Full Paper

# QoS-based Dynamic Replication in Mobile Peer-to-Peer 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@umr.edu

**Abstract**

The ever-increasing popularity and proliferation of mobile computing technology strongly motivate applications involving Mobile ad-hoc Peer-to-Peer (M-P2P) networks. Incidentally, an M-P2P network is one in which mobile hosts (MHs) interact directly with each other in a decentralized peer-to-peer (P2P) fashion. Notably, network partitioning may occur *frequently* in M-P2P networks due to user movement and/or users switching 'on' or 'off' their mobile devices, thereby decreasing data availability. We envisage the M-P2P network as a cluster of MHs, which has a cluster head (CH) for facilitating data validation and replica allocation. The main contribution of this work is the proposal of CLEAR (Consistency and Load-based Efficient Allocation of Replicas), which is a dynamic replica allocation scheme for M-P2P networks. For performing effective replica allocation, CLEAR considers a metric NQDC (Number of Queries answered with a Desired Consistency level) and load as criteria, and uses knowledge of users' schedules. Results of our extensive performance evaluation demonstrate that CLEAR is indeed effective in improving data availability in M-P2P networks.

## 1 Introduction

Rapid advances in wireless communication technology coupled with the ever-increasing popularity and proliferation of mobile devices (e.g., laptops, PDAs, mobile phones) provide a strong motivation for applications involving Mobile ad-hoc Peer-to-Peer (M-P2P) networks. Incidentally, an M-P2P network is one in which mobile hosts (MHs) interact directly with each other in a decentralized peer-to-peer (P2P) fashion. Notably, network partitioning may occur *frequently* in M-P2P networks due to user movement and/or users switching 'on' or 'off' their mobile devices, thereby leading to reduced data availability as compared to that of traditional stationary networks[1]. Hence, efficient dynamic data replication becomes a necessity in M-P2P networks for providing mobile users with high accessibility to relatively *fresh* data in *real-time*.

Applications involving zoological data would benefit tremendously from effective replication in M-P2P networks. Zoologists often need to look for habitat information in remote areas (e.g., dense forests). For example, a group of zoologists may wish to look for instances of a specific kind of micro-organism $X$ in a forest to understand $X$'s characteristics. Since technological infrastructure (e.g., base station) does not exist in these remote areas, the zoologists need to communicate among themselves using mobile devices in a P2P fashion. If a particular zoologist finds an instance of $X$, he needs to share some data (e.g., the type of environment where he found $X$'s instance, the number of instances of $X$ found), by means of replication, with his colleagues so that they can look for similar type of environment for locating more instances of $X$. Notably, only a certain *desired level of consistency* (as opposed to absolute consistency) is required for effective data sharing in such applications [18].

Incidentally, data validation is increasingly becoming a major concern in scientific data collection since erroneous data (collected due to human error or other factors) is detrimental to scientific databases, thereby making it necessary for the data to be validated by an authority figure (e.g., the supervisor of the zoolo-

[1]Data availability is less than 20% even in a wired environment [21].

gists). This motivates the need for one of the MHs to act as the *head* of the entire group of MHs in the M-P2P network. Similarly, effective replication in M-P2P networks would also facilitate applications involving disaster recovery. Incidentally, similar to zoologists, workers in disaster recovery scenarios (e.g., earthquakes, tsunamis) also need to share data (e.g., number of injured people, number of fatalities) with each other without any technological framework.

Replication techniques for traditional distributed systems [10] are *not* adequate for M-P2P networks partly because they do *not* address frequent network partitioning due to user movement and partly due to the generally limited resources (i.e., disk space, processing capacity, battery power) of MHs, which pose significant challenges to replication. Moreover, P2P replication services are not 'mobile-ready' [6, 20] as current P2P systems [13, 5] have mostly ignored data transformation, relationships and network characteristics. Understandably, changes in data with location and time create new research problems [3]. Existing replication techniques for mobile environments [22, 17, 11], which assume *stationary networks*, are also *not* adequate for M-P2P networks since they do not consider frequent network partitioning issues. Replication in M-P2P networks requires fundamentally different solutions [1, 14] than in [19, 15, 17] due to free movement of MHs and wireless constraints. Notably, content providers deploy thousands of cache servers at the point of presence of major ISPs (Internet Service Providers) such as AT&T and Sprint to keep the contents close to clients for reducing traffic [12]. However, this would become a performance bottleneck in case of an M-P2P network. Interestingly, the proposals in [7, 8, 9] consider frequent network partitioning w.r.t. replication in mobile ad-hoc networks (MANETs). Unlike the works in [7, 8, 9], our work uses load as a replication criterion, addresses different levels of replica consistency and deals with unequal-sized data items.

We envisage the M-P2P network as a cluster of MHs, which has a cluster head **(CH)** for facilitating data validation (which is absolutely necessary in our application scenarios) and replica allocation. The main contribution of our work is the proposal of **CLEAR** (*Consistency and Load-based Efficient Allocation of Replicas*), which is a dynamic replica allocation scheme for improving data availability in M-P2P networks. The main features of CLEAR are two-fold.

1. It considers a metric NQDC (Number of Queries answered with a certain Desired Consistency level) and load, while performing replica allocation.

2. It uses knowledge of users' schedules for identifying locations from where specific data items are likely to be accessed.

Results of our extensive performance evaluation demonstrate that CLEAR is indeed effective in improving data availability in M-P2P networks. The remainder of this paper is organized as follows. Section 2 discusses existing works, while Section 3 presents the problem context. Section 4 discusses issues concerning dynamic replica allocation in M-P2P networks, while our proposed CLEAR scheme has been presented in Section 5. Section 6 discusses the performance evaluation. Finally, we conclude in Section 7 with directions for future work.

## 2 Related Work

The work in [15] proposes a suite of replication protocols which maintain data consistency and transactional semantics of centralized systems, while providing flexibility and reasonable performance. The protocols in [14] exploit the rich semantics of group communication primitives and the relaxed isolation guarantees provided by most databases. The work in [4] discusses replication issues in MANETs. The proposal in [17] discusses a replication schema for distributed environments where connectivity is partial, weak, and variant as in mobile information systems.

Existing systems in this area include ROAM [19], Clique [20] and Rumor [6], while a scalable P2P framework for distributed data management applications and query routing has been presented in [16]. In [3], the problem of updates in truly decentralized and self-organizing systems, such as pure P2P systems, has been examined. The proposed update strategy is based on a hybrid push/pull Rumor spreading algorithm, the aim being to devise a fully decentralized robust communication scheme which provides probabilistic guarantees as opposed to ensuring strict consistency. The work in [1] investigates replication strategies for designing highly available storage systems on highly unavailable P2P hosts.

The proposals in [7, 8, 9] present three replica allocation methods with periodic and aperiodic updates, which take into account limited memory space in MHs for storing replicas, access frequencies of data items and the network topology, to improve data accessibility in MANET environments. Notably, the E-DCG+ approach [9] is among the most influential replica allocation approaches. By creating groups of MHs that are biconnected components in a network, E-DCG+ shares replicas in larger groups of MHs to provide high stability. In E-DCG+, 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 MH in that group. In the order of the RWR values of the group, replicas of data items are allocated until memory space of all MHs in the group becomes full. Each replica is allocated at an MH whose RWR value to the data item is the highest among MHs that have free memory space to create it. However, the architecture considered in [7, 8, 9] is not suitable for our application scenarios described earlier since it does not consider certain issues such as load sharing and tolerance to weaker consistency.

## 3 Context of the Problem

This section discusses the context of the problem. The problem context concerns an M-P2P environment comprising MHs, each of which has different amounts of memory space. Each MH distinguishes between two kinds of data items stored at itself, namely the data items that it owns and the replicas. Any data item $d_i$ is owned by only *one* MH, which can update $d_i$ autonomously at any time, but other MHs are *not* allowed to update $d_i$. We assume that each MH owns an equal number of data items, whose sizes may vary, hence memory space for storing replicas differs across the MHs. Bandwidth may vary across the MHs.

We envisage the M-P2P network as a cluster, where an MH with the maximum remaining battery power and processing capacity is selected as the *cluster head* (**CH**). Hence, CH is capable of transmitting messages directly via unicast to any given MH in the cluster. Notably, CH knows the list of data items at each MH. Due to the cluster covering a relatively small area and taking our application scenarios (discussed in Section 1) into consideration, the number of MHs in the cluster can be reasonably expected to be relatively small. We assume that CH backs up information using the Internet as an interface to handle failures and that some of the MHs have access to the Internet for backup purposes. In case CH fails or in case of network partitioning, these MHs can connect with the Internet to obtain the information, thereby enabling them to act as CH.

Sending and receiving messages expend the generally poor battery power of MHs, hence reduction of traffic is critical for optimizing power consumption. Each MH *periodically* sends some information for replica allocation (e.g., access statistics information, update logs, load) to CH. Update log of a given MH contains each data item $d_i$ that it updated in the last periodic time interval, the initial and updated values of the updated attribute of $d_i$ and the timestamp of the update. Update logs are sent only *periodically* to CH for optimizing traffic by piggybacking such logs onto messages containing replica allocation information, even though an MH may update its own data items *aperiodically.* For optimizing memory space usage, MHs are allowed to delete replicas whose access frequency falls below a certain threshold.In our proposed model, all queries, updates, replica allocations and query results must pass via CH for validation reasons, as required by our application scenarios. Observe that passing all the above data via CH reduces the number of hops in most cases because on an average, the number of hops between any two MHs through CH will be less than the number of hops between any two MHs without a CH, provided the two MHs under consideration are not neighbours [2, 14].

When CH receives a query $Q$ for data item $d_i$, CH determines the set of MHs which store $d_i$ or $d_i$'s replicas[2], and directs $Q$ to only *one* of these MHs, hence our architecture results in significantly lower querying traffic as compared to that of broadcast. Additionally, when a queried data item $d_i$ is not available, CH can determine quickly that $d_i$ is not available, and CH can drop the query immediately to further optimize querying traffic. Observe the *hybrid* nature of our architecture in that it preserves the autonomy of the MHs, while deploying CH for facilitating data validation and replica allocation. Now let us examine the advantages of our proposed architecture by contrasting with a distributed architecture from the perspectives of both querying as well as replica allocation.

In a distributed architecture, if any given MH does *not* know the list of data items stored at other MHs, querying would have to proceed by means of broadcast, which incurs high traffic. On the other hand, if each MH maintains information concerning the data items (and replicas) that are stored at all other MHs in the entire M-P2P network as well as the approximate schedules of these MHs, querying traffic would reduce. However, maintenance of such information itself would result in high traffic since it would require each MH to *periodically* broadcast a list of its data items and its schedule to all the other MHs, thereby defeating the very purpose of maintaining such information.

For performing optimal replica allocation in a distributed architecture, each MH needs to know certain information related to replica allocation as discussed earlier. In a distributed architecture, every MH would have to broadcast this information to every MH in the entire M-P2P network, thereby resulting in $O(N^2)$ number of messages, where $N$ is the total number of MHs. Contrast this with our architecture which requires only $O(N)$ messages since every MH needs to send this information only to CH.

Load $L_M$ of an MH $M$ is defined as follows:

$$L_M = \left( \sum_{i=1}^{d} (n_{d_i} \div s_{d_i}) \right) \div \eta_i \tag{1}$$

where $d$ represents the total number of data items that are in $M$'s job queue $n_{d_i}$ stands for access frequency of a given data item $d_i$ during recent time intervals and $s_{d_i}$ denotes the size of $d_i$. Observe how our definition of load takes variations in the respective sizes of the data items into consideration. Given that available bandwidth may differ significantly among MHs, we use $\eta_i$ as a parameter for normalizing the load of an MH w.r.t. available bandwidth. We compute $\eta_i$ as $(B_{P_i} \div$

[2]This is possible since CH knows the list of data items at each MH.

$B_{min}$), where $B_{P_i}$ represents the available bandwidth of $M$. A straightforward way of determining $B_{min}$ is to select a low bandwidth as $B_{min}$ e.g., we have used 28 Kbps as the value of $B_{min}$.

## 4 Issues concerning replica allocation in M-P2P networks

This section discusses issues concerning replica allocation in M-P2P networks.

### Determination of future user access patterns

The objective of replica allocation is to replicate the objects of an MH $M$'s interest at MH(s) that would be near to $M$'s future location at the time when $M$ would access the objects. Hence, maintaining some knowledge concerning *which* objects $M$ frequently accesses and *when* and from *where* $M$ is likely to access those objects becomes a necessity for performing effective replication in M-P2P networks. We propose that CH should maintain this information. Interestingly, in practice, given that the owner of each MH has some *schedule* in mind, these owners send their respective schedules to CH. Notably, even if an MH is not able to adhere strictly to its schedule, such knowledge would still possibly be useful for determining the general direction of motion of the MH.

### Replica consistency

The desired level of replica consistency is essentially application-dependent and primarily depends on how much replica consistency is required by the users of a given application as opposed to *absolute replica consistency.* For example, in case of MHs that deal with the number of hospital beds in an M-P2P disaster recovery network, high replica consistency would be necessary. In contrast, for MHs involved in file-sharing applications in zoological surveys, lower replica consistency could possibly suffice. Moreover, the ease of maintaining a desired level of replica consistency for any given data item $d_i$ should be estimated from the percentage change in the value of the attribute *Att* (which is updated) of $d_i$ and *not* from the *number* of updates to *Att*. Keeping this in mind, we define a measure NQDC, associated with each data item $d_i$, for reflecting the *effect* of updates on the ease of replica consistency maintenance for $d_i$. When CH periodically receives the update logs from each MH, it analyzes the update logs (including the timestamp values) to determine the consistency (w.r.t. desired consistency) with which queries were answered during the last time interval and computes the value of NQDC for each replica in the system as follows.

$$\begin{aligned} NQDC &= NQ \times C \quad if \;\; C \geq DC \\ &= 0 \quad otherwise \end{aligned} \qquad (1)$$

where $NQ$ indicates the number of queries answered by the replica during recent time intervals, $DC$ represents the desired replica consistency level for the particular application and $C$ represents the consistency level with which the queries were answered by the replica. The values of both $C$ and $DC$ vary between 0 and 1. For computing the value of $C$ for a given replica $r$ of a data item $d$, CH computes the percentage change $\Delta_{Att_r,Att_d}$ which should be applied to the attribute *Att* of $r$ to reflect the update.

$$\Delta_{Att_r,Att_d} = ( \;\; (Val_{Att_r} - Val_{Att_d}) \div Val_{Att_d}) \;\; ) \times 100$$

where $Val_{Att_r}$ and $Val_{Att_d}$ are the values of *Att* for $r$ and $d$ respectively. CH maintains a table $T_{\Delta,C}$ which contains a mapping between ranges of $\Delta_{Att_r,Att_d}$ and values of $C$. Given a particular value of $\Delta_{Att_r,Att_d}$, CH finds the corresponding value of $C$ from $T_{\Delta,C}$ and uses equation (1) to determine the NQDC value for $r$. The values of $C$ corresponding to each range of $\Delta_{Att_r,Att_d}$ are pre-specified and are essentially application-dependent. Additionally, in case multiple attributes of $d$ are updated, the value of $\Delta$ should be calculated as follows:

$$\Delta = ( \;\; \sum_{i=1}^{k} (W_i \times (Val_{i_r} - Val_{i_d}) \div Val_{i_d}) \;\; ) \times 100$$

where $k$ is the number of attributes that are updated. $W_i$ is a value between 0 and 1, which represents the weight of attribute $i$ such that $\sum_{i=1}^{k} W_i = 1$. The values of $W_i$ are pre-specified and depends upon the relative importance of each attribute to the users. $Val_{i_r}$ and $Val_{i_d}$ denote the values of attribute $i$ for the replica $r$ and the original data item $d$ respectively.

### Detection of hotspots

For hotspot detection purposes, each MH $M$ maintains a list data structure $D_L$ of the data items that it owns. Each element of $D_L$ is of the form ($d_i$, $s_{d_i}$, $nu_{d_i}$, $Loc_{access}$), where $d_i$ refers to a specific data item, $s_{d_i}$ is the size of $d_i$ and $nu_{d_i}$ represents the number of times that $M$ updated $d_i$ during recent time intervals. $Loc_{access}$ is a linked list data structure, each entry of which is of the form ($MH_{ID}$, $n_{d_i}$, $t_{d_i}$), where $MH_{ID}$ is the identifier of the MH which accessed $d_i$, $n_{d_i}$ is the number of accesses that $MH_{ID}$ made to $d_i$ during recent time intervals and $t_{d_i}$ denotes the average response time for transmitting $d_i$ to $MH_{ID}$.

As we shall see later, $t_{d_i}$ is used to evaluate the benefit of replicating $d_i$. The elements of $Loc_{access}$ are kept sorted in descending order of $n_{d_i}$ for facilitating quick identification of MHs that frequently access $d_i$. For taking *only* the recent hotspots into account, each MH *periodically* refreshes its $D_L$ by *completely* deleting the information in its $D_L$ and populating $D_L$ with fresh access information. Periodically, each MH sends its $D_L$ to CH. Upon receiving these $D_L$s, CH combines the $D_L$s to create its own $D_L$, which it keeps sorted in descending order of access frequency of the data items (normalized w.r.t. data item size).

### Cost-effectiveness of replicating a data item

Before actually replicating a data item $d_i$ (originally owned by an MH *src*) at an MH *dest*, CH uses a formula as a guideline to determine the cost-effectiveness of performing the replica allocation. (We shall discuss the selection of *dest* in Section 5.) Recall that the data to be replicated has to pass via CH for validation purposes. The communication cost $C_{d_i}$ of transmitting $d_i$ from *src* to *dest* via CH is computed as follows.

$$C_{d_i} \; = \; (\sum_{k=1}^{n_{hop}} \, (s_{d_i}/B_k\,)\,) \; + \; (\; s_{d_i} \; / \; B_{CH,dest}\;) \qquad (2)$$

where $s_{d_i}$ refers to $d_i$'s size and $B_k$ refers to the transfer rates of the respective connections between *src* and CH that $d_i$ must 'hop' through to reach CH, $n_{hop}$ refers to the number of hops required by $d_i$ to reach CH and $B_{CH,dest}$ is the bandwidth between CH and *dest*.

For calculating the benefit $B_{d_i}$ of replicating $d_i$ at *dest*, CH refers to its $D_L$ to find out $n_{d_i}$ and $t_{d_i}$ corresponding to $d_i$ for *dest* and computes $B_{d_i}$ as follows.

$$B_{d_i} = t_{d_i} \times n_{d_i} \qquad (3)$$

Suppose $Decide_{d_i}$ represents a boolean variable which is 'TRUE' if CH decides to perform the replication and 'FALSE' otherwise. Then, using (2) and (3), we have the following formula:

$$Decide_{d_i} = (B_{d_i} - C_{d_i} \geq TH) \qquad (4)$$

where $TH$ is a pre-defined application-dependent threshold parameter.

## 5 CLEAR: A Novel Dynamic Replica Allocation Scheme for M-P2P networks

This section discusses our proposed CLEAR scheme for effective dynamic replica allocation in M-P2P networks. Recall that each MH maintains a list $D_L$, which contains details concerning the data items stored at itself and periodically, each MH sends its $D_L$ to CH, which CH combines to create its own $D_L$. CH sorts the data items in its $D_L$ in descending order of their $(n_{d_i}/s_{d_i})$, where $n_{d_i}$ and $s_{d_i}$ represent the total number of accesses to a data item $d_i$ and the size of $d_i$ respectively. The data items, for which $(n_{d_i}/s_{d_i})$ exceeds a pre-specified threshold, are put into a list *DataRep*. Now CH traverses *DataRep* and deletes those data items, for which NQDC values fall below a certain threshold, from *DataRep*. This is because a data item $d_i$ having a low NQDC value implies that it would be difficult to maintain the replica consistency for $d_i$, thereby making $d_i$ an unsuitable candidate for replica allocation.

```
Algorithm CLEAR_REPLICA_ALLOCATION
$n_{d_i}$: Number of accesses to data item $d_i$
$s_{d_i}$: Size of data item $d_i$
$D_L$: List maintained by CH concerning information about data
items of all MHs

Sort data items in $D_L$ in descending order of $(n_{d_i}/s_{d_i})$
Select from $D_L$ those data items, whose $(n_{d_i}/s_{d_i})$ exceeds a
certain threshold, into a list DataRep
Traverse DataRep once to delete data items with low values
of NQDC

for each data item $d_i$ in DataRep
    Determine from $D_L$ the MH $MH_{max}$ which made maximum
    number of accesses to $d_i$
    Check MH schedules to create a list of $MH_{max}$'s k-hop
    neighbours
    Create a set $DEST$ consisting of $MH_{max}$ and its k-hop
    neighbours
    Delete MHs with low available memory space from $DEST$
    Delete MHs, which have low load difference with $d_i$'s owner,
    from $DEST$

    for each MH $M$ in $DEST$
        if ( $Decide_{d_i}$ != 'TRUE' )
            Delete $M$ from $DEST$

    Sort the remaining MHs in $DEST$ in ascending order of load
    Select the least loaded MH min into a list Cand
    From $DEST$, add MHs, which have low load difference
    with min, to Cand
    From Cand, select the MH with highest probability of
    availability as the destination MH for storing $d_i$'s replica
end
```

Figure 1: Algorithm for CLEAR replica allocation scheme executed by CH

Given a specific data item $d_i$ in *DataRep*, CH determines a destination MH for storing $d_i$'s replica as follows. First, CH consults its $D_L$ to determine the MH $MH_{max}$ which has made the *maximum* number of accesses to $d_i$ during recent time intervals. Incidentally, even though $MH_{max}$ accesses $d_i$ the maximum number of times, a number of other MHs in the vicinity of $MH_{max}$ may also be interested in accessing $d_i$. Moreover, it may not always be possible for $CH$ to replicate $d_i$ at $MH_{max}$ e.g., due to $MH_{max}$ being overloaded or $MH_{max}$ lacking the memory space for storing $d_i$. We define the set of MHs in the vicinity of $MH_{max}$ as comprising all the k-hop neighbours of $MH_{max}$. Results of our preliminary performance studies to determine the value of $k$ revealed that $k$=3 provides good performance for CLEAR. Hence, CH checks the schedules of all the MHs and considers $MH_{max}$ and the MHs that would be in the close vicinity of $MH_{max}$ in the near future as constituting the potential candidate set $DEST$ of MHs, where $d_i$ may be replicated.

CH traverses each MH in $DEST$ to delete those MHs, which have low available memory space. Then CH deletes all those MHs, whose load difference with the owner of $d_i$, falls below a pre-specified threshold. Next, CH computes the value of $Decide_{d_i}$ for each of the remaining MHs in $DEST$ and deletes from $DEST$ those MHs for which $Decide_{d_i}$ evaluates to 'FALSE'. Furthermore, CH sorts the remaining MHs in $DEST$ in ascending order of load. Finally, CH puts the least loaded MH $min$ of $DEST$ as well as other MHs, whose load difference with $min$ is not significant, into a list $Cand$. The MH in $Cand$ with highest probability of availability is selected as the destination MH for storing $d_i$'s replica. Notably, over a period of time, CH will know the probability of availability of the MHs in the M-P2P network by keeping records of such MH availability information in its log files. The algorithm for CLEAR replica allocation executed by CH is depicted in Figure 1.

For performing query redirection to replicas, CH first identifies the set *ReDirect* of MHs, which store a replica of the queried data item $d_i$. Then CH deletes those MHs from *ReDirect*, which have low load difference with the owner of $d_i$ and/or which are overloaded and/or low probability of availability. CH sorts the remaining MHs in *ReDirect* to select the least loaded MH $m$ into a set $S$. All the other MHs, whose load difference with $m$ is low, are also added to set $S$. Now CH sums up the NQDC values for all replicas (during the last time interval) stored at each of the MHs in $S$ and redirects the query to that MH of $S$, which has the highest total NQDC value, any ties being resolved arbitrarily.

## 6 Performance Evaluation

This section reports the performance evaluation of CLEAR. The MHs move according to the *Random waypoint model* [2] within a 1000 metre ×1000 metre area. A total of 200 data items in the entire network is uniformly distributed among the 50 MHs i.e., each MH owns 4 data items. Each query is a request for one of the data items residing in the entire system. We used the Zipf distribution for determining the number of queries to be directed to each MH. CH performs replica allocation *periodically* i.e., after every $RP$ queries that pass via CH. As in [9], we assume that the network topology does *not* change significantly during replica allocation since it requires only a few seconds. Table 1 summarizes the parameters used for our performance study. In Table 1, query interarrival rate of 100 queries/s implies that 100 queries pass via CH every second.

The metrics for our performance study are *average response time* (**ART**) of a query, *percentage success ratio* (**SR**) and **traffic** during the replica allocation period. We define SR as ( $Q_C/Q_T$ )*100, where $Q_C$ is the number of queries that were answered with the desired consistency level and $Q_T$ represents the total number of queries. We define traffic as the total hop count during the course of the experiment. We use the **E-DCG+** approach in [9] (discussed in Section 2) as reference since the E-DCG+ approach is among the most influential approaches for replica allocation in mobile ad-hoc networks. E-DCG+ is executed at every replica allocation period. As a baseline, we also compare CLEAR with an approach, designated as **NoRep**, which does *not* perform any replication.

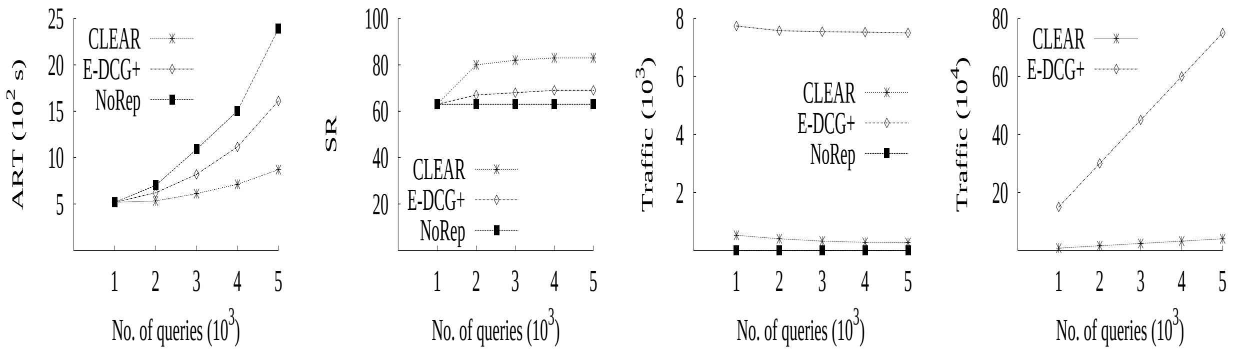
### Performance of CLEAR

We conducted a simulation experiment using the default values of the parameters shown in Table 1. CH performs replica allocation after every 1000 queries, hence the results in Figure 2a indicate comparable ART for all three approaches upto the time the first 1000 queries are issued. The difference in ART between CLEAR and E-DCG+ keeps on increasing as the number of queries increase. This occurs because CLEAR allocates replicas to relatively underloaded MHs and redirects queries for 'hot' data items to underloaded MHs that store those 'hot' data items. In contrast, since E-DCG+ does not consider load, it may allocate replicas to the overloaded MHs, hence queries may often need to retrieve data items from overloaded MHs, thereby incurring higher ART due to the large job queues at these MHs. The phenomenon of high waiting times in the job queues of overloaded MHs is even more pronounced in case of NoRep than for E-DCG+. The experimental log files revealed that CLEAR outperformed E-DCG+ and NoRep by upto 46% and 64% respectively in terms of ART.

Figure 2b indicates that SR remains relatively constant in case of NoRep since it depends only upon the probability of availability of the MHs. E-DCG+ provides better SR than NoRep because an MH being unavailable in case of NoRep implies that the query fails. But for E-DCG+, replication increases the probability of the query being answered by at least one of the MHs, given the low values of DC and WP for this experiment. After replica allocation, CLEAR provides higher SR as compared to E-DCG+ due to two reasons. First, CLEAR directs any query concerning a data item $d_i$ to an underloaded MH, which had provided the best NQDC value for $d_i$ during the previous time interval, thereby increasing the probability of obtaining higher SR. In contrast, E-DCG+ directs queries to any MH which contains $d_i$ without considering consistency issues. Second, since E-DCG+ may allocate replicas to overloaded MHs, updating data items in such MHs require more time since the updates

| Parameter | Default value | Variations |
|---|---|---|
| Number of MHs ($N_{MH}$) | 50 | |
| Zipf factor (ZF) | 0.9 | |
| Query Interarrival rate | 100 queries/s | |
| Bandwidth between MHs | 28 Kbps to 100 Kbps | |
| Size of a data item | 50 Kb to 350 Kb | |
| Memory space of each MH | 1 MB to 1.5 MB | |
| Probability of availability of an MH | 50% to 85% | |
| Number of queries | 5000 | |
| Replica allocation period $RP$ ($10^2$ Queries) | 10 | |
| Write probability (WP) | 20 | |
| Total number of data items | 200 | |
| Desired consistency level (DC) | 0.3 | |
| Size of message headers and meta-information | 220 bytes | |
| Speed of an MH | 1 metre/s to 10 metre/s | |
| Communication range of MHs (except CH) | Circle of 100 metre radius | |

Table 1: Parameters used in Performance Study



(a) ART (b) SR (c) Replica allocation Traffic (d) Querying Traffic

Figure 2: Performance of CLEAR

have to wait in the possibly large job queues of these overloaded MHs. Hence, the probability of answering queries with obselete data increases in case of E-DCG+. For both CLEAR and E-DCG+, SR changes only very slightly after the first replica allocation period because most of the required replica allocations were already performed in the first period.

During replica allocation, E-DCG+ requires every MH to broadcast its RWR values to every MH, while CLEAR requires each MH to send only one message to CH, thereby explaining the results in Figure 2c. Figure 2d depicts the querying traffic for CLEAR and E-DCG+. In case of E-DCG+, querying has to proceed by means of broadcast, thereby resulting in extremely high querying traffic for E-DCG+. However, for CLEAR, the querying traffic is significantly lower than that of E-DCG+ due to two reasons. First, CH can reach any MH within one 'hop'. Second, since CLEAR requires the queries as well as the corresponding query results to pass via CH, the total number of hops is usually much lower than that of broadcast. Observe that the traffic increases with increasing number of queries because larger number of queries imply higher traffic. Since our main focus is replica allocation, we shall not discuss querying traffic any further.

## 7 Conclusion

The ever-increasing popularity and proliferation of mobile computing technology strongly motivate applications involving M-P2P networks. Notably, network partitioning may occur *frequently* in M-P2P networks due to user movement and/or users switching 'on' or 'off' their mobile devices, thereby decreasing data availability . We have envisaged the M-P2P network as a cluster of MHs, which has a cluster head for facilitating data validation and replica allocation. We have proposed the CLEAR scheme for dynamic replica allocation in M-P2P networks to improve data availability. For performing effective replica allocation, CLEAR considers a metric NQDC and load as criteria, and uses knowledge of users' schedules. Results of our extensive performance evaluation demonstrate that CLEAR is indeed effective in improving data availability in M-P2P networks. In the near future, we plan to address continuous queries in an M-P2P network.

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

[4] L.D. Fife and L. Gruenwald. Research issues for data communication in mobile ad-hoc network database systems. *Proc. SIGMOD Record*, 32(2):22–47, 2003.

[5] Gnutella. http://www.gnutella.com/.

[6] 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.

[7] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. *Proc. IEEE INFOCOM*, 2001.

[8] T. Hara. Replica allocation in ad hoc networks with periodic data update. *Proc. MDM*, 2002.

[9] T. Hara and S. K. Madria. Dynamic data replication using aperiodic updates in mobile ad-hoc networks. *Proc. DASFAA*, 2004.

[10] A. Helal, A. Heddaya, and B. Bhargava. Replication techniques in distributed systems. *Kluwer Academic Publishers*, 1996.

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

[12] S. Kapadia, S. Ghandeharizadeh, and B. Krishnamachari. Comparison of replication strategies for content availability in c2p2 networks. *Proc. MDM*, 2005.

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

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

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

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

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

[18] E. Pitoura and B. Bhargava. Maintaining consistency of data in mobile distributed environments. *Proc. ICDCS*, 1995.

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

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

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

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