CLEAR: An Efficient Context and Location-based Dynamic Replication Scheme 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@umr.edu

Abstract. We propose CLEAR (Context and Location-based Efficient Allocation of Replicas), a dynamic replica allocation scheme for improving data availability in mobile ad-hoc peer-to-peer (M-P2P) networks. CLEAR exploits user mobility patterns and deploys a super-peer architecture to manage replica allocation efficiently. CLEAR avoids broadcast storm during replica allocation and eliminates the need for broadcast-based querying. CLEAR considers different levels of replica consistency and load as replica allocation criteria. Our performance study indicates CLEAR's effectiveness in improving data availability in M-P2P networks.

1 Introduction

In a mobile ad-hoc peer-to-peer (M-P2P) network, mobile hosts (MHs) interact with each other in a peer-to-peer (P2P) fashion. Rapid advances in wireless communication technology coupled with the ever-increasing popularity of mobile devices (e.g., laptops, PDAs, mobile phones) motivate M-P2P network applications. However, data availability in M-P2P networks is lower than in traditional stationary networks because of frequent network partitioning due to user movement and/or users switching 'on' or 'off' their mobile devices. Notably, data availability is less than 20% even in a wired environment [18]. Hence, dynamic replica allocation becomes a necessity to support M-P2P network applications such as disaster-recovery and sales applications. Suppose a group of doctors, each of whom has a schedule, are moving in an earthquake-devastated region, where communication infrastructures (e.g., base stations) do not exist. They need to share data (e.g., number of injured people, number of empty stretchers, number of fatalities) with each other on-the-fly using mobile devices. Similarly, moving salespersons, who generally have a schedule, need to share total sales profits by means of mobile devices. Incidentally, absolute consistency is not a requirement in such applications [15]. For simplicity, this work considers only numerical data.

Traditional replication techniques [9] and mobile replication techniques[20, 14, 10], which assume *stationary networks*, do not address frequent network partitioning issues. P2P replication services are not 'mobile-ready' [5, 17] as current P2P systems have mostly ignored data transformation, relationships and network characteristics. Understandably, changes in data with location and time create new research problems [3]. In essence, replication in M-P2P networks requires fundamentally different solutions [1, 11] than in [16, 12, 14] due to free

movement of MHs and wireless constraints. Interestingly, the techniques in [6–8] consider frequent network partitioning w.r.t. replication in mobile ad-hoc networks (MANETs), but they do not exploit user mobility patterns. Hence, they may unnecessarily replicate at MHs which would soon leave the region, while being unable to replicate at MHs that would soon enter the region. Moreover, they incur high traffic due to possible broadcast storm as the MHs exchange replica allocation-related messages with each other. Furthermore, they determine the reallocation period in an ad-hoc manner since none of the MHs has a global view, and perform replica allocation during every such reallocation period, regardless of whether it is actually required, thereby increasing traffic.

We propose CLEAR (Context and Location-based Efficient Allocation of Replicas), a dynamic replica allocation scheme for improving data availability in M-P2P networks. CLEAR deploys a super-peer architecture, the *superpeer* (**SP**) being an MH, which generally does not move outside the region and which has maximum remaining battery power and processing capacity. Since the M-P2P network covers a relatively small area, the total number of MHs can be expected to be low, thereby avoiding scalability problems. As we shall see later, this architecture avoids broadcast storm during replica allocation, eliminates broadcast-based querying since each MH knows about data and replicas stored at other MHs, and preserves P2P autonomy as queries need not pass via SP. SP knows the *schedule* of every MH comprising the MH's mobility pattern and the data items that the MH is likely to access at different times. This makes it possible for CLEAR to replicate at MHs that would soon enter the region, while avoiding replication at MHs that would soon leave the region, thereby facilitating better resource utilization, likely better query response times and increased data availability. Incidentally, if reallocation period does not match the MH moving pattern, data accessibility decreases. SP is able to determine a near-optimal reallocation period based on *global* information of MH schedules, hence it can better manage replica allocation. Finally, unlike existing works, CLEAR considers load, different levels of replica consistency and unequal-sized data items. Our performance study indicates that CLEAR indeed improves data availability in M-P2P networks with significant reduction in query response times and communication traffic as compared to some recent existing schemes.

2 Related Work

The work in [12] proposes a suite of replication protocols for maintaining data consistency and transactional semantics of centralized systems. The protocols in [11] 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 [14] discusses replication in distributed environments, where connectivity is partial, weak, and variant as in mobile information systems. Existing systems in this area include ROAM [16], Clique [17] and Rumor [5], while a scalable P2P framework for distributed data management applications and query routing has been presented in [13]. 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 [3], the aim being to devise a fully decentralized robust communication scheme for providing probabilistic guarantees as opposed to strict consistency. The work in [1] investigates replication strategies for designing highly available storage systems on highly unavailable P2P hosts.

The proposals in [6–8] 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 MANETs. Among these approaches, the E-DCG+ approach [8] 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 [6–8] is not suitable for our application scenarios described earlier since it does not consider user mobility patterns, load sharing and tolerance to weaker consistency.

3 Context of the Problem

In CLEAR's super-peer architecture, each MH maintains a list of the data items that it owns and the recent read-write logs (including timestamps) for hotspot detection purposes. Each data item d is owned by only *one* MH, which can update d *autonomously* anytime; other MHs cannot update d. Memory space at each MH, bandwidth and data item sizes may vary. To optimize memory space usage, MHs delete infrequently accessed replicas. We assume location-dependent data access [19] i.e., an MH is in region X will access data only from the MHs in X. SP backs up information using the Internet as an interface to handle failures and we assume that some of the MHs have access to the Internet for backup purposes. If SP fails or network partitioning occurs, these MHs can connect to the Internet to obtain information, thereby enabling them to act as SP.

We define the **load** of an MH M as $(\sum_{i=1}^{N_d}(n_{d_i} \div s_{d_i})) \div \eta_i)$, where N_d is the total number of data items in M's job queue, n_{d_i} is data item d_i's recent access frequency and s_{d_i} denotes d_i's size. Since bandwidth among MHs may vary, we use η_i to normalize load w.r.t. bandwidth. We compute η_i as $(B_{P_i} \div B_{min})$, where B_{P_i} represents the 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}. To estimate the *effect* of updates on the ease of maintaining replica consistency for any data item d, we compute a measure **NQDC** for each replica of d as $(NQ \times C)$, if $C \geq DC$, and 0 otherwise. NQ indicates the number of queries recently answered by the replica, DC represents the value of **desired consistency (DC)** and C is the consistency with which queries were answered by the replica. We use three different levels of replica

consistency, namely high, medium and low. SP maintains a table $T_{\epsilon,C}$, which contains the following entries: (x%, high), (y%, medium), (z%, low), where x, y, z are error-bounds, whose values are application-dependent and pre-specified by the system at design time. We assign the values of C for high, medium and low consistency as 1, 0.5 and 0.25 respectively. Similarly, the value of DC can be 'high', 'medium' or 'low' depending upon the application under consideration.

In practice, MH owners do not move randomly since they have some *schedule*. An MH M's schedule contains information concerning M's location during any given time period T and the data items required by M during T. Each MH owner initially sends his schedule to SP and if later on, his schedule changes significantly, he will keep SP updated about these changes by piggybacking such information onto replica allocation-related messages to SP. Thus, SP is able to exploit MH schedules for replica allocation purposes. Notably, even if an MH fails to adhere strictly to its schedule, the MH schedule information would still help SP in determining the MH's general direction of motion.

4 CLEAR: A Context and Location-based Dynamic Replica Allocation Scheme for M-P2P networks

This section discusses the CLEAR scheme. *Periodically* every TP time units, each MH sends a message containing the read-write log D_MH (including timestamps) of its own data items and the read log with timestamps for replicas R_MH (residing at itself) for the previous period, as well as its available memory and its load status to SP. SP combines the information in all these D_MHs and R_MHs to create D_SP (sorted in descending order of data item access frequency) and R_SP respectively. Then SP executes the replica allocation algorithm depicted in Figure 1.

As Figure 1 indicates, data items (of D_SP), whose access frequency exceeds a certain threshold ψ, are selected into a list *Rep*. Notably, $\psi = T_{Acc}$ / T_{num}, where T_{Acc} is the total of all the access frequencies of all the data items in D_SP and T_{num} is the total number of data items in D_SP. In Line 2 of Figure 1, SP computes the total NQDC value for a data item d by using the table $T_{\epsilon,C}$ and R_MH to compute the NQDC value for each replica of d and then summing up these NQDC values. Any data item d with low total NQDC value is deleted from *Rep* because low NQDC value implies that d is frequently updated, thereby making it more difficult to maintain the consistency of its replicas. As Line 3 implies, CLEAR performs replica allocation only if *Rep* is a non-empty list. Thus, SP allocates replicas on-the-fly only when necessary and not during every time period. Furthermore, CLEAR tries to replicate a data item d_i at the MH MH_{max}, which had the highest access frequency for d_i, or at one of M's k-hop neighbours. (A preliminary performance study revealed that k=3 provides good performance for CLEAR.) 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 SP 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. Hence, SP 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.

In Line 10 of Figure 1, Dec_d is 'TRUE' if ($B_d - C_d \geq TH$), where B_d is benefit, C_d is cost and TH is a pre-defined application-dependent threshold parameter. SP consults its D_SP to find out n_d and t_d, where n_d and t_d are the number of times d was accessed during the last period and t_d is the time taken for each of those accesses respectively. Then SP computes B_d as ($t_d \times n_d$). C_d involves transmitting d from *src* to *dest* via SP and is computed as $((\sum_{k=1}^{n_{hop}} (s_d/B_k)) + (s_d / B_{SP,dest}))$, where s_d is d's size and B_k refers to transfer rates of connections between *src* and SP that d must 'hop' through to reach SP, n_{hop} is the number of hops required by d to reach SP and $B_{SP,dest}$ is the bandwidth between SP and *dest*. Finally, observe how CLEAR takes the respective loads and available memory space of the MHs into consideration, while allocating replicas.

Algorithm *CLEAR_REPLICA_ALLOCATION*
D_SP: Sorted list maintained by SP concerning access information of data items of all MHs

1) Select from D_SP data items, whose access frequency exceeds threshold ψ, into list *Rep*
2) Traverse *Rep* once to delete data items with low total NQDC values
3) if *Rep* is non-empty
4) for each data item d in *Rep*
5) Determine from D_SP the MH MH_{max} which made maximum number of accesses to d
6) Check MH schedules to create a list of MH_{max}'s k-hop neighbours
7) Create a set $DEST$ consisting of MH_{max} and its k-hop neighbours
8) Delete MHs with low available memory space from $DEST$
9) Delete MHs, which have low load difference with d's owner, from $DEST$
10) for each MH M in $DEST$ { if (Dec_d != 'TRUE') Delete M from $DEST$ }
11) Select the least loaded MH from $DEST$ as destination MH
end

Fig. 1. Algorithm for CLEAR replica allocation scheme executed by SP

After performing replica allocation, SP sends a broadcast message to all MHs informing them about the replica allocations that have been performed, the data items at each MH, the NQDC values of the replicas stored at each MH, and load and availability status of each MH. When an MH misses a broadcast from SP (e.g., due to it having been switched 'off' or due to it newly joining the network), it contacts its neighbours to obtain the latest information broadcast by SP. When a query Q arrives at any MH M, M answers Q if it stores the queried data item d or its replica. Otherwise, M identifies the set $DirQ$ of MHs, which store d's replica. Then M deletes (from $DirQ$) overloaded MHs, whose load exceeds the average system load since our aim is to replicate *only* at underloaded MHs. M sorts the remaining MHs in $DirQ$ 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.

From S, M selects the MH, which had the highest NQDC value for d's replica during the last period, for redirecting Q, any ties being resolved arbitrarily. Observe the inherent P2P autonomy in CLEAR's architecture in that queries do not need to pass via SP since any MH has adequate information to redirect queries. Given a total of N MHs, our approach incurs during a given period at most $2N$ messages (1 message from each MH to CH, and 1 message from SP to each MH if SP decides to perform replica allocation) i.e., O(N) messages. In contrast, for a distributed architecture without an SP, each MH would have to broadcast its list of data items and replicas to every MH periodically to avoid flooding broadcast-based query retrieval, thereby resulting in O(N^2) messages for a given period.

5 Performance Evaluation

The MHs move according to the *Random waypoint model* [2] with speeds varying from 1 metre/s to 10 metres/s within a 1000 metre $\times$1000 metre area. Communication range of MHs (except SP) is a circle of 100 metre radius. MH memory space varies from 1 MB to 1.5 MB and each MH owns 4 data items, whose sizes vary from 50 Kb to 350 Kb. Each query requests one data item. Bandwidth between MHs varies from 28 Kbps to 100 Kbps, while probability of availability of an MH varies from 50% to 85%. Message header size is 220 bytes. Network topology does *not* change significantly during replica allocation since it requires only a few seconds [8]. 10 queries are issued in the network every second, the number of queries to be directed to each MH being determined by the Zipf distribution.

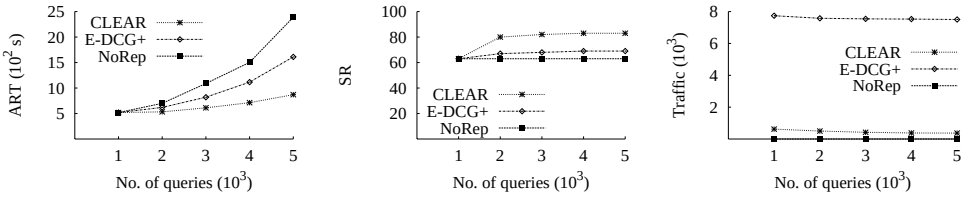
Performance metrics are *average response time* (**ART**) of a query, *percentage success ratio* (**SR**) and **traffic** (i.e., total hop-count) during replica allocation. SR is (Q_{DC}/Q_T)*100, where Q_{DC} and Q_T denote number of queries answered with the desired consistency level and total number of queries respectively. As reference, we adapt the **E-DCG+** approach [8] to our scenario since it is among the most influential replica allocation approaches for MANETs. E-DCG+ is executed at every replica allocation period. As a baseline, we compare CLEAR with an approach **NoRep**, which performs no replication. Table 1 summarizes our performance study parameters. Recall from Section 4 that TP is the time interval at which each MH sends its information to SP, based on which SP decides whether to allocate replicas, hence TP is *not* necessarily the reallocation period.

Figure 2 depicts the average response time for a given number of queries for default values of the parameters in Table 1. As replica allocation is done every 100 seconds (i.e., after every 1000 queries since 10 queries are issued per second), Figure 2a indicates comparable ART for all three approaches for the first 1000 queries. Subsequently, the difference in ART between CLEAR and E-DCG+ keeps on increasing due to two reasons. First, unlike E-DCG+, CLEAR considers MH mobility patterns, hence it is capable of allocating replicas at MHs that would soon enter the region, while avoiding MHs which would soon depart from the region. Second, CLEAR allocates replicas to relatively underloaded

Parameter	Default value	Variations
Number of MHs (N_{MH})	50	10, 20, 30, 40
Zipf factor (ZF)	0.9	0.1, 0.3, 0.5, 0.7
Time interval (10^2 seconds) at which each MH sends message to SP (TP)	1	2,3,4,5
Write probability (WP)	20	10,30,40
Desired consistency level (DC)	Low	Medium, High

Table 1. Parameters used in Performance Study

MHs and redirects queries to replicas stored at underloaded MHs. However, since E-DCG+ does not consider load, it may allocate replicas to overloaded MHs, thereby incurring higher ART due to large job queues. Since NoRep does not perform replication, load imbalance is even more pronounced in case of NoRep than for E-DCG+. Experimental log files revealed that CLEAR outperformed E-DCG+ and NoRep by upto 46% and 64% respectively in terms of ART.

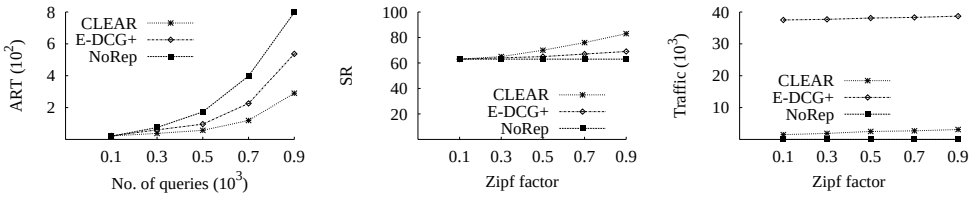


(a) ART (b) SR (c) Allocation Traffic

Fig. 2. Performance of CLEAR

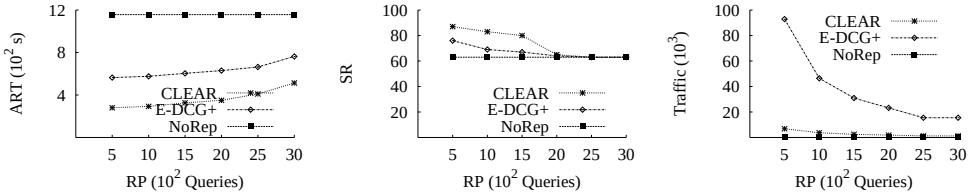
In Figure 2b, CLEAR provides higher SR than E-DCG+ due to two reasons. First, unlike E-DCG+, CLEAR considers consistency issues while directing a query (since it uses NQDC values). Second, updates to replicas are likely to be faster for CLEAR than for E-DCG+ since CLEAR allocates replicas to underloaded MHs, while E-DCG+ may allocate replicas to overloaded MHs with large job queues. For both CLEAR and E-DCG+, SR changes only very slightly after the first replica allocation period because most of the required replica allocations had already been performed in the first period. For NoRep, SR remains relatively constant since it depends only upon the probability of availability of the MHs. Both CLEAR and E-DCG+ provide better SR than NoRep because they perform replication, which increases data availability. 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 CLEAR requires each MH to send only one message to SP and SP to send one broadcast message to all MHs, thus incurring $O(N_{MH})$ messages, which explains the results in Figure 2c.

Effect of variations in the workload skew: Figure 3 depicts the results when the zipf factor (ZF) is varied. For high ZF values (i.e., high skew), CLEAR outperforms E-DCG+ in terms of ART and SR due to the reasons explained for Figure 2. As the skew decreases, the effect of CLEAR's load-based replica allocation becomes less pronounced, hence performance gap between CLEAR and E-DCG+ keeps reducing. Figure 3c's explanation is same as that of Figure 2c.



(a) ART (b) SR (c) Traffic

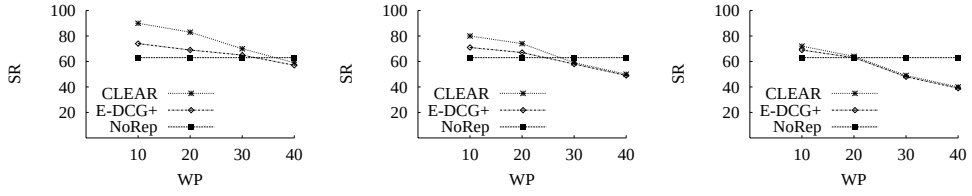
Fig. 3. Effect of variations in the workload skew



(a) ART (b) SR (c) Traffic

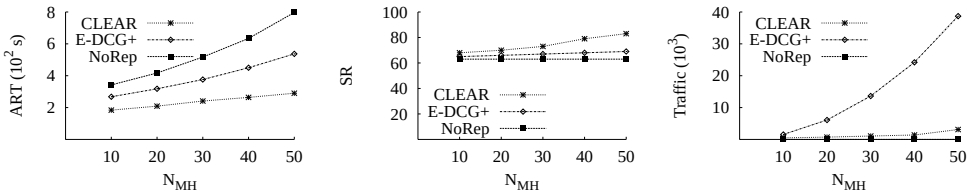
Fig. 4. Effect of variations in the replica allocation periods

Effect of variations in the replica allocation periods: Recall that after every TP queries, SP decides whether to perform replica allocation. Figure 4 indicates the results of varying TP. This experiment was conducted with 50 MHs issuing a total of 6000 queries to each other. Figure 4a depicts the ART for all the 6000 queries. CLEAR provides lower ART than E-DCG+ due to the reasons explained for Figure 2a. At low value of TP, there are higher number of possible replica allocation periods, hence any load imbalance can be corrected relatively quickly, thereby decreasing ART for both CLEAR and E-DCG+. However, as TP increases, lesser replica allocation periods occur, hence ART gap between the approaches reduces. In Figure 4b, SR is higher for low values of TP in case of both CLEAR and E-DCG+ since the replicated data are refreshed more frequently. As TP increases, replicated data are refreshed less frequently, hence SR decreases. In particular, CLEAR and E-DCG+ do not provide lower SR



(a) DC=Low (b) DC=Medium (c) DC=High

Fig. 5. Effect of variations in write probability



(a) ART (b) SR (c) Traffic

Fig. 6. Effect of variations in the number of MHs

than NoRep essentially because this experiment concerned low values of WP and DC. Additionally, increase in TP implies decreased number of possible replica allocation periods, thus explaining the results in Figure 4c.

Effect of variations in write probability (WP): We varied WP to examine the impact on SR. Figure 5 depicts the results. In Figure 5a, as WP increases, SR decreases for both CLEAR and E-DCG+ primarily due to more replicas becoming inconsistent with increasing WP. But, both CLEAR and E-DCG+ still provided higher SR than NoRep upto the point where WP was slightly higher than 30% due to DC being 'low'. However, at WP=40%, both CLEAR and E-DCG+ provided slightly lower SR as compared to that of NoRep due to larger number of replicas becoming inconsistent. As DC increases to 'medium' and 'high', more strict replica consistency is required, hence as WP increases in these cases, it becomes more difficult for both CLEAR and E-DCG+ to answer queries with the desired consistency level. Hence, in Figures 5b and 5c, both CLEAR and E-DCG+ are outperformed by NoRep in terms of SR. However, we believe that this is a small price to pay as compared to the large ART gains achieved by CLEAR.

Effect of variations in the number of MHs: To test CLEAR's scalability, we varied the number N_{MH} of MHs, keeping the number of queries proportional to N_{MH}. The number of possible replica allocation periods was set at 5 for each case. Figure 6 depicts the results. At high values of N_{MH}, CLEAR outperforms E-DCG+, the explanation being same as that of Figure 2. However, as N_{MH} decreases, performance gap between the approaches keeps decreasing due to limited opportunities for replica allocation. Replica allocation traffic for E-DCG+ dramatically decreases with decreasing N_{MH} due to reduced broadcast traffic.

6 Conclusion

We have proposed the CLEAR dynamic replica allocation scheme for improving data availability in M-P2P networks. CLEAR exploits user mobility patterns and deploys a super-peer architecture that facilitates replica allocation, while maintaining P2P autonomy. CLEAR avoids both broadcast storm during replica allocation as well as broadcast-based querying. CLEAR considers different levels

of replica consistency and load as replica allocation criteria. Our performance study indicates that CLEAR indeed improves M-P2P data availability.

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. 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.
6. T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. *Proc. IEEE INFOCOM*, 2001.
7. T. Hara. Replica allocation in ad hoc networks with periodic data update. *Proc. MDM*, 2002.
8. T. Hara and S. K. Madria. Dynamic data replication using aperiodic updates in mobile ad-hoc networks. *Proc. DASFAA*, 2004.
9. A. Helal, A. Heddaya, and B. Bhargava. Replication techniques in distributed systems. *Kluwer Academic Publishers*, 1996.
10. Y. Huang, A. P. Sistla, and O. Wolfson. Data replication for mobile computers. *Proc. ACM SIGMOD*, 1994.
11. B. Kemme. Implementing database replication based on group communication. *Proc. Future Directions in Distributed Computing*, 2002.
12. B. Kemme and G. Alonso. A new approach to developing and implementing eager database replication protocols. *Proc. ACM TODS*, 25(3), 2000.
13. V. Papadimos, D. Maier, and K. Tufte. Distributed query processing and catalogs for peer-to-peer systems. *Proc. CIDR*, 2003.
14. E. Pitoura. A replication scheme to support weak connectivity in mobile information systems. *Proc. DEXA*, 1996.
15. E. Pitoura and B. Bhargava. Maintaining consistency of data in mobile distributed environments. *Proc. ICDCS*, 1995.
16. 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.
17. B. Richard, D. Nioclais, and D. Chalon. Clique: A transparent, peer-to-peer replicated file system. *Proc. MDM*, 2003.
18. S. Saroiu, P.K. Gummadi, and S.D. Gribbler. A measurement study of peer-to-peer file sharing systems. *Proc. MMCN*, 2002.
19. G. Tsuchida, T. Okino, T. Mizuno, and S. Ishihara. Evaluation of a replication method for data associated with location in mobile ad hoc networks. *Proc. ICMU*, 2005.
20. O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *Proc. ACM TODS*, 22(4):255–314, June 1997.