An Energy-Efficient Scheme for Reliable Web Access in Wireless Mobile Environment based on Mobile Agents

H.Y.Hu, Hua Hu
College of Computer and Information Engineering
Zhejiang Gongshang University
Hangzhou City, Zhejiang Province, 310018,China
hhy@mail.zjgsu.edu.cn

Abstract

Abstract: *Web access in mobile wireless environments suffers more challenge because of mobile device's limited storage and the unreliable wireless network environments. It requires some facilities both in the wired network and the terminals so that to overcome the difficulties of unreliable wireless link, mobility and mobile devices capability limitation. This paper presents a fault-tolerance Web page access scheme, which uses mobile agents to facilitate seamless logging of access activities for recovery from failure. Compared with other schemes by simulation tests, our scheme shows its efficiency and reliability.*

1. Introduction

Wireless access to Internet enables a user equipped with a wireless capable device to access the Internet anywhere and perform his desired operation [1]. However, for the limited hardware capability, when performing wireless Web page access, the user usually prefetches some frequently used web pages into the local cache of his mobile unit (MU) and then disconnects MU from the web server to save its battery energy and the wireless communication cost. During disconnection, the user uses the Web pages cached in MU to access the Web and the write operations to these Web pages performed by the user are recorded into a log. When the MU reconnects to networks again, the log is sent to the Web server for reintegration and the conflicts with updates performed by other users need to be resolved. And these three phases are termed as hoarding, disconnection, and reintegration.

Until now, various mechanisms proposed in [2], [4], [5], [6], [7] support disconnected Web page access in wireless environments in the above three phases. However, none of these works have concerned on how to keep the wireless Web page access process fault-tolerance, when the MU suffers a handoff or even a failure without losing its efficiency. This article presents a new scheme named MAWA based on mobile agents to cope with the failure scenario, which uses mobile agents to facilitate seamless logging of access activities for recovery from failure.

The reminder of this paper is organized as follows. Section 2 overviews the related works. Section 3 presents the system framework. Section 4 discusses the update propagation algorithms. Performance analysis of the algorithms is given in section 5. Finally, Section 6 concludes the paper.

2 Related Works

Until now, most existing schemes for wireless Web page access just support read-only Web page access during MU's disconnection from Internet. However, with the rapidly

increasing of distributed web authoring and form-based electronic commerce web applications [7], supports to write operations are also needed.

Work reported in [11] proposes the concepts of lock and version. Each PUT request indicates the original version of the web page from which its new revision is derived. If the web page has already been updated by other applications, the update request of the client is denied. The client can then fetch the new version of the page to resolve conflicts, and lock the web page to ensure there is no update for the page from other clients at the same time. The client can lock multiple web pages on the same web server to ensure an atomic update be done on these web pages.

In [12], the Caubweb system designs a HTTP client proxy running on MU to cache staging updates during the disconnection phase. When MU reconnects with Internet again, the PUT requests from the HTTP client proxy are accepted by a PUT script running on the HTTP server. MU can update the page in its local cache by keeping two versions of it: the original version of it and the up-to-date version modified.

In [5], the system uses a cache manager named Venus on the client-side. During disconnection, all updates to the Web pages made by the client are recorded into an operation log. Upon reintegration, the Venus resynchronizes its local cache with the server. If Venus detects a divergence, an application specific resolver (ASR) is invoked to resolve the difference. If the ASR fails to resolve the difference, then a manual repair tool running on the client side is invoked.

In [14], the authors integrate the concept of coherency interval of supporting disconnected Web browsing proposed in [9] with the concept of versioning and locking as proposed in [11] to support disconnected write operations for wireless Web access, and presents three update propagation algorithms. The goal of it is to identify the length of the disconnected period so that to minimize the total communication costs during the reintegration phase.

However, none of these works concern on how to cope with the scenario when the MU suffers a handoff or even a failure. And in these works, during the reintegration phase, MU has to communicate with Web server for many times, thus, improving the burden of the Web server. We present a new Web page access mechanism based on mobile agents to resolve these problems. It uses mobile agent between the Web server and MU to support fault-tolerance wireless Web access, and to reduce the overall communication cost.

3 System Framework

A mobile computing system (see fig.1) is composed of a static network and a dynamic wireless network. The static network comprises the fixed hosts and the communication network. Some of the fixed hosts, namely, base stations, are augmented with a wireless network and they provide gateways for communication between the wireless and static network. A mobile unit can connect with the static network with the support of base station. When the mobile unit moves from one physical cell to another, the base station responsible for it will be changed. This process called handoff is transparent to the mobile unit.

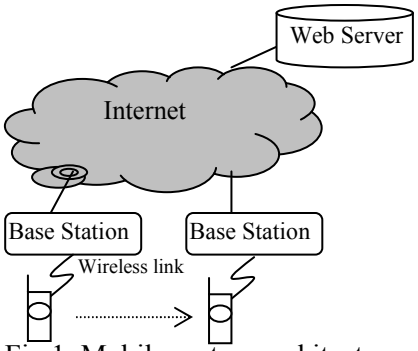

Fig.1. Mobile system architecture

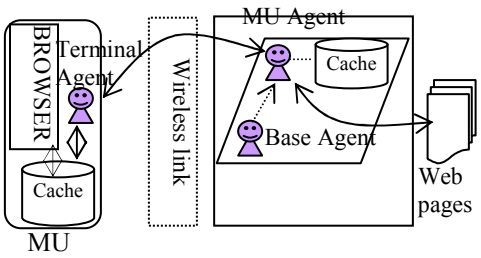

Fig.2. System framework of MAWA

There are three kinds of agents in MAWA as follows (see figure 2):

Terminal agent (TeAg). This static agent runs in the mobile unit. Requests for Web pages from browsers performed by users are transferred to this agent. It also maintains a local cache to store the cached pages and the log of updates to Web pages. Upon reconnection, TeAg communicates with MuAg to propagate the updates.

Base station agent (BsAg). This static agent resides on base station with responsibility for creating, storing, sending and receiving an instance of MuAg according to the request from MU or other BsAg. It maintains the local MuAg queue, and transfers the message or data between MU and its MuAg respectively.

MU agent (MuAg). Created by BsAg, this agent is a mobile agent. It is responsible for a certain MU. In MAWA, MU doesn't communicate with Web server directly. All the data of Web pages and messages sent from one part are first transmitted to MuAg, and it sends them to the other. MuAg makes a log of these data of Web pages and messages in a cache on the base station. This cache also plays an important role in our scheme. When MuAg gets some new version of Web pages from Web server based on the requests from MU, it computes the difference between the fresh content and the cache content, and then sends the difference to the MU. When MU suffers a failure, MuAg uses the cache to help it recover. When MU moves to another base station, the correlated MuAg and cache are also moved to that base station under the help of BsAg.

There are also three phases in MAWA scheme as hoarding, disconnection and reintegration. In hoarding phase, MU prefetches some Web pages into its local cache. In disconnection phase, all updates to Web pages performed by MU are recorded into a log stored in the local cache. In reintegration phase, TeAg propagates the log of the updates to MuAg. And MuAg propagates it to Web server. The detailed MAWA scheme is presented in section 4.

4. Reliable Web Access Scheme

In our scheme, we integrate the concept of versioning and locking proposed in [11], and MU disconnects from network for a coherency interval as proposed in [9]. To analyze the fault-tolerance scheme quantitatively, we need to define the following parameters. For a Web page *i* cached by MU, the update rate done to it by MU is denoted as λ_i^m and the update rate done by all other users is denoted as λ_i^o . Thus, for read-only cached Web pages, its λ_i =0. There are two general cost parameters, C_m and C_w . C_m is the average one-way communication cost of transmitting a simple message over the wired network; C_w is the average one-way

communication cost of transmitting a data packet carrying a Web page over the wired network. α is denoted as the ratio of the bandwidth of wired network to the bandwidth of wireless network. Then the average cost of transmitting a message from base station to MU over wireless link is denoted as αC_m, and the average cost of transmitting a data packet over wireless network is αC_w. C_{rm} is the cost of resolving Web page conflicts performed by MU. When a MuAg migrates from one base station to another, the cost is denoted as C_{Ag}. The rate of MU suffering a handoff is denoted as λ_h. The failure rate of MU is denoted as λ_f. We suppose that both handoff and failure arrive at the system as an exponential distribution.

4.1 Three Kinds of Web Page Update Propagation

Suppose MU is in disconnection phase with the length of disconnection period L_x. And after the period Lx, MU reconnects to network to perform update propagation. Thus, at the time of $L_x - n\alpha C_w$ (here, n is the number of Web pages cached by MU), MuAg inquires Web server about the version state information of the Web pages which MU has prefetched before disconnection and applies to Web server for locking all these Web pages[1]. If such a Web page has already been updated by other users, Web server sends the new version Web page to MuAg. In reintegration phase, MU connects to network again and sends base station an inquiry message[2] indicating which cached pages it have already updated in disconnection phase. After receiving the message, BsAg in the base station searches local MuAg queue to transfer the message to the corresponding MuAg. Then there are several possible cases as follows (here we suppose MU has prefetched only one Web page i):

1. If the Web page i has been updated by MU (see fig.3), and not been updated by other users, then MuAg sends MU a message informing MU to propagate the modified Web page to Web server. MU propagates the Web page to MuAg. MuAg puts it to Web server and releases the lock. The overall communication cost is $3\alpha C_m + \alpha C_w$.
2. Shown in fig.4, for the Web page i cached by MU, if some other user has already updated it when MU is in disconnection phase, then MuAg sends the data of the new version Web page to MU. Based on the new version received, MU applies a merge algorithm to resolve the update conflict and propagates the data carrying the differences of the updated Web page to MuAg. MuAg puts the updated Web page to Web server and releases the lock. The overall communication cost is $2\alpha C_m + 2\alpha C_w + C_{rm}$.
3. For the Web pages i cached by MU, if there are no updates to it during disconnection phase, MU must perform a *forced update* on the page (because of the real-time requirements of online Web applications) when it reconnects with network again (see figure 5). Thus, after receiving a reply message indicating *forced update* from MuAg, MU performs forced updates (the cost is the same as C_{rm}) and propagates the updates to MuAg. MuAg propagates the updated Web page sent by MU to Web server, and releases the lock. So the overall communication cost in this case is $3\alpha C_m + \alpha C_w + C_{rm}$.

[1] We will see late that all the Web pages cached by MU will be updated by it. So, MuAg has to lock them all to ensure that there are no updates from other users when MU is updating them.
[2] For a certain Web page cached, MU sets in the inquiry message the identifier "U" meaning that this page has not been updated, "A" meaning that this page has been updated; MuAg sets in the message sent to MU the identifier "P" meaning that the updates to this page can be propagated, "F" meaning that this page needs force update, "R" meaning that this page needs to resolve conflicts,

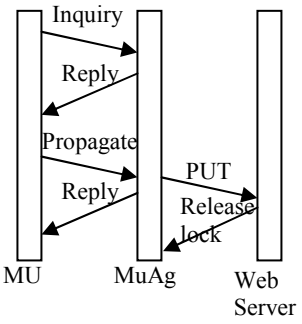

Fig.3. Case 1

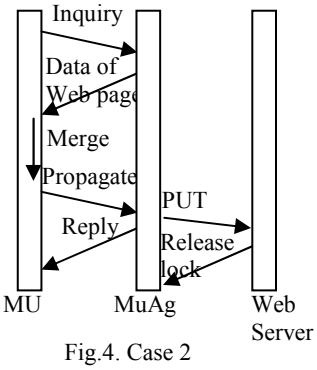

Fig.4. Case 2

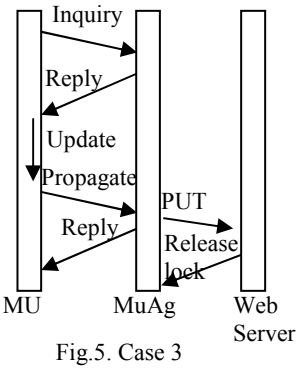

Fig.5. Case 3

Supposing that updates to Web page *i* arrive at the system as an exponential distribution and the length of disconnection period is *Lx*, then *Pi* , the probability of updates to page *i* performed by other users during MU's disconnection phase L_x, is given as

$$p_i = 1 - e^{-\lambda_i^o (L_x - mnC_w)} \tag{1}$$

Also, q_i, the probability that Web page *i* has been updated by MU during the disconnection period of L_x, is as follows

$$q_i = 1 - e^{-\lambda_i^m L_x} \tag{2}$$

4.2 Normal Web Page Update Propagation

Suppose that MU prefetches a set of Web pages in local cache (the number is *n*) in hoarding phase and updates them during disconnection period. Upon reconnection, MU propagates the modified Web pages to Web server. The algorithm is shown below (see fig.6):

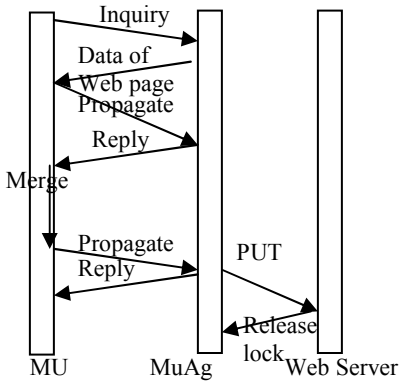

Fig.6. Multiple Web pages update propagation

1 MU sends an inquiry message to base station indicating which pages it has already updated.

2 After receiving the message, MuAg sends a data packet carrying new version of the pages updated by other users to MU. The communication cost is $\sum_{i=1}^{n} p_i \alpha C_w$.

3 When MU receives the data packet sent by MuAg, it knows immediately which pages can be accepted by Web server. MU propagates them to MuAg. The cost is $\sum_{i=1}^{n} q_i (1 - p_i) \alpha C_w$.

4 Depending on the data packet received, MU performs forced updates for the pages that could have been rejected by Web server, including: 1) for pages updated by MU and also by other users, the cost is $\sum_{i=1}^{n} q_i p_i (\alpha C_w + C_{rm})$; 2) for pages not updated by MU but updated by other users, the cost is $\sum_{i=1}^{n} (1-q_i) p_i (\alpha C_w + C_{rm})$; 3) for pages not updated by MU and also not by other users, the cost is $\sum_{i=1}^{n} (1-q_i)(1-p_i)(\alpha C_w + C_{rm})$. Then, MU propagates the updated Web pages to MuAg.

5 MuAg receives the updated Web pages and sends an ACK message to MU.

6 MuAg puts the updated Web pages to Web server and release the locks.

Thus, the overall communication cost of multiple Web page update propagating is as follows:

$$C_s = 3\alpha C_m + \sum_{i=1}^{n} p_i \alpha C_w + \sum_{i=1}^{n} q_i (1 - p_i) \alpha C_w + \sum_{i=1}^{n} (1-q_i)(1-p_i)(\alpha C_w + C_{rm}) + \sum_{i=1}^{n} q_i p_i (\alpha C_w + C_{rm}) + \sum_{i=1}^{n} (1-q_i) p_i (\alpha C_w + C_{rm})$$

The minimized value of C_{s1} is obtained as a solution of

$$\frac{\partial C_s}{\partial L_x} = 0 \quad \text{and} \quad \frac{\partial^2 C_s}{(\partial L_x)^2} > 0$$

4.2 Web Page Update Propagation under Handoff

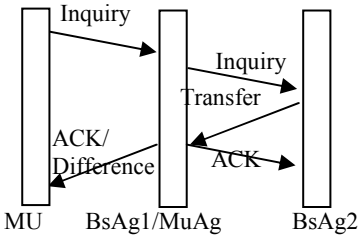

Fig.7 Algorithm for case 1

There are two situations of MU suffering a handoff: a) Suffering a handoff in disconnection phase; b) Suffering a handoff in reintegration phase. As handoff is transparent to MU, when MU moves to another base station, it isn't aware of this. An algorithm proposed in MAWA is shown below to resolve the first case:

1 After the disconnected period, MU sends an inquiry message to the base station. BsAg1 on this base station searches local MuAg queue to find the correlated MuAg and transfers the message to it.

2 If BsAg1 can't find the correlated MuAg, it knows that MU suffers a handoff. With the help of visitor location register (VLR) and home location register (HLR)(the cost is denoted as C_f), BsAg1 gets the information of the previous base station (with BsAg2 on it) the MU had previously connected with.

3 BsAg1 sends an inquiry message to BsAg2.

4 BsAg2 receives the message and searches local MuAg queue to find the correlated MuAg. BsAg2 sends the MuAg with its log to BsAg1.

5 Receiving the instance of MuAg and its log data, BsAg1 adds the MuAg into its local queue and transfers the message sent by MU to it.

6 Then MuAg communicates with MU to help it finish propagating updates, and the following process is the same as those shown in section 4.1.

Summarizing 1-6, when MU suffers a handoff in disconnection phase (supposing the number of pages is *n*), the average cost is

$$C_{s2} = C_s(1-P_h) + (C_s + C_f + C_m + nC_w + C_{Ag})P_h$$

Here P_h, the probability of MU suffering a handoff in disconnected phase, is the value $1-e^{-\lambda_h L_x}$.

For the second case that MU suffers a handoff in reintegration phase, the algorithm dealing with it follows as:

1 a) If MU is propagating updated web pages or sending a message to base station at that time, then the data or message is received by BsAg1 on the new base station.

b) If MU is waiting to receive the message or data of the Web pages sent by MuAg at that time, then since it moves to a new base station MU can't receive them. After an average waiting time C_{wait} ($\approx \alpha C_w$), MU sends an inquiry message to the base station.

2 Once receiving the data packet or message from a MU, BsAg on the new base station searches local MuAg queue to find the MuAg responsible for the MU. Then the following process is the same as the scheme presented above to deal with the situation that MU suffers a handoff in disconnected phase.

In the second case, the average cost of MU propagating *n* updated Web pages is

$$C_{s3} = C_s(1-P_h) + (C_s + \alpha C_m + C_{wait} + C_f + C_m + C_{Ag} + nC_w)P_h$$

Here P_h, the probability of MU suffering a handoff in reintegration phase, is as follows

$$P_h\{L_x < X < L_x + C_s \mid X > L_x\} = \int_{L_x}^{L_x+C_s} \lambda_h e^{-\lambda_h t} dt / (1-F(L_x)) = 1 - e^{-\lambda_h C_s}$$

4.3 Web Page Update Propagation under Failure

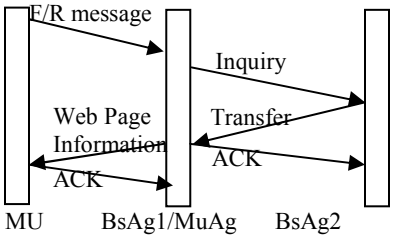

Fig.8 Wireless Web access when MU suffers a failure

For the brittleness of MU's hardware, MU suffers a failure occasionally. There exist three situations of MU suffering a failure: a) MU fails in disconnection phase; b) MU fails in reintegration phase; c) MU suffers a handoff and a failure simultaneously.

An algorithm given in MAWA to dealing with these three situations:

1 After a failure, MU reconnects to network and sends a message indicating "failure/ recovery" to the base station since it is aware of the failure of itself.

2 BsAg on the base station receives the message and searches local MuAg queue to find the MuAg responsible for the MU.

a) If such a MuAg can be found, BsAg transfers the message to it.

b) If such a MuAg can not be found, BsAg knows that MU has moved from some other base station here. With the help of VLR and HLR, BsAg finds the

previous base station MU had connected with and sends a message to it. The BsAg' on that base station receives this message and finds out the MuAg. BsAg' sends the MuAg and its log to BsAg. BsAg adds the MuAg into its local MuAg queue and transfers the message indicating " failure/recovery" sent by MU to it.

3 MuAg receives the message and checks its log:
   a) If MU has not propagated any updated Web pages, MuAg sends MU a message indicating which pages MU hasn't propagated the updates and the data of the up-to-date versions of all the Web pages that MU had prefetched.
   b) If MU has already propagated some updated Web pages, MuAg sends MU a message indicating which pages MU hasn't propagated updates with the updated pages that MU has already propagated and the up-to-date version of the Web pages that need forced updating.

4 MU receives them
   a) If the failure occurs when MU is in disconnected phase, MU will be in disconnection phase again.
   b) If the failure occurs when MU is in reintegration phase, MU proceeds with its update propagation as presented in section 4.1.

Since the failures occurring in disconnected phase doesn't influence the overall cost of update propagation in reintegration phase, we only take the case that failure occurs in reintegration phase into accounts. Summarizing 1-4 above, the average cost of propagating *n* updated Web pages when MU suffers a failure in reintegration phase is:

$$C_{s4} = (1-P_f)C_{s3} + P_f((1-P_h)(C_{s2} + \alpha C_m + n\alpha C_w) + P_h(C_{s2} + \alpha C_m + n\alpha C_w + C_m + nC_w + C_f + C_{Ag}))$$

Here P_f , the probability that MU suffers a failure in reintegration phase, is as follows

$$P_f\{L_x < X < L_x + C_s \mid X > L_x\} = \int_{L_x}^{L_x+C_s} \lambda_f e^{-\lambda_f t} dt / (1-F(L_x)) = 1 - e^{-\lambda_f C_s}$$

5 Performance Study

5.1 Simulation Parameters

We use a set of IBM NetVistas to simulate several MUs, and use a set of Dell PowerEdge 1400SC servers to run as a Web server and several base stations. The servers are connected with 100Mb/s fast Ethernet. To compare MAWA with MPUPA (multiple-page update propagation algorithm) that are proposed in [14], this section simulates the effect of different parameters (λ_i^o , λ_i^m , C_{rm} , C_w) on them and the results are shown in section 5.2. Table 1 lists the values of input parameters below.

Table 1 Parameters Input

Parameters	Value
Failure rate λ_f	(0.005, 0.2)
Handoff rate λ_h	(0.003125,0.2)
Wireless network factor α	10
Update rate for web page i λ_i $(=\lambda_i^o + \lambda_i^m)$	(0,15)updates/hour
Average cost of transferring a message over wired network C_m	0.01
Average cost of transferring a web page over wired network C_w	(0.5, 5)

Average cost of resolving update conflicts C_{rm}	(30, 180)
Average cost of transferring a MuAg C_{Ag}	0.1
Bandwidth of a wireless channel	9.6Kb/s
Wired link factor for Intra-MSC message transfer	2
Wired link factor for Inter-MSC message transfer	3

5.2 Simulation Results and Analysis

The multiple-page update propagation algorithms MPUPA and MAWA are compared in fig.9-17, under the effect of the different parameters λ_i^m , C_{rm} and C_w . In this case MU are assumed to updates ten Web pages each time and the updated rate λ_i of each Web page i is selected in the range of [0,15] randomly. In figure 9-11, λ_i^m / λ_i varies in the set of {0.2, 0.6, 1}. In figure 12-14, C_{rm} varies in the set of { 30, 90, 150}. While in figure 15-17, C_w varies in the set of {0.5, 2.5, 5}. Observed from the figures, MAWA is distinctly more effective than MPUPA. The reason is that, in MAWA, MU first sends MuAg a message to inquire the state of the cached Web pages and then decides to take the following actions. In this way, much wireless communication cost on unnecessary page update propagation is avoided. And as a result, it saves the wireless communication cost greatly. On the other hand, all the data of Web pages in MAWA is sent to MU by the MuAg not the Web server, and the communication cost on wired network is also saved in this way.

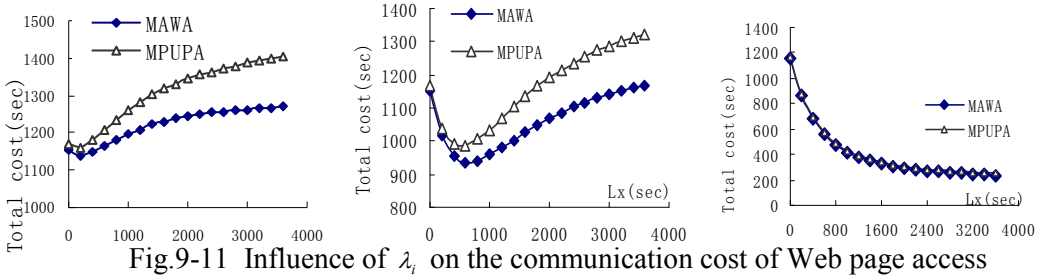

Fig.9-11 Influence of λ_i on the communication cost of Web page access

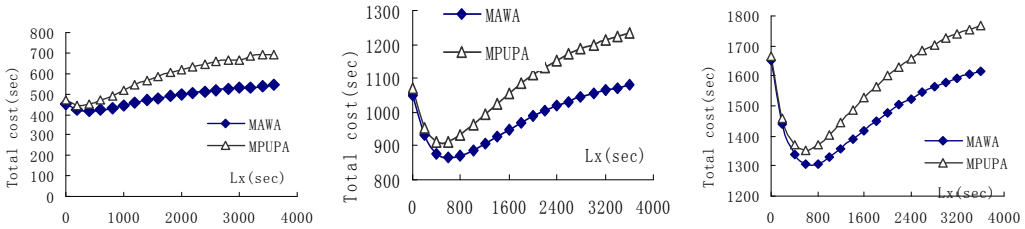

Fig.12-14 Influence of C_{rm} on the communication cost of Web page access

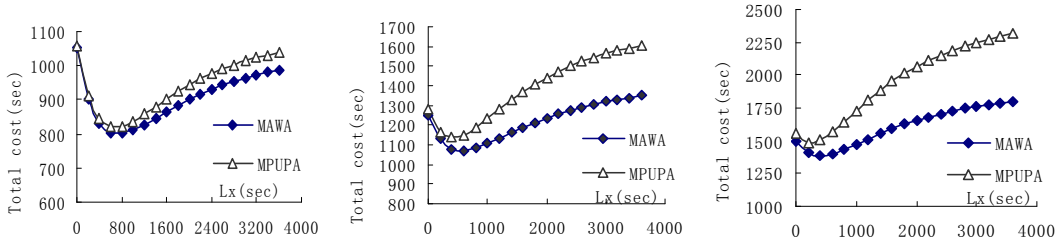

Fig.15-17 Influence of C_w on the communication cost of Web page access

Figs.18-23 shows the influence of handoff rate λ_h and failure rate λ_f on the recovery cost of MU in MAWA. We fix C_w =1.5 second, C_{rm} =100 seconds, C_m =0.01 second, λ_i =10

updates/hour, λ_i^m / λ_i varies in the set of {0.2, 0.6, 1}, λ_h varies in the set of {0.003125, 0.00625, 0.0125, 0.025, 0.05, 0.1, 0.2}, and λ_f varies in the set of {0.005, 0.02, 0.1}. Observed from the figure 18-20, both for single-page update propagation (in this scenario MU prefectches and update one single Web page each time) and for multiple-page update propagation (in this scenario, MU prefetches and update ten Web pages and the λ_i of each Web page is selected randomly), the recovery cost increases with λ_f increased and this means that MU suffers more failure during a fixed period. For a fixed λ_f, the recovery cost also increases with λ_h increased and this means that when MU suffers a failure, the probability of MU suffering a handoff simultaneously increases too. Noticed that in the case of single-page update propagation, when λ_i^m / λ_i =1, all the updates to Web page are only done by MU, and all the updates propagated by MU will be accepted by Web server. So MU can prolong L_x freely without worrying about performing forced updates. And the value of C_s is not large so the probability of suffering a handoff and a failure is distinctly increased with the increased value of λ_h and λ_f. However in the case of multiple-page update propagation, though the value of λ_f and λ_h vary in a large range, both $P_f \approx 1.0$ and $P_h \approx 1.0$ for a large value of C_s. So, though under different values of λ_f and λ_h, the recovery cost changes little.

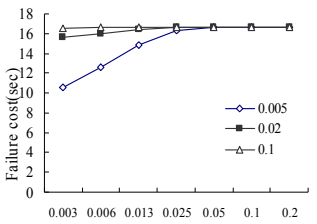

Fig.18. λ_i^m / λ_i =0.2

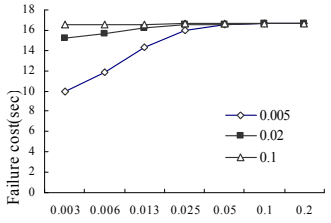

Fig.19. λ_i^m / λ_i =0.6

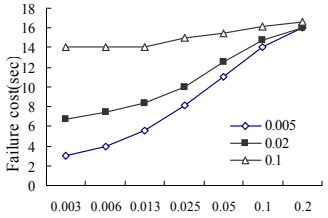

Fig.20. λ_i^m / λ_i =1

Fig.18-20. Effect of λ_f and λ_h on the recovery cost of single-page update propagation

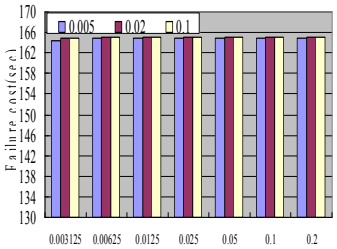

Fig.21. λ_i^m / λ_i =0.2

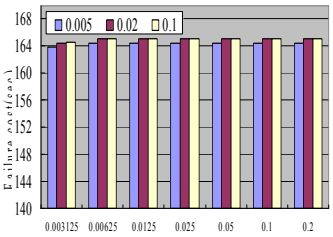

Fig.22. λ_i^m / λ_i =0.6

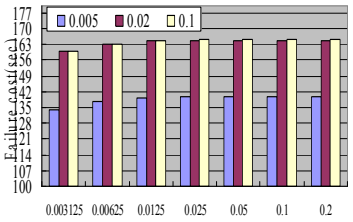

Fig.23. λ_i^m / λ_i =1

Fig.21-23. Effect of λ_f and λ_h on the recovery cost of multiple-page update propagation

6 Conclusion

A reliable and effective scheme called MAWA is presented in this paper for MU to perform Web page update propagation in wireless mobile environments. This scheme uses mobile agents between Web server and MU to avoid the unnecessary update propagation performed by MU, so that MU can save its communication cost. The mobile agent MuAg also helps MU to perform Web page updates reliably, when it suffers a handoff or even a failure. And the simulation results show the behavior of this mechanism.

References

[1] Chander Dhawan. Mobile Computing: A systems Integrator's Handbook. McGraw-Hill, USA.

[2] J. Jing, A.S. Helal. Client-Server Computing in Mobile Environments. ACM Computing survey, 1999,31(2): 117-157.
[3] E. Pitoura and G. Samaras. Data Management for Mobile Computing. Kluwer Academic Publishers, 1998.
[4] H. Chang et al. Web Browsing in a Wireless Environment:Disconnected and Asynchronous Operation in ARTour Web Express. Proc. Third ACM/IEEE Conf. Mobile Computing and Networking (MobiCom '97), 260-269. 1997
[5] J.J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. ACM Transactions on Computer Systems, 1992,10(1): 3-25.
[6] Z. Jiang and L. Kleinrock. Web Prefetching in a Mobile Environment. IEEE Personal Communications. 1998, 5(5): 25-34.
[7] M.S. Mazer and C.L. Brooks. Writing the Web while Disconnected. IEEE Personal Communications. 1998, 5(5): 35-41.
[8] A. Joshi, S. Weerawarana, E. Houstis. On Disconnected Browsing of Distributed Information. Proceeding of the 7th IEEE Workshop on Research Issues in Data Engineering RIDE, 101-107,1997.
[9] R. Floyd, R. Housel and C. Tait. Mobile web access using eNetwork Web Express. IEEE Personal Communications, 1998, 5(5): 47-52.
[10] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, K. Raatikainen. OptimizingWorld Wide Web for weakly connected mobile workstations: An indirect approach. Proceeing of the 2nd International Workshop on Services in Distributed and Networked Environments, 153-161,1995.
[11] E.J. Whitehead Jr and M.Wiggins. WEBDAV: IEIF Standard for Collaborative Authoring on the Web. IEEE Internet Computing, 1998, 2(5): 34-40.
[12] M.S. Mazer and C.L. Brooks. Writing the web while disconnected. IEEE Personal Communications, 1998,5(5):35-41.
[13] M.F. Kaashoek, T. Pinckney, and J.A. Tauber. Dynamic documents: mobile wireless access to the WWW. IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, Dec. 179-184, 1994.
[14] I.Chen, N.A.Phan, I.L.Yen. Algorithms for Supporting Disconnected Write Operations for wireless Web Access in Mobile Client-Server Environments. IEEE Transactions on mobile computing, 2002, 1(1):48-58.
[15] D.VanderMeer, A.Datata, K.Dutta. Mobile User Recovery in the Context of Internet Transactions. IEEE Transactions on Mobile Computing, 2003,2(2):132-146.
[16] T.Park, N.Woo, H.Y.Yeon. An Efficient Recovery Scheme for Mobile Computing Environments. Proceedings of the Eighth International Conference on Parallel and Distributed Systems. 2001.
[17] C.P.Martin, K.Ramamritham. Support for recovery in mobile Systems. IEEE Transactions on Computers. 2002, 51(10):1219-1224.

Authors

HaiYang Hu received the B.E. and M.E. degrees, from NanJing Univ. in 2000 and 2003, respectively. He received the Dr. Eng. degree from NanJing Univ. in 2006. He has been working as an assistant professor in the Dept. of Computer and Information Engineering, ZheJiang Gongsang Univ., from 2006. His research interest includes Object-oriented technology, Network computing.

Hua Hu received the Dr. Eng. degree from Zhejiang Univ. in 1997. He has been working as an full professor in the Dept. of Computer and Information Engineering, ZheJiang Gongsang Univ. His research interest includes Object-oriented technology, Software Engineering and Network computing.