} n_i \ ) \ \times \ \rho \ \times \ \mu \tag{1}$$

where n_i is the number of accesses made to d by the i^{th} MH in the order specified. Here, the weight coefficient w equals (N/N_{MH}), where N is the number of different MHs which queried the data item d e.g., N = 10 means that 10 different MHs queried d. N_{MH} is the total number of MHs in the network. Thus, σ increases with increase in the number of MHs served by d. Thus, given two data items with equal access frequencies, the score of the data item that serves a larger number of MHs would be higher. This is in contrast with existing works [16], which do not consider the origin of queries. In essence, the weight coefficient w ensures fairness in serving multiple MHs.

In Equation 1, ρ is the spatial density of the region from where the query originated. We consider ρ since we want to replicate a data item to spatially denser neighbourhoods, in order to serve more MHs. ρ = ($Num_{MH}/Area$), where Num_{MH} is the number of MHs in the region from which the query was issued and $Area$ is the area of the region. In our agricultural application, Num_{MH} would be the number of agriculturists in a given region of the agricultural area, while $Area$ would be the area of that region. Notably, GN G knows Num_{MH} since every MH entering G's region needs to register with G. $Area$ is pre-defined w.r.t. the application and G knows the value of $Area$ e.g., for the agricultural application, it could be the area of one region of the agricultural area. Each GN periodically computes its ρ and piggybacks this information in its periodic broadcast to all the MHs in its region. Hence, when an MH issues a query, it puts the value of ρ in the header of its query.

μ is a weight factor for normalizing the data score w.r.t. image size. In M-P2P networks, users often issue queries for small-sized data items due to limited memory space in their mobile devices, hence more replicas for small-sized data items are likely to be allocated. Consequently, replicas may seldom be allocated for large-sized images and this would be unfair to users who

issue queries for these images. μ ensures that larger-sized images would have a fair opportunity of being replicated. We consider three different ranges of image sizes, namely *small*, *medium* and *big*, for which we assign the values of μ to be 0.25, 0.5 and 0.75 respectively. These size ranges are application-dependent.

The score σ_G of a data item d (or replica) w.r.t. a given GN G is the sum of the scores of d at each MH within G's region. Hence, σ_G equals $(\sum_{i=1}^{\eta} \sigma_i)$, where η is the number of MHs in G's region, and σ_i is d's score at the i^{th} MH.

Prevention of thrashing conditions

To address prevention of thrashing, each MH keeps track of the number of deallocations of each replica at itself over a period of time. We define a metric designated as the Flip-Flop Ratio (FFR). The FFR of a replica r at a given MH M is computed as follows:

$$FFR = (N_{dealloc} \div T_{dealloc}) \times (size_r \div T_{size}) \tag{2}$$

where $N_{dealloc}$ is the number of times that r has been deallocated at M, and $T_{dealloc}$ is the total number of deallocations of all the replicas at M during recent time period. $size_r$ is the size of the replica, while T_{size} is the sum of the sizes of all the replicas at M. Thus, the value of FFR is always between 0 and 1. We normalize FFR w.r.t. replica size to minimize the probability of thrashing of large data items since the effect of thrashing is more pronounced for such items due to bandwidth and energy considerations. Periodically, every MH computes the FFR for each replica r stored at itself. As the FFR value of r increases, the probability of thrashing of r also increases. Hence, r should not be deallocated if its FFR value exceeds a certain threshold, which we shall designate as ω. We compute ω as follows:

$$\omega = \left(\sum_{i=1}^{N_{Rep}} FFR_i \right) \div N_{Rep} \tag{3}$$

where N_{Rep} is the total number of replicas at all the MHs in the entire M-P2P network and FFR_i is the value of FFR for the i^{th} replica. Thus, ω is the average value of FFR across all the replicas in the network.

5 Energy-aware Query Model of CADRE

This section discusses the energy-aware query model of CADRE.

User queries Q are of the form $\{Q_{id}$, (k_1,k_2,...,k_n), τ_{max}, ttl, $maxsize\}$, where Q_{id} is the unique identifier of a query, and k_i are user-specified keywords e.g., if an M-P2P user requests an image of 'asparagus officinalis'[2], k_1= 'asparagus' and k_2 = 'officinalis'. Here, τ_{max} is the *deadline* for query response from the user's perspective. Notably, M-P2P ephemerality necessitates query deadlines. ttl is the maximum time-to-live of a query, and it is application-dependent, while $maxsize$ is the user-specified maximum size of the query result.

Recall that our M-P2P applications support both *local* and *remote* querying. To support efficient querying, each MH *periodically* sends its list of data items and replicas to its corresponding GN. Thus, GN is able to periodically broadcast the list of available items within its region to the MHs, thereby enabling a query issuing MH M to distinguish whether its query is *local* or *global*. When an MH enters a region R, it registers with the GN G in R, and G provides the MH with the list of data items currently available in R. (We assume that an MH will have to register with one GN.)

Processing of local queries: Query-issuing MP M_I uses a broadcast mechanism i.e., it broadcasts the query Q for a data item d to its neighbouring MHs, which in turn, forward it to their neighbouring MHs and so on. If an MP M_S receiving Q contains d (or its replica), it puts its MP_{id} (unique identifier of an MP) and its remaining energy into the query message and informs M_I about d's size. Otherwise, it just puts its MP_{id} and its remaining energy into the query message, increments the number of hops in the query message, and forwards Q to its one-hop neighbours.

M_S returns the size of d to M_I only if it estimates that it can satisfy the τ_{max} and ttl constraints. M_S estimates the time to answer Q based on d's size, the bandwidth that it can make available for d

[2] 'Asparagus officinalis' is the scientific name for asparagus.

and its knowledge of the previous history of the network by examining queries which pass through itself as well as by periodically exchanging messages with its neighbours. Thus, local queries need not pass through GN, thereby preserving P2P autonomy.

When M_I receives messages from possibly multiple MPs, which host d or its replica, it lists the query paths associated with each of these messages. (Recall that each MP appends its MP_{id} to the query messages.) Let L denote the list of possible query paths from M_I to the queried data item. From this list L of query paths, M_I selects the query path as follows.

First, all paths, for which the querying-serving MH M_S has energy below $Energy_{Th}$, are deleted from L. Here, $Energy_{Th}$ is an energy threshold, at which the energy of an MH is low such that it will die out if it answers the query. Notably, $Energy_{Th}$ is application-dependent. Second, all paths, which contain at least one relay MH with (remaining) energy below $Energy_{Th}$, are deleted from L. (In case no remaining query path exists in L, the query is not answered because we give higher priority to preserving network connectivity over answering a single query.)

Third, for the remaining query paths, we define the **total energy consumed** in a given query path as (($\sum_{i=1}^{r} Energy_i$) + $Energy_{M_S}$), where r is the number of relay MPs in that path, $Energy_i$ is the remaining energy of the i^{th} relay MH, and $Energy_{M_S}$ is the energy of the query-serving MH M_S. Thus, M_I selects the query path for which total energy consumption is minimized.

Processing of remote queries: The query-issuing MH M_I sends the remote query Q to its corresponding GN, which forwards Q to its neighbouring GNs. (Recall that GN periodically broadcasts the list of data items to MHs in its region, hence M_I knows when it is issuing a *remote* query.) If any of the neighbouring GNs contains the item d queried by Q, they ask the MH, which hosts d, to send d to M_I. Otherwise, the neighbouring GN will forward it to its neighbouring GN and so on. Each GN adds its unique identifier to the query header to ensure that it does not process the same query more than once. Contrast this with an architecture without any GNs, where every remote query would require a broadcast, thereby resulting in a broadcast storm. Thus, our architecture avoids broadcast storm in case of *remote queries* due to GNs collaborating with each other.

Observe the *hybrid* nature of our architecture in that it uses a P2P paradigm within a region, while deploying a distributed superpeer-based model across different regions. Notably, as in the case for local queries, remote queries are also processed in an energy-aware manner i.e., the query path with the least energy consumption is selected. This is possible without any intervention from the respective GNs because the query message contains the remaining energy of each MH in the query path, thereby allowing M_I to select the query path with the lowest energy consumption.

After selecting the query path, M_I sends a message to the target host MP M_S of d in the selected query path to indicate its interest to download d from M_S. Then M_S transfers d to M_I through the relay MPs in the selected query path.

6 CADRE: A dynamic replication scheme for M-P2P networks

This section discusses the details of the CADRE replication scheme for M-P2P networks. In CADRE, each data item is owned by only *one* MH. Available memory space at each MH, bandwidth and data item sizes may vary. We define the **load** L_i of an MH M_i as follows:

$$L_i \; = \; J_{i,t_j} \; \div \; (\; B_{M_i} \; \div \; B_{min} \;) \tag{4}$$

where J_{i,t_j} represents the job queue length of M_i at time t_j, and B_{M_i} is the available bandwidth of M_i. A straightforward way of determining B_{min} is to select a low bandwidth as B_{min} e.g., we have used 56 Kbps as the value of B_{min}. Observe how our definition of load addresses bandwidth heterogeneity among the MHs.

Table 1 summarizes our notations. Let us now examine how MH M_S, which serves the query request, maintains access statistics of queries issued to itself for facilitating replication.

Maintenance of access statistics at each MH

M_S distinguishes between accesses made to its own data items from within the region of its corresponding GN (i.e., *internal accesses*) and accesses to its own data items from MHs that are moving within the region of other GNs (i.e., *external accesses*). Hence, in Table 1, D_{int} and D_{ext} are lists

Parameter	Significance
img	A given queried image
img_{id}	Identifier of img
M_I	Identifier of the query issuing MH
M_S	Identifier of the MH serving the query request
GN_I	Identifier of the GN in whose region M_I is currently moving
$maxsize$	User-specified maximum query result size
t	Time of query issue
D_{int}	List summarizing internal accesses to M_S's own data items at M_S
D_{ext}	List summarizing external access to M_S's own data items at M_S
R_{int}	List summarizing internal accesses to replicas at M_S
R_{ext}	List summarizing external access to replicas at M_S

Table 1: Summary of Notations

in which M_S summarizes the internal accesses and external accesses respectively to its own data items. D_{int} guides M_S in selecting its own data items that should be replicated within the region of its corresponding GN, while D_{ext} facilitates M_S in determining its own data items that should be replicated at regions covered by other GNs. Using Table 1, each entry in D_{int} is of the form (img_{id}, M_I), while entries in D_{ext} are of the form (img_{id}, GN_I, M_I). M_S uses the entry of GN_I in D_{ext} to decide the GN, within whose region the given data item should be replicated.

M_S also differentiates between its own data items and the replicas that are stored at itself. Thus, in Table 1, R_{int} and R_{ext} are lists in which M_S summarizes the internal accesses and external accesses respectively to the replicas stored at itself. Using Table 1, each entry in R_{int} is of the form (img_{id}, M_I), while entries in R_{ext} are of the form (img_{id}, GN_I, M_I). Notably, here img_{id} refers to the identifier of the *replica* stored at M_S, while for the lists D_{int} and D_{ext}, img_{id} represented the identifier of M_S's own data item. Besides this difference, the data structures of R_{int} and R_{ext} are essentially similar to that of D_{int} and D_{ext} respectively. R_{int} and R_{ext} guide M_S in computing replica scores w.r.t. different GNs, thereby facilitating M_S in deallocating replicas that have low scores w.r.t. a particular GN. The lists D_{int}, D_{ext}, R_{int} and R_{ext} are *periodically* refreshed to reflect *recent* access statistics. This is performed by periodically deleting all the existing entries from these lists and then re-populating them with fresh information from the recent queries. Such refreshing

is especially important due to the dynamic changes in access patterns in M-P2P networks.

Selection of candidate data items for replication

Using its D_{int} and R_{int} respectively, each MH computes the score of each of its items (i.e., its own data items and replicas stored at itself), which were accessed by MHs from within the region of its corresponding GN G. Since D_{int} and R_{int} summarize the *internal accesses*, these scores are w.r.t. G. Similarly, from its D_{ext} and R_{ext} respectively, each MH calculates the score of each of its items, which were accessed by MHs that are outside the region of G. In this case, MHs from the respective regions corresponding to multiple GNs may have accessed a particular item, hence the scores of data items and replicas are computed w.r.t. each GN *separately*. *Periodically*, each MH sends all these scores to G. Upon receiving these scores from all the MHs in its region, G sums up the score of each item (w.r.t. each GN) from each MH within its region, thereby computing the total score of each item w.r.t. each GN.

Intuitively, internally accessed items should be replicated at MHs within G's region, while the externally accessed items need to be replicated at MHs in the regions of other GNs. Hence, when selecting candidate items for replica allocation, G distinguishes between internally accessed and externally accessed data items. For the internally accessed items, G sorts these items in descending order of their scores. G considers those items, whose scores exceed the average score ψ, as candidates for replication. ψ equals ($(1/N_d) \sum_{j=1}^{N_d} \sigma_j$), where N_d is the total number of items and σ_j is the score of the j^{th} item. Observe how G prefers items with relatively higher scores for replica allocation due to the higher importance of these items.

For the externally accessed items, G computes the score of each data item d w.r.t. every (external) GN from whose region at least one access was made for d. Then G creates a list $L_{Suggest}$ of these items, each entry of which is of the form (img_{id}, σ, GN_I), where img_{id} is the identifier of the item, and σ is the score of the item w.r.t. GN_I, which is the identifier of a given external GN. Then G sorts the items in $L_{Suggest}$ in descending order of σ. G considers items (of $L_{Suggest}$), whose scores exceed the threshold λ, as candidates for replication. (The remaining items are deleted from

$L_{Suggest}$.) λ equals ($(1/N_d) \sum_{j=1}^{N_d} \sigma_j$), where N_d is the total number of items accessed by external GNs, and σ_j is the score of the j^{th} data item w.r.t a given external GN.

Observe the similarities in determining the candidate data items for replication for internally and externally accessed data items, the difference being that the case for externally accessed data items is more complicated due to the computation of data scores w.r.t. multiple GNs. Furthermore, G does not participate in allocating replicas for the selected candidate items in $L_{Suggest}$. Instead, for each candidate item, G just sends a message to the *relevant* external GN, which will perform the actual replica allocation at some MH within its region. Given img_{id}, the relevant external GN is the corresponding GN_I in $L_{Suggest}$. Note that, just as G suggests external GNs to replicate items, the external GNs also suggest G to replicate items that have been accessed at these external GNs by the MHs of G's region. We shall henceforth refer to the list of items, for which G needs to allocate replicas, as I_{Rep}. Thus, I_{Rep} comprises two types of items: (a) items that are stored at the MHs within its own region R (i.e., *internal items*) (b) items which are stored at MHs outside R (i.e., *external items*). External items are recommended to G by the other (external) GNs.

Selection of a destination MH for storing the replica

Given a data item d to be replicated, the GN determines the destination MH within its region for hosting the replica of d based on the remaining energy of the MHs, the time required for downloading the data item to the destination MH and the MH's load.

First, GN G determines the MH M_{max}, which made the maximum number of accesses to d. This facilitates bringing d nearer to the origin of most of the requests for d. Then G creates a list L_{Dest}, which consists of M_{max} and the n-hop neighbours of M_{max}. Notably, we also consider the n-hop neighbours of M_{max} because M_{max} may not have adequate available memory space and/or remaining energy to host d's replica. Our preliminary experiments indicated that n = 3 is appropriate for our application scenarios, hence CADRE considers MHs, which are upto 3 hops of M_{max}, as candidates for hosting d's replica.

From L_{Dest}, G first deletes all MHs, whose available memory space is less than the size of d's

replica, because such MHs obviously cannot host d. Then, from L_{Dest}, G deletes all MHs, whose energy is below $Energy_{avg}$, which is the average of the remaining energy of all the MHs in L_{Dest}. Thus, CADRE aims at replicating d only at MHs with relatively higher energy because high-energy MHs are more likely to be able to provide better service to the network by possibly answering more queries on d's replica for longer duration of time. This also facilitates improved data availability due to the preservation of network connectivity, which reduces network partitioning.

Then from L_{Dest}, G deletes all MHs, whose download time for replica allocation exceeds $Download_{Avg}$. Here, download time refers to the time required for transferring the replica from the replica's source MH to the destination MH. $Download_{Avg}$ is the average of the download times for the (remaining) MHs in L_{Dest}. This facilitates in minimizing the total time required for completing the replication procedures. G estimates the time required for the replica to be transmitted from the replica's source MH to the destination MH by using its knowledge about the past statistics of the network as well as the size of the replica.

Finally, as we shall see shortly in Section 7, CADRE selects the destination MH for storing the replica by using MH load as a criteria on the remaining MHs in L_{Dest}. Observe that even though CADRE aims at minimizing replication overheads by trying to allocate the replica at an MH with relatively low download time for replica allocation, this does not necessarily imply that replica allocation occurs only at the one-hop neighbours of M_{max}. As a single instance, the one-hop neighbours of M_{max} may all be having low remaining energy. In essence, replica allocation in CADRE can occur at any MH within three hops of M_{max}.

7 Algorithms for replication in CADRE

In CADRE, each GN executes replica allocation and deallocation within the region that it covers. In addition to the scores of items, each MH also sends its load status, energy status, available memory space status and the FFR values of the replicas stored at itself to the corresponding GN in its region. Figure 1 depicts the CADRE replication algorithm, which is executed by a given GN G for allocating replicas at MHs within its own region. The list I_{Rep} in Figure 1 comprises items

that are candidates for replica allocation by G. (We have discussed the composition of the list I_{Rep} earlier in Section 6.) Line 1 of Figure 1 indicates that CADRE allocates replicas starting from the data item with the highest score, thus preferring data items with higher scores since these data items are important to the network as a whole.

Lines 3-7 indicate how CADRE determines the candidate list L_{Dest} of the potential destination MHs for storing the replica, as described earlier in Section 6. Observe how CADRE considers the available memory space and remaining energy of the MHs, as well as the download times required for the replica to be transmitted from the replica's source MH to the destination MH. Notably, the values of the thresholds such as $Energy_{Avg}$ and $Download_{Avg}$ are computed as discussed earlier in Section 6.

CADRE avoids replica allocation at overloaded MHs primarily because such MHs would not be able to provide good service due to their large job queues, which would force queries to incur long waiting times and consequently, higher response times. As Line 11 indicates, CADRE allocates replicas of data items with relatively high scores to underloaded MHs. Notably, the score of an item d may not have a direct correlation with the load imposed by d on the MH M, if d were to be replicated at M. This is primarily because the score depends not only on the total access frequency of d, but also on the *number* of MHs that accessed d. However, we believe that assigning items with high scores to relatively underloaded MHs is reasonable because it facilitates better query response times for items, which are important to a larger number of MHs.

Lines 12-33 depict how the score of the item d to be replicated at the MH M is compared with the scores of the existing replicas. The algorithm keeps adding the existing replicas to the list $L_{dealloc}$ as long as the sum of the scores of these replicas is less than the score of d. If the total size of the replicas in $L_{dealloc}$ exceeds the size of d, the replicas in $L_{dealloc}$ will be deallocated in order to replicate d at M. Otherwise, the replicas in $L_{dealloc}$ will not be deallocated. In essence, we are first *simulating* the eviction of replicas, and if d can be replicated at M by removing a set of existing replicas, whose combined score is less than that of d, we deallocate those existing replicas in favour of d. This is justifiable because d is more important to the network than these deallocated

Algorithm *CADRE*

I_{Rep}: List of data items that are candidates for replica allocation

(1) Sort data items in I_{Rep} in descending order of σ
(2) for each data item d in I_{Rep}
(3) Find the MH M_{max} which has made maximum number of accesses to d
(4) Add M_{max} and its n-hop neighbours to a set L_{Dest}
(5) From L_{Dest}, delete MHs with low available memory space
(6) From L_{Dest}, delete MHs with remaining energy below $Energy_{Avg}$
(7) From L_{Dest}, delete MHs with replica download time above $Download_{Avg}$

(8) if L_{Dest} is an empty list
(9) **break**
(10) else
(11) Sort the MHs in L_{Dest} in ascending order of load

(12) for each MH M in L_{Dest}
(13) Create a list L_R of the replicas stored at M
(14) From L_R, delete replicas with FFR above threshold ω

(15) if L_R is an empty list
(16) **break**
(17) else
(18) Sort L_R in ascending order of σ
(19) $L_{dealloc}$ = empty /* List of deallocations */
(20) σ_{cnt} = 0, $size_{cnt}$ = 0

(21) for each replica r in L_R
(22) /* $size_d$ is d's size and $size_r$ is r's size */
(23) /* σ_d is d's score and σ_r is r's score */
(24) $\sigma_{cnt} = \sigma_{cnt} + \sigma_r$
(25) $size_{cnt} = size_{cnt} + size_r$
(26) if $\sigma_d < \sigma_{cnt}$
(27) **break**
(28) else
(29) if $size_d < size_{cnt}$
(30) Deallocate all entries in $L_{dealloc}$ from M
(31) Allocate d at M
(32) else
(33) Add r to $L_{dealloc}$
end

Figure 1: CADRE replication algorithm

replicas based on the scores. However, when the total score of the replicas in $L_{dealloc}$ exceeds d's score, we do not deallocate the replicas because the combined importance of these replicas is higher than that of d.

Line 14 indicates that we do not deallocate replicas whose FFR values exceed the pre-defined threshold ω, the primary reason being to avoid thrashing conditions. High FFR value of a replica means that it has been allocated and deallocated multiple times. Thus, a replica with FFR value above ω is likely to be accessed again, which might require it to be allocated once again, thereby increasing the probability of thrashing. Incidentally, if there is still some available memory space at some MHs after the CADRE algorithm has been executed for all the candidate data items, the algorithm is executed multiple times until none of the MHs have adequate memory space for storing replicas.

8 Performance Evaluation

This section reports our performance evaluation using simulation. For the simulation, we have used our own implementation.

The parameter values used in our performance evaluation have been selected carefully based on other works such as [16], where a similar environment has been considered for performance evaluation. Additionally, some of the parameter values have been selected according to our environment and our application scenarios. As a single instance, our application scenarios typically concern situations, where the number of MHs in a given region is not large. For example, in the application scenario concerning agriculturists in an agricultural area, the number of agriculturists in the network (within a given region) would typically be less than 50. Hence, in our experiments, we have set the number of MHs to 50.

The bandwidth range of 28-100 Kbps is in consonance with the available bandwidth ranges, which could be reasonably expected in our application scenarios. We have set the size of a data item in the range of 1-10 MB because our data items are generally in that range e.g., images/video-clips of plants, soil and historical artefacts. In particular, we do not consider big-sized data such

as movies. We have set the relocation period to 200 seconds since it is a reasonable value for our application scenarios. In general, relocation period is tied to replication, and it is essentially application-dependent.

We consider *five* different regions. Each region has 50 MHs and 1 GN. MHs in each region move according to the *Random waypoint model* [2] within the region, the area of the region being 1000 metre $\times$1000 metre. The *Random waypoint model* is appropriate for our application scenarios, which involve random movement of users. GNs move within their respective regions and we assume that they are able to communicate with each other. Each region contains 200 data items that are uniformly distributed among 50 MHs i.e., each MH owns 4 data items. Each query is a request for either a local data item or a remote one. For query routing purposes, we have used the AODV protocol. In all the experiments presented here, 60% of the queries were remote ones, while the other 40% were local queries. We had performed experiments with different percentages of remote and local queries, the results indicating increasing query response times with increasing percentage of remote queries. Since these experiments exhibited similar trends, we do not present the results here due to space constraints.

Periodically, every TP seconds, each GN decides whether to perform replica allocation. Network topology does *not* change significantly during replica allocation since it requires only a few seconds [16]. In all our experiments, 20 queries/second are issued within each region, the number of queries directed to each MH being determined by the Zipf distribution. Communication range of all MHs (except the GNs) is a circle of 100 metre radius. Notably, initial energy of an MH is selected to be between 90000 to 100000 energy units using a random number generator. Table 2 summarizes the performance study parameters, which are the same for each of the five regions.

We ran each experiment 10 times, hence the experiment result values are the average of these runs. The respective confidence intervals for our experiments ranged from 90% to 95%.

Performance metrics are **average response time** (**ART**) of a query, **data availability** (**DA**) and communication **traffic** (**TR**) for replica allocation. ART = $(1/N_Q)\sum_{i=1}^{N_Q}(T_f - T_i)$, where T_i is the time of query issuing, T_f is time of the query result reaching the query-issuing MH, and N_Q is the

Parameter	Default value	Variations
No. of MHs (N_{MH}) in each region	50	10, 20, 30, 40
Zipf factor (ZF)	0.9	0.1, 0.3, 0.5, 0.7
Allocation period TP (10^2 s)	2	1, 3, 4, 5, 6
Queries/second	20	
Bandwidth between MHs	28 Kbps to 100 Kbps	
Probability of MH availability	50% to 85%	
Size of a data item	1 MB to 10 MB	
Available Memory at an MH	10 MB to 20 MB	
Initial energy of an MH	90000 to 100000 energy units	
Speed of an MH	1 metre/s to 10 metres/s	
Size of message headers	220 bytes	

Table 2: Performance Study Parameters

total number of queries. ART includes the download time, and is computed only for the successful queries. Notably, unsuccessful queries are not considered for ART computations.

DA = (N_S/N_Q)*100, N_S being the number of queries that were answered successfully. Each query has a 'time-to-live' (TTL) i.e., queries that are not answered within n hops are dropped. All our experiments use n = 6 since preliminary experiments indicated that it is a reasonable value for our application scenarios. In CADRE, queries can fail due to their deadlines not being satisfied or due to MHs being unavailable or due to network partitioning, or due to exceeding the TTL. We define TR as the total hop-count for replica allocation during the experiment.

In our simulation model, we have also considered the probability of interruption during data transfer as a parameter with the chances of interruption/failure (during data transfer) being 10% to 20%. However, our simulation model does not consider complete disconnections. In particular, the occurrence of complete interruptions during data transfer is in the realm of networking, which is not our primary focus in this paper. However, we do acknowledge the importance of this issue and leave this issue open to further research. Furthermore, the overhead introduced by the GN is negligible in comparison to the traffic for allocation and deallocation of data items. Notably, the overhead introduced by the GN and the replica allocation is part of the communication delay.

Performance comparison w.r.t. existing approaches: Our work cannot be meaningfully

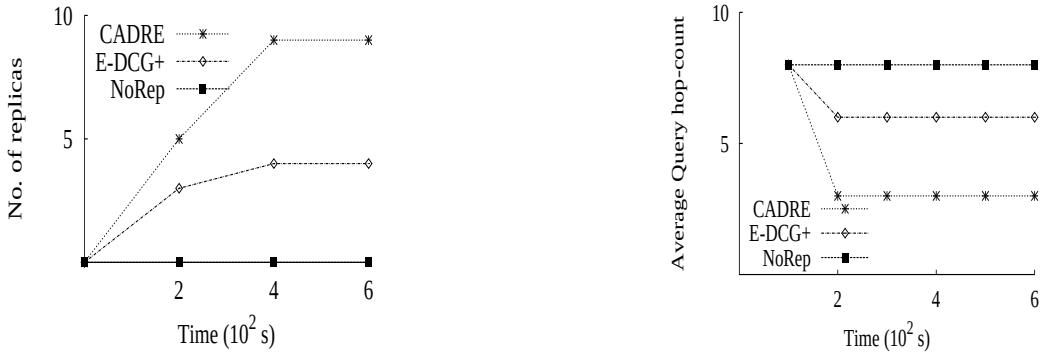
compared with replication schemes for traditional distributed environments [22, 21, 28] and replication schemes for static P2P networks [27, 8, 1] because these schemes are too static to be deployed in M-P2P environments as they assume peers' availability and fixed topology. In a similar vein, incentive schemes for improving data availability in static P2P networks [12, 14, 19, 24] are not directly comparable to our work because they do not consider peer mobility and limited energy resources of the peers, which are characteristic of M-P2P environments. Finally, our work cannot also be directly compared with incentive schemes for improving data availability in M-P2P networks [36, 35] because they consider a data dissemination model, while we consider a query-based model. Moreover, they do not address replication. The "push-based" data dissemination model in [36, 35] will increase traffic significantly as compared to our "pull-based" querying model, thereby resulting in significantly higher energy consumption of MHs, and consequently, making it unsuitable for our M-P2P application scenarios, where the energy of devices is generally limited.

For meaningful performance comparison purposes, we adapt the **E-DCG+** approach [16] to our scenario and compare our work against the E-DCG+ approach since it is the closest to our proposed M-P2P replication approach. E-DCG+ is executed at every reallocation period. Notably, E-DCG+ allocates replicas based on data item access frequencies and it does not consider collaborative replica deallocation, fairness in replication and preservation of MH energy. As a baseline, we also compare CADRE with an approach **NoRep**, which does not perform replica allocation.

Effect of fair replica allocation

We conducted an experiment to observe the number of replicas created by CADRE and E-DCG+ for a single 'hot' data item d over a period of time. This data item was selected randomly from the top 10% hottest data items. Figure 2a depicts the results. The confidence interval for the results of this experiment is 93%.

For both CADRE and E-DCG+, the number of replicas increases over time in response to conditions necessitating replica allocation. However, the number of replicas does not increase indefinitely over time and eventually plateaus after some time due to competition among replicas



(a) No. of replicas (b) Average query hop-count

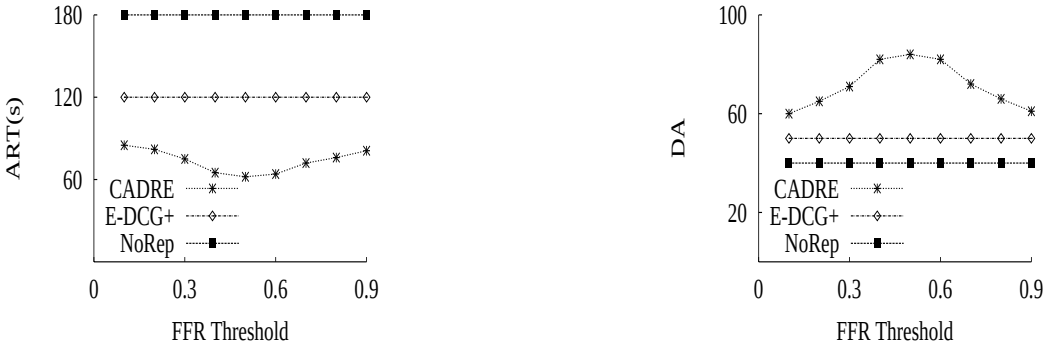
Figure 2: Effect of fair replica allocation

for MH memory space. Observe that CADRE creates more replicas than E-DCG+. This is because CADRE would create a replica for a data item d, which is accessed by a large number of MHs, even if d's total access frequency is low, in which case E-DCG+ would not create any replica. Thus, CADRE creates replicas for more data items than E-DCG+ since CADRE selects candidate items for replication based on scores as opposed to total access frequencies.

Figure 2b indicates the average number of hop-counts required for querying the same data item d during different periods of time. These results were averaged over a total of 1200 queries. Initially, before replica allocation had been performed, all three approaches required comparable number of hops for querying d. After replica allocation has been performed, CADRE requires lower number of hops than E-DCG+ to answer queries on d since CADRE creates more replicas for d, as discussed for Figure 2a. More replicas generally decrease the querying hop-count since it increases the likelihood of queries being answered within lower number of hops and provide multiple paths to locate a queried data item. Furthermore, CADRE's effective preservation of the energy of MHs implies better network connectivity (i.e., less frequent network partitioning), which also reduces the querying hop-counts in case of CADRE. E-DCG+ requires lower number of querying hop-counts than NoRep essentially due to replication.

Effect of thrashing prevention

Recall that CADRE deallocates only those replicas, whose FFR values are less than that of the FFR threshold ω. Figures 3a and 3b depict the effect of variations in ω on the ART and DA of CADRE. The confidence interval for the results of this experiment is 95%.



(a) Average Response Time (b) Data Availability

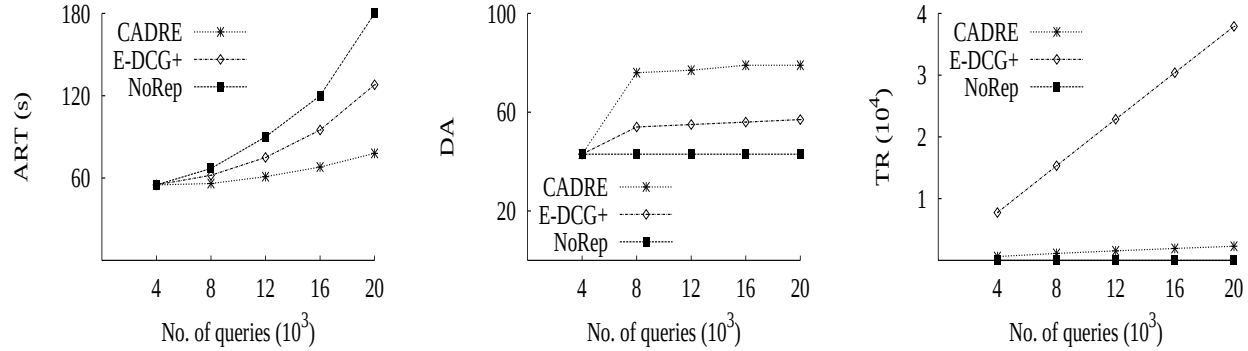
Figure 3: Effect of thrashing prevention

E-DCG+ and NoRep show relatively constant ART and DA as these approaches are independent of ω. For high values of ω, the FFR of more replicas fall below ω, thereby making the occurrence of a large number of deallocations more likely. This is likely to lead to thrashing conditions, and consequently increased ART and decreased DA. However, when the value of ω is low, the FFR of few replicas fall below ω. This makes deallocation too 'conservative' in the sense that replicas, which should have been deallocated to create memory space for 'hot' items, will not be deallocated, Thus, items with high scores cannot be allocated due to lack of space, thereby adversely affecting the performance of CADRE. As the results in Figures 3a and 3b indicate, CADRE performs best at intermediate values of ω i.e., $0.4 \leq \omega \leq 0.6$. By computing ω for this experiment according to our analytical formula in Equation 3, we obtain $\omega \approx 0.54$, which experimentally justifies Equation 3.

Performance of CADRE

We conducted a simulation experiment using default values of the parameters in Table 2. Figure 4 depicts the results. The confidence interval for the results of this experiment is 94%.

Figure 4a indicates that the performance gap between CADRE and E-DCG+ keeps increasing



(a) Average Query Response Time (b) Data Availability (c) Replica Allocation Traffic

Figure 4: Performance of CADRE

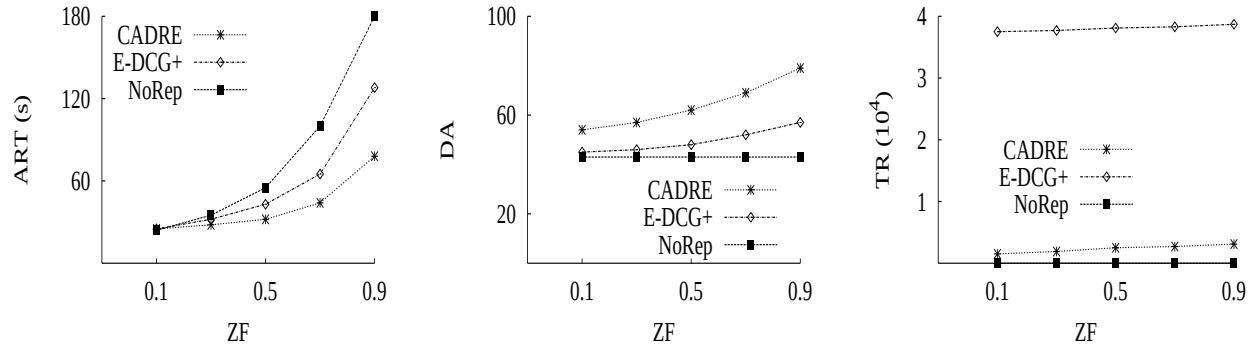
over time due to several reasons. First, in case of CADRE, replica allocation and deallocaton are both performed by the GNs. Since a GN has good regional knowledge concerning load status, available memory space status and energy status of MHs as well as network topology, it is able to better manage replication in its region. In contrast, for E-DCG+, replica allocation is performed in a distributed manner by individual MHs, which lack good regional knowledge. Second, CADRE allocates replicas *only* to MHs with relatively low loads, thereby ensuring relatively short waiting times for queries at the job queues of these MHs, and consequently, reduced query response times. However, since E-DCG+ does not consider MH load, it may allocate replicas to overloaded MHs, thereby resulting in high query response times at these MHs due to their large job queues. Third, CADRE uses the FFR threshold to facilitate the prevention of thrashing conditions, which E-DCG+ does not address. Notably, preventing thrashing can improve performance significantly when large-sized items (e.g., images) are present, as in our application scenarios. Fourth, unlike E-DCG+, CADRE creates replicas for more data items since it considers fairness in replica allocations via data scores, as discussed for the results in Figure 2. Fifth, unlike E-DCG+, CADRE takes MH energy into account, which improves query response times due to better network connectivity.

The results in Figure 4b suggest that CADRE provides higher data availability than E-DCG+ essentially due to the reasons discussed for Figure 4a. Incidentally, during replica allocation, E-DCG+ requires every MH to broadcast its RWR values to every MH, thereby incurring O(N_{MH}^2) messages, while CADRE requires each MH to send only one message to its corresponding GN,

and the GN broadcasts a message to each MH, thus incurring O(N_{MH}) messages, which explains the results in Figure 4c.

Effect of variations in the workload skew

Figure 5 depicts the results when the zipf factor (ZF) is varied. The confidence interval for the results of this experiment is 90%.



(a) Average Query Response Time (b) Data Availability (c) Replica Allocation Traffic

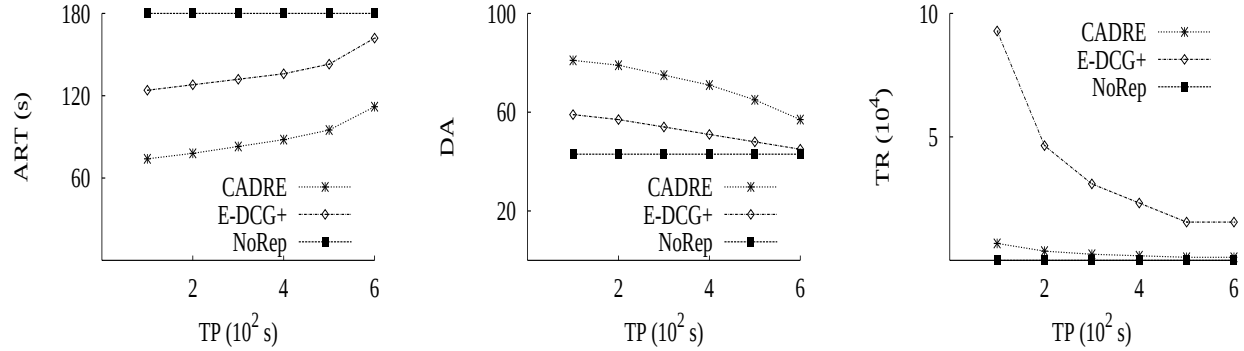
Figure 5: Effect of variations in the workload skew

For high ZF values (i.e., high skew), both CADRE and E-DCG+ perform better than NoRep in terms of ART and DA due to more replica allocations in response to load-imbalance conditions. The performance gap (in terms of ART and DA) between CADRE and E-DCG+ increases with increasingly skewed workloads essentially due to the reasons explained for Figures 2 and 4. In particular, unlike E-DCG+, CADRE uses MH load as a replication criteria, thereby making it more sensitive to load-imbalance conditions. However, the performance of all three approaches is comparable at lowly skewed workloads since such workloads do not necessitate replica allocations. The explanation for Figure 5c is essentially the same as that of Figure 4c.

Effect of variations in the replica allocation period

Recall that every TP seconds, a GN decides whether to allocate replicas. Figure 6 depicts the results of varying TP. The confidence interval for the results of this experiment is 92%.

At lower values of TP, more number of replica allocation periods occur, hence load imbalances



(a) Average Query Response Time (b) Data Availability (c) Replica Allocation Traffic

Figure 6: Effect of variations in the replica allocation period TP

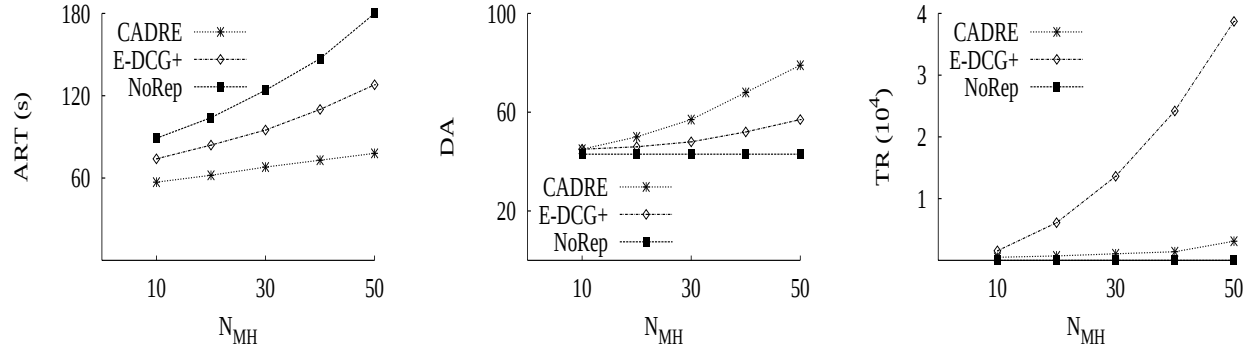
are corrected quickly in response to changing access patterns, thereby improving ART and DA for both CADRE and E-DCG+. As TP increases, load imbalances are corrected less frequently, hence performance degrades for both CADRE and E-DCG+. For NoRep, ART and DA remain relatively constant because they depend only upon probability of MH availability. The explanation for the results in Figure 6c is similar to that of Figure 4c. In particular, replica allocation traffic decreases dramatically with increasing TP due to decreased number of replica allocation periods.

Effect of variations in the number of MHs

To test CADRE's scalability, we varied the number N_{MH} of MHs, keeping the number of queries proportional to N_{MH}. Figure 7 depicts the results. The confidence interval for the results of this experiment is 94%. At high values of N_{MH}, CADRE outperforms E-DCG+ due to the reasons explained for Figures 2 and 4. As N_{MH} decreases, the performance gap decreases due to limited replication opportunities. Replica allocation traffic for E-DCG+ dramatically decreases with decreasing N_{MH} due to reduced broadcast traffic.

9 Conclusion

We have proposed CADRE, which is a dynamic replication scheme for improving the typically low data availability in *dedicated* and *cooperative* M-P2P networks. CADRE collaboratively performs both replica allocation and deallocation in tandem to facilitate effective replication and to avoid



(a) Average Query Response Time (b) Data Availability (c) Replica Allocation Traffic

Figure 7: Effect of variations in the number of MHs

'thrashing' conditions, while addressing fair replica allocation across the MHs. Such collaboration is facilitated by a hybrid super-peer architecture in which some of the mobile hosts act as the 'gateway nodes' (GNs) in a given region. GNs facilitate both search and replication. Furthermore, CADRE considers the replication of images at different resolutions to optimize the usage of the generally limited memory space of the MHs. CADRE also facilitates the optimization of the limited energy resources of MHs during replication.

Our performance evaluation demonstrates that CADRE is indeed effective in improving data availability in M-P2P networks with significant reduction in query response times and low communication traffic during replication as compared to a recent existing scheme as well as a baseline approach, which does not consider any replication. In the near future, we plan to integrate CADRE's replica allocation and deallocation algorithms with incentive models for peer participation to further improve data availability in M-P2P networks.

References

[1] R. Bhagwan, D. Moore, S. Savage, and G. M. Voelker. Replication strategies for highly available peer-to-peer storage. *Proc. Future Directions in Distributed Computing*, 2003.

[2] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocol. *Proc. MOBICOM*, pages 159–164, 1998.

[3] L. Buttyan and J. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. *Technical Report DSC/2001/001, Swiss Federal Institute of Technology,Lausanne*, 2001.

[4] L. Buttyan and J.P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), 2003.

[5] B. Carpentieri, M. Weinberger, and G. Seroussi. Lossless compression of continuous-tone images. *Proc. IEEE*, 2000.

[6] K. Chen and K. Nahrstedt. iPass: an incentive compatible auction scheme to enable packet forwarding service in MANET. *Proc. ICDCS*, 2004.

[7] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modelling incentives for collaboration in mobile ad hoc networks. *Proc. WiOpt*, 2003.

[8] A. Datta, M. Hauswirth, and K. Aberer. Updates in highly unreliable replicated peer-to-peer systems. *Proc. ICDCS*, 2003.

[9] D.F. Ferguson, C. Nikolaou, and Y. Yemini. An economy for managing replicated data in autonomous decentralized systems. *Proc. International Symposium in Autonomous Decentralized Systems*, pages 367–375, 1993.

[10] D.F. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic algorithms for load balancing in distributed computer systems. *Proc. ICDCS*, pages 491–499, 1988.

[11] L.D. Fife and L. Gruenwald. Research issues for data communication in mobile ad-hoc network database systems. *ACM SIGMOD Record*, 32(2):42–47, 2003.

[12] P. Golle, K.L. Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. *Proc. Electronic Commerce*, 2001.

[13] R. Guy, P. Reiher, D. Ratner, M. Gunter, W. Ma, and G. Popek. Rumor: Mobile data access through optimistic peer-to-peer replication. *Proc. ER Workshops*, 1998.

[14] M. Ham and G. Agha. ARA: A robust audit to prevent free-riding in P2P networks. *Proc. P2P*, pages 125–132, 2005.

[15] T. Hara and S.K. Madria. Consistency management among replicas in peer-to-peer mobile ad hoc networks. *Proc. IEEE SRDS*, 2005.

[16] T. Hara and S.K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing*, 2006.

[17] http://compression.ca/act/.

[18] Y. Huang, A. P. Sistla, and O. Wolfson. Data replication for mobile computers. *Proc. ACM SIGMOD*, 1994.

[19] S. Kamvar, M. Schlosser, and H. Garcia-Molina. Incentives for combatting free-riding on P2P networks. *Proc. Euro-Par*, 2003.

[20] Kazaa. http://www.kazaa.com/.

[21] B. Kemme. Implementing database replication based on group communication. *Proc. Future Directions in Distributed Computing*, 2002.

[22] B. Kemme and G. Alonso. A new approach to developing and implementing eager database replication protocols. *ACM TODS*, 25(3), 2000.

[23] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Computers*, 38(5):705–717, 1989.

[24] N. Liebau, V. Darlagiannis, O. Heckmann, and R. Steinmetz. Asymmetric incentives in peer-to-peer systems. *Proc. AMCIS*, 2005.

[25] Jinshan Liu and Valerie Issarny. Service allocation in selfish mobile ad hoc networks using vickrey auction. *Proc. Current Trends in Database Technology - EDBT Workshops revised papers, LNCS 3268*, 2004.

[26] A. Mondal, S.K. Madria, and M. Kitsuregawa. CLEAR: An efficient context and location-based dynamic replication scheme for mobile-P2P networks. *Proc. DEXA*, 2006.

[27] V. Papadimos, D. Maier, and K. Tufte. Distributed query processing and catalogs for peer-to-peer systems. *Proc. CIDR*, 2003.

[28] E. Pitoura. A replication scheme to support weak connectivity in mobile information systems. *Proc. DEXA*, 1996.

[29] D. Ratner, P.L. Reiher, G.J. Popek, and G.H. Kuenning. Replication requirements in mobile environments. *Mobile Networks and Applications*, 6(6), 2001.

[30] B. Richard, D. Nioclais, and D. Chalon. Clique: A transparent, peer-to-peer replicated file system. *Proc. MDM*, 2003.

[31] S. Saroiu, P.K. Gummadi, and S.D. Gribbler. A measurement study of peer-to-peer file sharing systems. *Proc. MMCN*, 2002.

[32] V. Srinivasan, P. Nuggehalli, C.F. Chiasserini, and R. R. Rao. Cooperation in wireless ad hoc networks. *Proc. INFOCOM*, 2003.

[33] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proc. ACM SIGCOMM*, 2001.

[34] O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *ACM TODS*, 22(4):255–314, June 1997.

[35] O. Wolfson, B. Xu, and A.P. Sistla. An economic model for resource exchange in mobile Peer-to-Peer networks. *Proc. SSDBM*, 2004.

[36] B. Xu, O. Wolfson, and N. Rishe. Benefit and pricing of spatio-temporal information in Mobile Peer-to-Peer networks. *Proc. HICSS-39*, 2006.

[37] Yuan Xue, Baochun Li, and Klara Nahrstedt. Channel-relay price pair: Towards arbitrating incentives in wireless ad hoc networks. *Journal of Wireless Communications and Mobile Computing, Special Issue on Ad Hoc Networks, Wiley InterScience*, 2005.

[38] Yuan Xue, Baochun Li, and Klara Nahrstedt. Optimal resource allocation in wireless ad hoc networks: A price-based approach. *IEEE Transactions on Mobile Computing*, 2005.