Computing and Informatics, Vol. 27, 2008, 785–798

AN ADAPTIVE CONTEXT-AWARE TRANSACTION MODEL FOR MOBILE AND UBIQUITOUS COMPUTING

Feilong Tang, Minyi Guo, Minglu Li

Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 200240, China
e-mail: {tang-fl, guo-my, li-ml}@cs.sjtu.edu.cn

Ilsun You

School of Information Science
Korean Bible University, South Korea
e-mail: isyou@bible.ac.kr

Revised manuscript received 10 January 2008

Abstract. Transaction management for mobile and ubiquitous computing (MUC) aims at providing mobile users with reliable and transparent services anytime anywhere. Traditional mobile transaction models built on client-proxy-server architecture cannot make this vision a reality because 1) in these models, base stations (proxy) are the prerequisite for mobile hosts (client) to connect with databases (server), and 2) few models considered context-based transaction management. In this paper, we propose a new network architecture for MUC transactions, with the goal that people can get online network access and transaction even while moving around; and design a context-aware transaction model and a context-driven coordination algorithm adaptive to dynamically changing MUC transaction context. The simulation results have demonstrated that our model and algorithm can significantly improve the successful ratio of MUC transactions.

Keywords: Mobile and ubiquitous computing, mobile transaction, context awareness, transaction model, algorithm

1 INTRODUCTION

Mobile and ubiquitous computing (MUC) is a new distributed computing paradigm [1]. Through MUC, people can get online access to their preferred services even while moving around, by sharing computing, communication and information services anytime anywhere. Open MUC environment is prone to failures caused by devices, applications, networks, basic services of ubiquitous systems [3]. To hide the complexity of service processes from users as much as possible, ubiquitous systems have to self-adapt dynamically changing transaction context and intelligently handle failures and recovery [1, 3]. Therefore, advanced transaction management is another key technology [2] to make MUC a reality, ensuring the reliability of MUC systems in the disappearance way. Researchers have discussed the necessity to set up reliable MUC systems, while pointed out that traditional mobile transaction proposals are not directly practicable to MUC transactions because MUC environments have the following features [2, 9–12]:

High mobility. MUC allows users to move anytime anywhere while enjoy services transparently. With the users' movement, nodes, data and services that can be directly accessed keep on continuously changing.

Uncertain support from fixed communication infrastructure. Traditional mobile transaction models rely on wired fixed networks, where transaction managers and main data copies are distributed on fixed hosts (base stations and servers) [10]. In MUC environments, however, computing and service are extended from fixed wired networks to various self-organized wireless networks so that most ubiquitous transactions are completely executed on mobile devices.

Transaction management is highly related to computing paradigm. The above characteristics of MUC environments present severe challenges to MUC transaction management in the following aspects:

Transaction processing architecture. Traditional mobile transaction models are structured on client-proxy-server architecture [10], where mobile hosts (clients) initiate mobile transactions; databases (server) perform application operations in the transactions; and base stations (proxy) bridge the clients and the servers. Unlike traditional mobile computing, MUC environments do not guarantee the support of wired infrastructure (e.g., base stations); and mobile ubiquitous devices may access to both fixed hosts and mobile neighboring nodes in the peer-to-peer way [10].

Context-aware transaction model. Context awareness is a basis to realize transparent ubiquitous transaction services. Therefore, MUC transaction model should be able to self-adapt to dynamical transaction context, and thus provide general support for various ubiquitous applications.

Context-driven transaction management. Highly mobility of MUC users makes context of ubiquitous transactions change continuously. To adapt changing transaction context, MUC transaction management should be aware of context dynamically, and should intelligently optimize the distribution and execution modes of ubiquitous transactions for improving performance of MUC systems and reducing users' intervention as much as possible.

In this paper, we investigate how to solve the above challenges. Firstly, we propose a transaction processing architecture characteristic to MUC environments, targeting on complete support for the ubiquitous access. Next, we present a context-aware MUC transaction model that handles transactions in event-context-action mechanism proposed in this paper. Finally, we design a context-driven coordination algorithm to adaptively manage transactions based on dynamical transaction context.

The rest of this paper is organized as follows. In the next section, we review related work. Section 3 proposes an architecture of MUC transaction processing. Section 4 presents a context-aware MUC transaction model. In Section 5, we present a context-driven transaction coordination algorithm. Experiments and evaluation on our model and algorithm are reported in Section 6. Finally, Section 7 concludes this paper with a discussion on our future work.

2 RELATED WORK

Traditional mobile transaction models focused on variable bandwidth, network disconnection, replication, synchronization, hand-off and resource restriction of mobile hosts. Clustering [6] groups semantics-related databases within a cluster. Each piece of data is kept in two versions: weak consistency (local consistency) version for weak transactions and strict consistency (global consistency) version for strict transactions. Strict and weak transactions are executed when a MH (mobile host) is strongly and weakly connected with fixed networks. Similar to clustering, two-tier replication [5] maintains a master copy for base transactions and multiple replicated copies for tentative transactions. HiCoMo (High Commit Mobile) [7] keeps two kinds of tables: base tables for base transactions and aggregate tables for HiCoMo transactions. Moreover, Moflex [4] submits subtransactions to mobile transaction managers located in base stations, which then submit the subtransactions to databases. Each Moflex transaction is composed of a set of dependency relationships, hand-off rules and expected final states. These mobile transaction models all require the support of base stations; however, base stations can not be a prerequisite for online MUC transactions. In addition, a MUC transaction model has to adapt dynamical transaction context. Therefore, the above mobile transaction models do not work well in MUC environments.

There have been some reports on MUC data and transaction management. Franklin [2] discussed requirements for data management in MUC environments, and analyzed how to manage data according to MUC features from three aspects: support for mobility, context awareness and support for collaboration. MoGATU [10] set up a data management framework for MUC, which abstracts all P2P devices in information providers, information consumers and information managers. Unfortunately, these proposals did not solve basic issues of MUC transactions especially on adaptive transaction model and context-driven transaction management.

3 MUC TRANSACTION PROCESSING ARCHITECTURE

MUC aims at providing mobile users with ubiquitous online services. Therefore, transaction processing in MUC environments should not be restrictive to fixed base stations because such an architecture is easy to cause the following unavailable service problem which severely restricts users' mobility.

3.1 Unavailable Service Problem

In client-proxy-server based traditional mobile transaction models, base stations connected to fixed network are at the center, working as gateways among mobile clients and fixed servers as well as mobile transaction managers in most models. Transactional operations are mainly executed on fixed database servers. MHs can communicate with base stations only when they locate within individual cells, i.e., MHs cannot access to database servers if the MHs move out of the range of any wireless cell. We call such a situation an *unavailable service problem.*

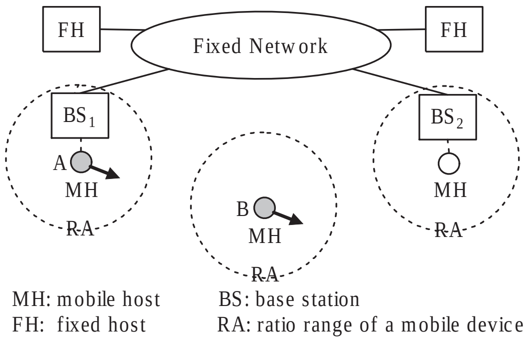

Fig. 1. An example of unavailable service problem

We illustrate the unavailable service problem by an example shown in Figure 1. A MH can communicate with BS_1 when it locates at the place A. However, it cannot connect to any base station if it moves to the place B because the radio range of the MH is not able to cover any base station.

3.2 A Robust Architecture for MUC Transactions

To make online ubiquitous services a reality, we argue that emerging MUC systems should include various self-organized wireless mesh networks, each device maintaining multiple connections with neighboring nodes. Accordingly, we propose an architecture for MUC transactions, as shown in Figure 2, where mobile devices that support the same wireless communication protocols (e.g., Wi-Fi) automatically discover neighbors and interconnect with them to set up wireless mesh networks.

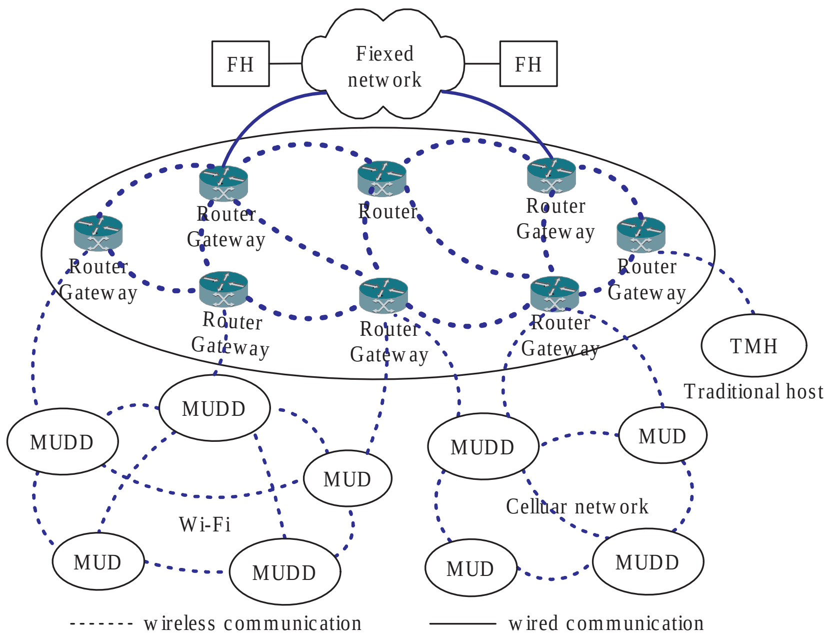

Fig. 2. A mobile transaction architecture for MUC environment

The proposed network architecture is composed of three layers: fixed network in the top layer, wireless mesh routing backbone in the middle layer, and wireless mesh sub-networks in the bottom layer. This is a layered full mesh network, each wireless node can forward data for other nodes so that any two nodes may communicate by multiple hops. In Figure 2, dashed and solid lines represent wireless and wired communications, respectively; thicker lines mean that corresponding links have higher bandwidth; FH is a fixed host connected to a fixed network; MUD refers to a mobile ubiquitous device without any database while MUDD with database(s); TMH is a traditional mobile host without mesh functionalities.

MUDs (e.g., smart mobile phones) and MUDDs (e.g., powerful laptops) interconnect in wireless links to establish low-level wireless mesh networks automatically and dynamically. MUDs only initiate global MUC transactions as clients while MUDDs can not only initiate transaction requests but also execute subtransactions.

In this paper, we specify a MUC transaction is the one that is initiated by any mobile device (MUD or MUDD), and entirely executed by mobile MUDD nodes. A detailed definition of MUC transaction model will be presented in the following section.

4 CONTEXT-AWARE TRANSACTION MODEL

4.1 Context of MUC Transactions

Transaction context includes information from physical space, information space and human activities relevant to transaction processing. Summarily, context of MUC transactions mainly concerns the following dimensions:

- Wireless network: connectivity, bandwidth, delay, ratio of losing packets, cost, stability.
- Mobile device: computing_capacity, available_memory, available_battery, available_data, available_cache, security.
- Location: longitude and latitude (or relative position).
- User: profile, purpose, requirements.
- Time: starting_time and ending_time.

This paper focuses on how to manage MUC transactions based on changing transaction context because of page limitation. We will present how to collect, manage and reason the context of MUC transactions in another paper.

4.2 Context-Aware Transaction Model

Definition 1 (MUCT)**.** A MUC transaction (MUCT) is a 6-tuple MUCT = (T, CT, ECA, D, TS, FS), where:

- $\mathrm{T} = \{\mathrm{T}_i \mid 1 \leqslant i \leqslant n\}$: the set of all subtransactions in a MUCT;
- $\mathrm{CT} = \{\mathrm{CT}_i \mid 1 \leqslant i \leqslant n\}$: the set of compensating transactions of all subtransactions;
- $\mathrm{ECA} = \langle \mathrm{ECA}_i \rangle$: the list of ECA rules for all subtransactions;
- $\mathrm{ECA}_i = \langle \mathrm{E}_i, \mathrm{C}_i, \mathrm{A}_i \rangle$: the list of 3-tuple with event, context and action for a specified subtransaction T_i;
- D: the set of dependencies between T_i and T_j (T_i, $\mathrm{T}_j \in \mathrm{T}$);
- $\mathrm{TS} = \{\mathrm{S}_i \mid 1 \leqslant i \leqslant n\}$: the set of states of all subtransactions;
- FS: the set of acceptable final states.

We explain each tuple in MUCT model in more details. T denotes all subtransactions in a MUC transaction. In this paper, we only consider compensable transactions, i.e., each subtransaction T_i ($\mathrm{T}_i \in \mathrm{T}$) can associate a compensating transaction C_i. All compensating transactions C_i ($1 \leqslant i \leqslant n$) of a transaction T make up of CT, the set of compensating transactions.

$\mathrm{ECA} = \langle \mathrm{ECA}_i \mid 1 \leqslant i \leqslant n \rangle$ is a list of ECA rule descriptors, where ECA_i is for the subtransaction T_i and ECA_i has higher priority than ECA_{i+1}. In particular, each $\mathrm{ECA}_i = \langle \mathrm{E}_i, \mathrm{C}_i, \mathrm{A}_i \rangle$ is also a list of 3-tuple (event, context and corresponding action), describing multiple execution models for a subtransaction T_i based on context, where

- $\mathrm{E}_i = \{\mathrm{E}_{ij}\}$: the set of events that occur during the execution of T_i;
- $\mathrm{C}_i = \{\mathrm{C}_{ik}\}$: the set of context associated with a subtransaction T_i. C_i covers five dimensions: wireless network, mobile device, location, user and time. Each dimension of C_i has to meet a specified condition for the execution of T_i.
- $\mathrm{A}_i = \{\mathrm{A}_{il}\}$: the set of actions. For example, a user wants to reserve airplane tickets from travel agent A. If the link is disconnected, transaction manager may try to connect another travel agent B to resume the ticket reservation.

D is the set of dependencies between T_i and T_j (T_i, $\mathrm{T}_j \in \mathrm{T}$). We define two kinds of dependencies: *successful submission dependency* ($\lessdot_s$) and *failed submission dependency* ($\lessdot_f$). $\mathrm{T}_i \lessdot_s \mathrm{T}_j$ means that T_j cannot start to execute until T_i successfully submits. $\mathrm{T}_i \lessdot_f \mathrm{T}_j$ denotes that T_j can start to execute only if T_i fails to commit or is compensated if T_i has committed.

$\mathrm{TS} = \{\mathrm{S}_1, \mathrm{S}_2, \ldots, \mathrm{S}_n\}$ is the set of states of all subtransactions in a MUC transaction. S_i, the state of T_i, is one of five possible states: I, E, S, F and C (see Table 1). Each subtransaction starts with state 'I' and ends with 'S' or 'F'.

Symbol	*State*	*Description*
I	initial state	T_i has not yet started to execute
E	executing state	T_i is executing and has not submitted
S	submitted state	T_i has successfully submitted
F	failed state	T_i has failed to submit and rollbacked to previous state
C	compensating state	compensating transaction C_i of T_i is executing

Table 1. Transaction state description

FS is a special set of states with expected final results. A MUC transaction may has multiple proper final states that the user is ready to accept, with different priorities.

5 CONTEXT-DRIVEN COORDINATION ALGORITHM FOR MUC TRANSACTIONS

For a MUC transaction $\mathrm{T} = \{\mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_n\}$, global transaction T is initiated by a mobile device (called *request node*) and subtransactions are distributed to n mobile devices (called *execution node*), as illustrated in Figure 3. CATM (context-aware

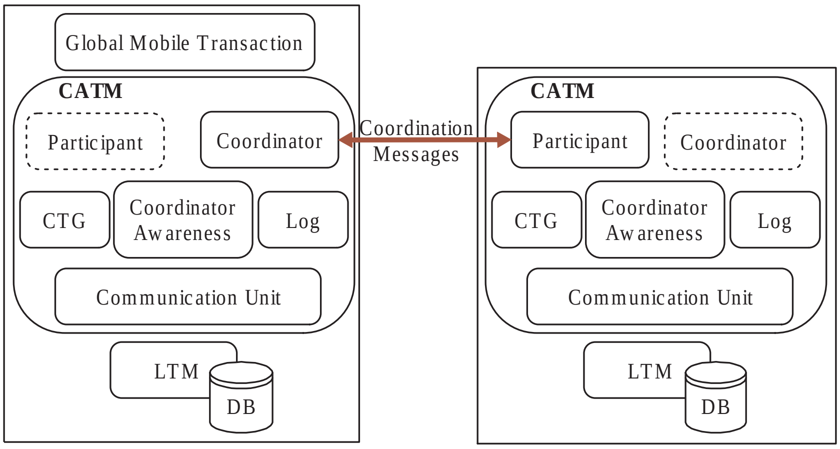

Fig. 3. A MUC transaction system

transaction manager) is responsible for MUC transaction management, where *Coordinator* and *Participant* execute coordination algorithm to orchestrate activities in a MUC transaction. During the execution of MUC transactions, *Context Awareness* module collects and monitors transaction context for dynamically adjusting transaction processing policies. *Log* service records coordination and state information in order to recover from potential failures. *CTG* (compensating transaction generator) automatically generates compensating transactions during the execution of MUC transactions. *Communication Unit* sends and receives messages for transaction processing. LTM (local transaction manager) manages a local subtransaction T_i.

Our context-driven coordination algorithm for the proposed MUC transaction model consists of two parts: *Global_transaction_management* and *Subtransaction(T_i)_execution*, which are executed by a coordinator in a request node and n Participants (see Figure 3) located in n execution nodes.

Global_transaction_management, shown in Figure 4 a), is for coordinating global MUC transactions, where a subtransaction T_i is submitted to an execution node only when 1) it has not executed(S_i=‘I’); 2) corresponding event occurs; and 3) its context C_i is qualified. By *qualified context*, we denote that current context of a transaction T_i satisfies the requirements of T_i’s execution.

A global MUC transaction T is executed until its state achieves one of acceptable final states or any subtransaction state S_i ($1 \leq i \leq n$) is not ‘I’. The order of execution of subtransactions depends on transaction dependency D. The state of each subtransaction is set according to its execution result. Finally, if one of acceptable final states has been achieved, the algorithm confirms all submitted subtransactions. On the other hand, if the algorithm can not achieve any acceptable final state, it requires subtransactions submitted previously to execute their compensating transactions. Note that the algorithm can dispatch each subtransaction to

```
Global_transaction_management
{ TS={I,I,...,I};
  repeat
    select subtransaction $T_i$ based on D;
    if (event occurs) {
      check current context $C_i$;
      num=0;
      while (num<RetryNum)
        if (qualified context) {
          send $T_i$ to corresponding node;
          $S_i$='E';
          wait incoming messages;
          if (message is Successful)
            $S_i$='S';
          else if (message is Failed)
            $S_i$='F'; }
        num++;
      endWhile       }
  until ((TS∈FS) or (any $S_i \neq$ 'I'))
  if (TS∈FS)
    confirm submitted subtransactions;
  else
    compensate submitted subtransactions;}
```

(a) Coordination algorithm in a request node

```
Subtransaction($T_i$)_execution
{ suc:=true;
  repeat
    execute application operations;
    if (fail to execute current operation)
      suc:=false;
  until ((all operations finish) or (not suc))
  if (suc)
  {
    send Successful message to requester;
    wait incoming message;
    if (receive Confirm message)
      report execution results to requester;
    else if (receive Cancel message)
    {
      execute $T_i$'s compensating transaction;
      send Cancelled message;
    }
  } else {
    rollback to previous state;
    send Failed message;
  }
}
```

(b) Coordination algorithm in execution nodes

Fig. 4. Context-driven coordination algorithm for MUC transactions

specified mobile nodes for at most RetryNum times in terms of current transaction context.

Subtransaction(T_i)_execution, depicted in Figure 4 b), actually executes individual subtransactions, under the control of the Coordinator. If a subtransaction is successfully executed, it will be confirmed or compensated in terms of the message from the Coordinator. Otherwise, if failed, it automatically rollbacks to previous system state. To achieve people-oriented MUC, it is also very important to automatically generate compensating transactions. We have proposed some basic ideas in [8].

6 EXPERIMENTS AND EVALUATION

We have implemented a prototype system to test the feasibility of our MUC transaction processing architecture and the effectiveness of our context-aware transaction model and context-driven coordination algorithm.

6.1 Experiment Environment

In our system, there were totally 100 self-organized mobile nodes. Each node acted as a request node as well as an execution node. The features of mobility are simulated by changing link states. Let wireless links among nodes disconnect in the probability *DisconnectProb*. In addition, we modeled system load in the number of concurrent MUC transactions (simplified *NumMobiTran*). These MUC transactions were randomly initiated and concurrently executed in the system. Each of them consisted of two subtransactions. Our current experiments concentrated on how to adapt to changing network connectivity and bandwidth.

We tested and compared two kinds of transactions: context-aware transaction (*CATran*) and non-context-aware transaction (*NonCATran*). For CATran transactions, if context of a subtransaction T_i is unqualified, the transaction manager in a request node resends T_i to other nodes for at most RetryNum (RetryNum > 1) times. On the other hand, NonCATran transaction fails if at least one link among a request node and execution nodes is unqualified. In experiments, we set RetryNum $= 3$.

6.2 Results and Evaluation

Highly mobility and frequent network disconnection significantly decrease the probability of successful transaction commits so that successful ratio (*SucRatio*) of MUC transactions is one of important performance metrics. Therefore, we evaluate MUC transactions in SucRatio which means the ratio of the number of successfully submitted transactions to the number of totally initiated transactions within a given period.

6.2.1 System Load

In this experiment, we varied the number of concurrent MUC transactions from 100 to 500, where link disconnection probability is fixed such that DisconnectProb $= 0.1$. The performance results obtained for the two kinds of transactions CATran and NonCATran are shown in Figure 5. From this figure, we can see that the SucRatio of the system degrades for both strategies as the transaction load increases, and for all ranges of the transaction load CATran performs better than NonCATran. This is because CATran may redispatch a transaction for at most RetryNun times if previous requests failed while NonCATran sends a transaction request only once. It can be expected that the more RetryNum, the higher SucRatio of CATran transactions.

The reason for the decrease in the SucRatio with both execution strategies is that more load on the physical resources caused more heavy data conflicts.

6.2.2 Link State

When a link disconnects or has not enough bandwidth, nodes connected with the link can no longer initiate transaction requests or report execution results. Therefore, for both CATran and NonCATran transactions, link states have significant impact on the SucRatio. In this experiment, we measured the transaction SucRatio by varying the *DisconnectProb* from 0.1 to 0.7 in increments of 0.1. The performance results

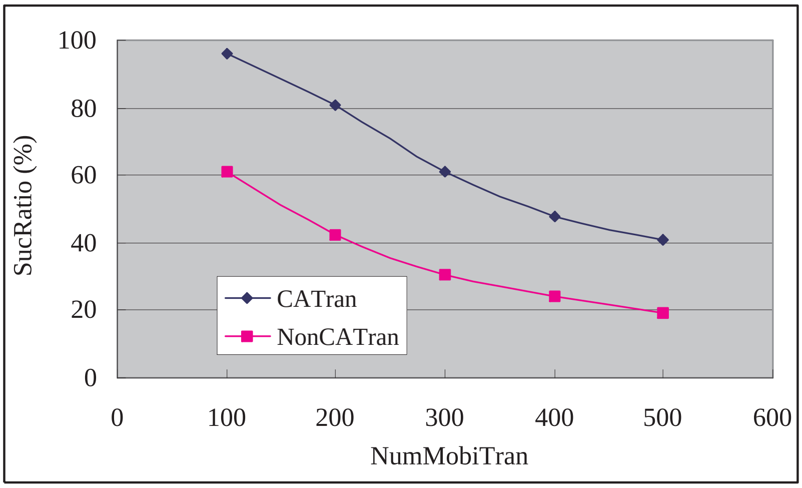

Fig. 5. Successful ratio vs. the number of concurrent MUC transactions

are shown in Figure 6, where the number of concurrent MUC transactions in system was fixed such that NumMobiTran = 50.

As we can see, as the value of link disconnection probability increases, the performance of the system becomes worse with both execution strategies CATran and NonCATran. The reason is that with the increment of failure probability of wireless links, more subtransactions can not be sent to targeted nodes. However, the relative performance of CATran and NonCATran is not affected by the probability of wireless link failure because CATran transactions tried to resend subtransactions to other nodes more times than that in NonCATran transactions.

7 CONCLUSIONS AND FUTURE WORK

We have proposed a new network architecture for MUC transaction processing, presented a context-aware transaction model and designed a context-driven coordination algorithm. The proposed network architecture breakthroughs the structural limitation of traditional mobile transaction models, allowing people to access ubiquitous services anytime anywhere by self-organized wireless networks. Our context-aware transaction model and context-driven coordination algorithm can adapt to dynamical transaction context, significantly improving the successful ratio of MUC transactions.

We are going to investigate light-weight mechanisms to automatically generate compensating transactions for MUC environments. Further, we also consider to design efficient protocols for non-compensable MUC transactions.

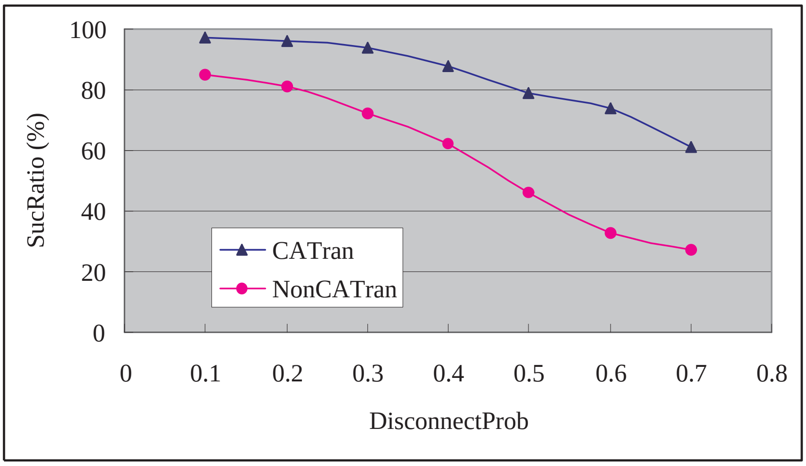

Fig. 6. Successful ratio vs. the disconnection probability of wireless links

Acknowledgements

This work is supported by the National Natural Science Foundation of China (NSFC) (Grant Nos. 60773089 and 60533040), the National High Technology Research and Development Program (863 Program) of China (Grant Nos. 2006AA01Z199 and 2006AA01Z172), and Natural Science Foundation of Shanghai Municipality of China (Grant No. 05ZR14081).

REFERENCES

[1] Weiser, M.: The Computer for the 21st Century. Scientific American, September 1991, Vol. 265, No. 3, pp. 66–75.

[2] Franklin, M.: Challenges in Ubiquitous Data Management. Informatics, 2001, pp. 24–33.

[3] Chetan, S.—Ranganathan, A.—Campbell, R.: Towards Fault Tolerance Pervasive Computing. Technology and Society Magazine, IEEE, Vol. 24, 2005, No. 1, pp. 38–44.

[4] Ku, K. I.—Kim, Y. S.: Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems. Research Issues in Data Engineering (RIDE), 2000, pp. 39–46.

[5] Gray, J.—Helland, P.—O'Neil, P. et al.: The Dangers of Replication and a Solution. ACM SIGMOD Record, Vol. 25, 1996, No. 2, pp. 173–182.

[6] Pitoura, E.—Bhargava, B. K.: Data Consistency in Intermittently Connected Distributed Systems. IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 11, 1999, No. 6, pp. 896–915.

[7] Lee, M.—Helal, S.: HiCoMo: High Commit Mobile Transactions. Kluwer Academic Publishers, Distributed and Parallel Databases (DAPD), Vol. 11, 2002, No. 1, pp. 73–92.

[8] Tang, F. L.—Li, M. L. et al.: Automatic Transaction Compensation for Reliable Grid Applications. Journal of Computer Science and Technology, Vol. 21, 2006, No. 4, pp. 529–536.

[9] Satyanarayanan, M.: Pervasive Computing: Vision and Challenges. IEEE Personal Communications, August 2001, pp. 10–17.

[10] Perich, F.—Joshi, A.—Finin,T. et al.: On Data Management in Pervasive Computing Environments. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, 2004, No. 5, pp. 621–634.

[11] Perich, F.—Joshi, A.—Yesha, Y. et al.: Neighborhood-Consistent Transaction Management for Pervasive Computing Environments. Proceedings of 14th International Conference on Database and Expert Systems Applications (DEXA '03), LNCS 2736, September 2003, pp. 276–286.

[12] Banavar, G.—Beck, J.—Gluzberg, E. et al.: Challenges: An Application Model for Pervasive Computing. Mobile Computing and Networking, 2000, pp. 266–274.

Feilong Tang received his Ph. D. degree in Computer Science and Technology from Shanghai Jiao Tong University in 2005. From May 2004 to June 2005, he researched on grid computing in the University of Hong Kong. Now, he works with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests focus on grid and pervasive computing, distributed transaction processing, wireless sensor networks, computer network and distributed computing.

Minyi Guo received his Ph. D. degree in computer science from University of Tsukuba, Japan. Before 2000, he had been a research scientist of NEC Corp., Japan. He is now a full professor at the Department of Computer Software, The University of Aizu, Japan. His research interests include pervasive computing, parallel and distributed processing, parallelizing compilers and software engineering. He is a member of the ACM, IEEE, IEEE Computer Society, and IEICE.

Minglu Li is a full professor and associate director in Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Now he also is a director of Grid computing center and Web Services research center of Shanghai Jiao Tong University, Grid expert of Ministry of Education, P. R. China, and expert-in-chief of ShanghaiGrid project. His research interests mainly include grid computing, Web Services, wireless sensor network and multimedia computing.

Ilsun You received his M. Sc. and Ph. D. degrees in the Division of Information and Computer Science from the Dankook University, Seoul, Korea in 1997 and 2002, respectively. He is now an assistant professor in the School of Information Science at the Korean Bible University. His research interests include MIPv6 security, key management, authentication and access control. He is a member of the IEEK, KIPS, KSII, and IEICE.