A Reliable and Configurable E-commerce Mechanism based on Mobile Agents in Mobile Wireless Environments*

Haiyang Hu, Hua Hu, Jie Chen
College of Computer and Information Engineering, Zhejiang Gongshang University
hhy@mail.zjgsu.edu.cn

Abstract

Because of the limited capability of mobile devices and lower bandwidth of wireless network, Web-based e-commerce in mobile wireless environments suffers more challenges and it needs an efficient and reliable mechanism to support the process. This paper presents a flexible and configurable mechanism, which uses mobile agents as brokers of mobile device to communicate with Web server, thus to reduce the unnecessary wireless communication costs of mobile client. It can also facilitate mobile device recovery back when suffering a failure to increase the reliability of the whole application process. Compared with other mechanisms through performance analysis, our mechanism shows its efficiency and reliability.

1. Introduction

Using mobile devices to communicate with each other has become a normal life in modern society [1]. People can use the mobile devices to do different commercial applications without caring about their physical locations. However, because of mobile device's limited hardware capability, and the unstable and expensive wireless links, doing e-commerce with mobile devices in mobile wireless environments faces many technology challenges. Thus, the users don't want to connect with the Web server through wireless links continuously for a long time and they need a flexible mechanism that can help them to complete the application process.

This paper presents a new mechanism named MAEC based on mobile agents to solve the above problems. In this mechanism, mobile agent is used as a broker between the server and the mobile client to support a flexible and configurable e-commerce application, and to minimize the overall communication cost. The rest of this paper is organized as follows: Section 2 presents the system model. Section 3 discusses the mobile agent-based mechanism for Web-based e-commerce, which includes the assembly configuration for mobile agents and their protocols. Performance analysis of this mechanism is given in section 4. We present the simulation in section 5. Relation works and conclusion of this paper are given in section 6.

2. System Model

When performing e-commerce application in mobile wireless environments, the user usually[8][9] prefetches some frequently used information from the Web server into the local cache of his mobile device, then disconnects from the web server to save the device's battery energy and the wireless communication cost. During disconnection, the user uses the

* This paper is supported by the National Natural Science Foundation of China granted by Nos. 60673179, the Natural Science Foundation of Zhejiang province granted by Nos. Z106727, and the Science and Technology Foundation of Zhejiang province granted by Nos. 2006C21021.

information cached in the mobile devices to access the Web server; the operations that will be performed to the Web server are recorded into a log. When the mobile device reconnects to the networks again, the log is sent to the Web server for reintegration. These three phases are termed as hoarding, disconnection, and reintegration. However, if the operations are rejected by the Web server in the reintegration phase, the user has to renew his operations. In this way, the mobile device will connect to the Web servers continuously for a long time. Thus, the burdens of Web server are increased and the user has to pay for the expensive costs of wireless links. We use mobile agents to do the operations replacement of the user himself, and this mechanism has two advantages: first, when mobile agents access the Web server, the user needn't connecting to the wireless network continuously; second, when the user wants to renew his operations, he just sends the configuration profile to the agent and it will adjust to perform the operations according to this profile.

In MAEC, we design two kinds of agents as follows.

UsAg: This static agent resides on base station in the wired network with responsibility for creating, storing, sending and receiving an instance of MoAg according to the request from the mobile client. It maintains a local MoAg queue, and transfers the message or data between the mobile client and its MoAg respectively. It can also help the user reconfigure his corresponding MoAg to perform new operations. UsAg makes a copy of the profiles transmitted by the users in a log on the base station. This log plays an important role in helping the mobile client restarting his e-commerce application, when his mobile device suffers a failure.

MoAg: Created by UsAg, this agent is a mobile agent. It is responsible for a certain mobile client. In MAEC, the mobile client doesn't communicate with Web server directly. All the operations performed by the users are first transmitted to MoAg, and it sends them to Web server. When the mobile device moves to another area, the correlated MoAg also moves to the new base station with the help of UsAg. MoAg's inner structure can be reconfigured by the UsAg to perform new operations.

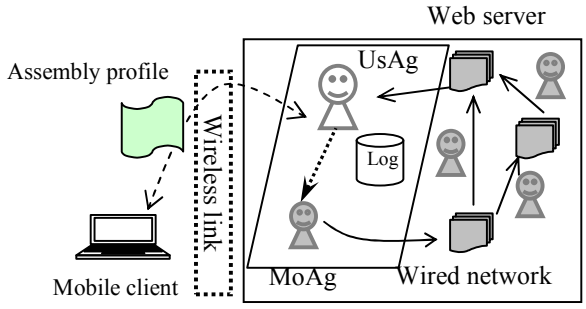

Figure 1. System model

3. Configurable Structure for Assembly

In MAEC, MoAg is created by UsAg in accordance to the assembly profile (Asp) transmitted by the user, which can be defined formally as:

Definition $Asp = <AT, CR, CM, CL, AP>$, here:

- *AT* is the assembly type. There are two structures in our mechanism: primary and nested. The primary structure contains three architecture styles: sequence (SEQ), parallel (PAR), loop. The nested structure is combined by several primary styles.
- *CR* is a set of Web servers that MoAg will access.
- *CM* is a set of operations provided by Web servers that the user wants to perform.
- *CL* is a set of physical locations that Web servers reside on.

- *AP* is a set of private operations provided by the user himself to be performed for filtering and selecting the results of the e-commerce activities.

We present the the styles in BNF:

1. *AT* = SEQ

<*Assembly Type*> ::= < *SEQ* >
< *SEQ* >::= < *Condition* >;<*Primary*>;< *Ref* >;< *MethodCall* >;< *AfterProcessing* >{;< *Ref* >;< *MethodCall* >;< *AfterProcessing* >;}
< *SEQ1* >::=< *Condition* >;<*Nested*>;< *Assembly Type* >
<*MethodCall*> ::= < *InterfaceName* >< *MethodName* >{< *ParametersList* >}&< *MethodType* >
<*MethodType*> ::= <*Mode* > < *Type*>{;<*Mode*> <*Type*>}

2. *AT* = PAR

< *Assembly Type* > ::= < *PAR* >
<*PAR*> ::= <*Condition* >;<*Primary*>;< *Ref* >;< *MethodCall* >{;< *Ref* >;< *MethodCall* >;}
<*PAR1*> ::= <*Condition* >;<*Nested*>;< *Assembly Type* >
<*MethodCall* > ::= <*InterfaceName*><*MethodName*> {<*Para metersList*>}&<*MethodType*>
<*MethodType*> ::= <*Mode*><*Type*>{;<*Mode*><*Type*>}

3. *AT* = SLOOP

<*Assembly Type*> ::= < *SLOOP* >
< *SLOOP* >::= < *Condition* >;<*Primary*>;< *Ref* >;< *MethodCall* >;< *AfterProcessing* >{;< *Ref* >;< *MethodCall* >;< *AfterProcessing* >;}
< *SLOOP1* >::=< *Condition* >;<*Nested*>;< *Assembly Type* >
<*MethodCall*> ::= < *InterfaceName* >< *MethodName* >{< *ParametersList* >}&< *MethodType* >
<*MethodType*> ::= <*Mode* > < *Type*>{;<*Mode*> <*Type*>}

As shown in figure 2, when UsAg receives such an *Asp*, it creates the corresponding MoAg and a facilitator: MoAg manager.

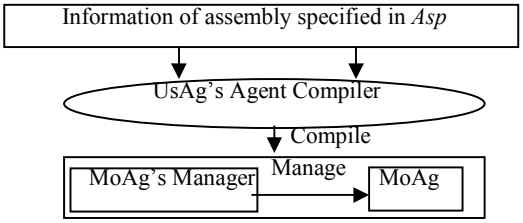

Figure 2. Creating MoAg and its manager

MoAg Manager is a transient entity, which is responsible for managing the MoAg. Corresponding to the type of Asp, there are also different types of MoAg Manager, which incorporates the control flow semantics of the Asp. Figure.3 shows the structure of a manager in "PAR" style. From the figure, it can be seen that the key part in implementing MoAg Manager is the synchronization of agent(s), so that proper communication of data and control transfer can be achieved. Specifically, the run() method is the entry point of MoAg Manager, it runs on the main thread. After proper initialization, the main thread suspends and waits for the agent(s) to return. After coming back, the agent(s) will call the appropriate put(…)

method in MoAg Manager to return the results, at this point, the thread running the agent(s) suspends, waiting for the decision made further by the MoAg Manager.

```
Class PAR-MoAgManager()
{  //Instance variables definitions for parameters;
   //Instance variables definitions for private data;
    public  PAR-MoAgManager(){//constructor}
    public  synchronized  void run(){
      // create the number of MoAgs that need sending out;
      //create the MoAg with Itinerary type ;
      //transfer values of instance variables from MoAg manager to MoAg;
      //start all of the MoAg;
      //waiting for all the MoAg returning back;
    }
    public synchronized void put_1()  {
     // transfer the values of instance variables in MoAg 1 back to MoAg
Manager;}
    ……
    public  synchronized put_n()  {
       // transfer the values of instance variables in MoAg N back to MoAg
       Manager;}
}
```

Figure 3. A PAR-style MoAg Manager

4 The Process of Web Server Access

4.1 Normal Web Server Access

In MAEC, we integrate the concept of versioning and locking proposed in [11], and mobile device disconnects from network for a coherency interval as proposed in [9]. To analyze MAEC quantitatively, we need to define the following parameters. The user with his mobile device in disconnection phase is with the length of disconnection period L_x. For the information of Web server *i* cached by the user, the operation rate performed to it by the user 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 information, 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 the information from Web server over the wired network. We also use it to denote the cost of transmitting an assembly profile 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 mobile device 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 for the user renewing his assembly profile on mobile device. When a MoAg migrates from one node to another in the wired network, the cost is denoted as C_{Ag}. The rate of mobile device suffering a failure is denoted as λ_f. We suppose that failure arrive at the system as an exponential distribution. C_S is the average cost of the user completing his e-commerce application successfully.

Now, suppose that the user has prefetched the information needed from Web servers in the mobile device's local cache in hoarding phase. These information are also stored in UsAg' log. During disconnection period, the operations performed to the information are stored. Before the reconnection phase coming, UsAg detects the Web servers relate to the user's e-

commerce to see whether they have already been updated by other users. If some Web server has been updated, UsAg will download the latest information of the e-commerce into its log. Then, upon reconnection, the user sends an inquiry message to the UsAg and UsAg will do as follows (see the protocol in fig.4):

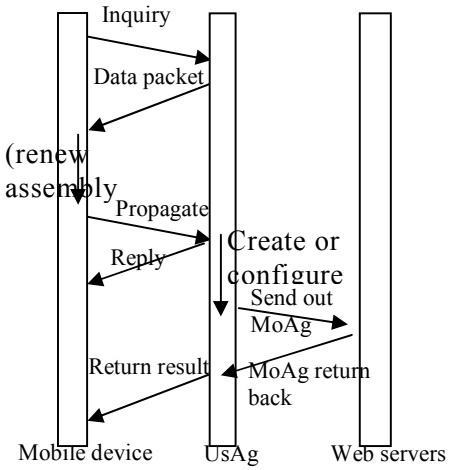

Figure 4. Communication protocol

1 The user sends an inquiry message to UsAg.
2 After receiving the message, UsAg sends a data packet carrying new version of the information downloaded from Web servers to the user.
3 Depending on the data packet received, the user immediately knows that whether he will have to renew his operations. If not, the user sends the assembly profile to UsAg directly. Otherwise, the user has to reconfigure his assembly plan, and sends the renewed assembly profile to UsAg.
4 UsAg receives the profile and sends an ACK message to MU.
5 UsAg creates (or configures) the MoAg and sends it to the Web servers.
6 When MoAg coming back, the UsAg sends the result to the mobile client.

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

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

Assume that the user needs to access *n* Web servers in his e-commerce application, the probability that he needs not renewing his assembly profile in the reintegration phase is:

$$P = \prod_{1 \le i \le n} (1 - p_i) \tag{2}$$

As shown in Fig.4, the total cost that will be consumed in reintegration phase is :

$$C_S = P(2\alpha C_m + 2\alpha C_w + C_{Ag}) + (1-P)(2\alpha C_m + 2\alpha C_w + C_{Ag} + C_{rm}) \tag{3}$$

The minimized value of C_S is obtained as a solution of

$$\frac{\partial C_s}{\partial L_x} = 0 \quad \text{and} \quad \frac{\partial^2 C_s}{(\partial L_x)^2} > 0 \tag{4}$$

4.2 Web Server Access under handoff

There are two situations of mobile client suffering a handoff: a) Suffering a handoff in disconnection phase; b) Suffering a handoff in reintegration phase. As handoff is transparent to mobile client, when mobile client 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, mobile client sends an inquiry message to the UsAg1 on this base station searches local MoAg queue to find the correlated MoAg and transfers the message to it.
2 If UsAg1 can't find the correlated MoAg, it knows that mobile client suffers a handoff. With the help of visitor location register (VLR) and home location register (HLR)(the cost is denoted as C_f), UsAg1 gets the information of the previous base station (with UsAg2 on it) the mobile client had previously connected with.
3 UsAg1 sends an inquiry message to UsAg2.
4 UsAg2 receives the message and searches local MoAg queue to find the correlated MoAg. UsAg2 sends the MoAg with its log to UsAg1.
5 Receiving the instance of MoAg and its log data, UsAg1 adds the MoAg into its local queue and transfers the message sent by mobile client to it.
6 Then MoAg communicates with mobile client to help it finish propagating updates, and the following process is the same as those shown in section 4.1.

Summarizing 1-6, when mobile client 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 mobile client suffering a handoff in disconnected phase, is the value $1 - e^{-\lambda_h L_x}$.

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

1 a) If mobile client is propagating updated web pages or sending a message to base station at that time, then the data or message is received by UsAg1 on the new base station.
b) If mobile client is waiting to receive the message or data of the Web pages sent by MoAg at that time, then since it moves to a new base station mobile client can't receive them. After an average waiting time C_{wait} ($\approx \alpha C_w$), mobile client sends an inquiry message to the base station.
2 Once receiving the data packet or message from a mobile client, UsAg on the new base station searches local MoAg queue to find the MoAg responsible for the mobile client. Then the following process is the same as the scheme presented above to deal with the situation that mobile client suffers a handoff in disconnected phase.

In the second case, the average cost of mobile client 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 mobile client 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 The Structure of MoAg

In our approach, the MoAg is a tuple (*A, I, CT, CS, F*) where: *As* is the *Asp* carried by MoAg; *I* is the identifier of this agent; *CT* are the current services executed by agent; *CS* is current state of agent, its value is "WAITING", "NOTREADY" or "READY"; *F* is the agent's function body.

To gain a high efficient composition, The MoAg will dynamically split and melt to compose the services as parallel as possible, so that the total cost will be minimized. We define agent splitting and melting as follows.

Definition 1 (*Dynamically splitting*). For *Asp* carried by MoAg *A*, $|CR|=k$, there are no such two e-services in *CR* that S_i, $S_j \in CR$, and $DEP(S_i, S_j)$. Then $SPL(A| CR)=\{A_i | 1 \le i \le k\}$, and for each subagent A_i, $A_i.I$ is unique, $A_i.CT= S_i \in CR$, $1 \le i \le k$.

Definition 2 *(Dynamically melting)***.** For MoAg A_i, A_j if $A_i.CT= A_j.CT_j=S_C$, $A_i.CS$="WAITING", $A_j.CS$="NOTREADY", then $A_N = MEL(A_i, A_j|S_C)$, $A_N.I$ is unique, $A_N = A_i \cup A_j$, and $A_N.CT=S_C$. If $S_C.CS$="READY", then $A_N.CT$="READY".

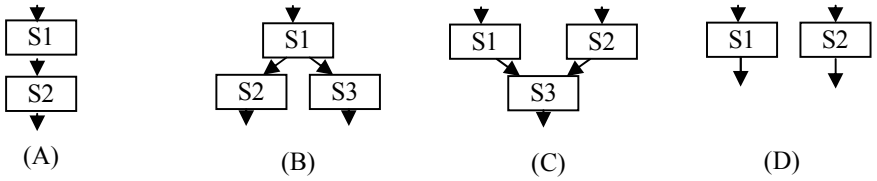

Figure 5. Four structures of e-services composition

In *Asp*, there are different possible kinds of control flow between the e-services as shown in Fig.5. In (A), e-service *S2* only depends on *S1* and there is only one service *S2* depending on *S1*. In this case, coordination agent sequentially executes *S2* after the execution of *S1*. In (B), both *S2* and *S3* depend on *S1*. In this case, after the execution of *S1*, agent *A1* splits itself into two subagents *A12*, *A13*, and then *A12* begins to execute *S2* and *A13* to execute *S3* respectively. In (C), *S3* depends on both *S1* and *S2*. In this case, suppose there are two agents *A1*, *A2* executing services *S1* and *S2* respectively. When agent *A1*and *A2* finish their execution of *S1* and *S2*, they melt into a new agent *A3* with the help of agent server, and *A3* starts to executing service *S3*. In (D), neither *S1* nor *S2* depends directly or indirectly on the other. And in this case, as shown in (A)-(C), there are two different agents in the environment to call them in parallel.

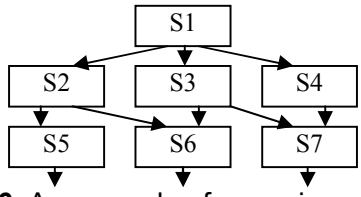

Figure 6. An example of e-service composition

For example, in fig.6, the seven services in such a control structure are represented in the Asp. A coordination agent *A1* parsed by agent server depending on this *Asp* is sent out to fulfill the composition. After calling service *S1*, *A1* splits itself into three subagents *A12*, *A13* and *A14*, with each to call one of the following services *S2*, *S3* and *S4* respectively, so that the

three services are executed in parallel. And after that, agent *A12* splits into *A125* and *A126* to execute *S5* and *S6* in parallel. Agent *A13* splits into *A136* and *A 137* to execute *S6* and *S7* in parallel. As *S6* is parameter dependence on *S2* and *S3*, neither *A126* nor *A136* can fulfill the execution of *S6* alone. So, while calling service *S6*, agent *A126* and *A136* melt together into a new agent *A6* to fulfill the execution. And agent *A137* and *A14* melt into agent *A7* to call the service *S7*. After calling the services *S5*, *S6* and *S7*, the three agents *A125*, *A6* and *A7* proceed with their execution in parallel. From this example, we can see that the services are composed as parallel as possible to save the total cost.

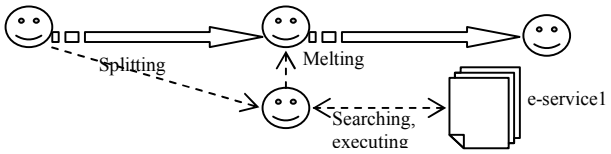

Fig.7. Fault-tolerance searching and executing

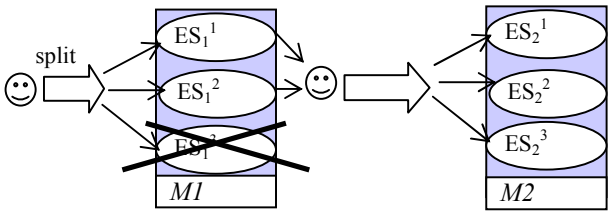

Figure 8. Fault-tolerance searching and executing

In the dynamic environments, often some candidate e-service in composition are shut down and can't be connected, thus, coordination agent must dynamically find some other e-service having the matching behavior in the environment to perform a fault-tolerance composition at runtime. As shown in Fig.7 and 8, to realize a fault-tolerance composition, the agent dynamically splits into multiple subagents to search the environment for the candidate Web services providing the same service method. And during this process, the failure of several subagents won't prevent the coordination agent from proceeding with its composition. As in figure 8, there being three candidate e-ervices (ES_1^1, ES_1^2, ES_1^2) providing the same service method *M1*, coordination agent splits itself into three subagents, each calling a certain e-services. After the calling, these subagents will again melt into a new one to call the following e-services.

5 Performance Study

We use a set of IBM NetVistas to simulate a set of mobile devices, and use a set of Dell PowerEdge 1400SC servers to run as Web servers and base stations. The servers are connected with 100Mb/s fast Ethernet. Table 1 lists the values of input parameters needed. The values of these parameters come from [14], [16] and [20].

Table 1. Input parameters

Input Parameter	Value
Failure rate λ_f	(0.005, 0.2)
Wireless network factor α	10
Update rate for Web server 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 second
Average cost of transferring a web page over wired network C_w	(0.5, 5) second
Average cost of renewing assembly profile C_{rm}	(30, 180)second
Average cost of transferring a MoAg C_{Ag}	0.1 second
Bandwidth of a wireless channel	9.6 K/S

We compare our mechanism with SPUPA and MPUPA that are proposed in [14] and test the effects of different parameters (λ_i^o , λ_i^m , C_{rm} , C_w) on them. To demonstrate the effect of (λ_i^o , λ_i^m) on the two mechanisms, we fix C_{rm} =100 seconds, C_w =1.5 second, λ_i =10 updates/hour, λ_f =0.2, C_m =0.01second. The value of λ_i^m / λ_i varies from 0.2 to 1.0. Figs.9-11 shows the results. Observed from the figures 9 and 10, the minimized cost of update propagation in reintegration phase with the value of L_x is in the range of (200,600). But this requires the mobile device to reconnect to network more frequently. And in this way for every 400 seconds the mobile device has to reconnect to network to do reintegration for about 110 seconds. When the curves slope gently, the mobile client can get a longer disconnection period just for paying a little additional communication cost. For instance as λ_i^m / λ_i = 0.6, when L_x prolongs from 400 seconds to 3200 seconds, the additional cost paid by the mobile client increases only 25 second. So if the mobile client chooses an appropriate L_x during the period when curves slope gently, the overall cost for network connection can be reduced. From Fig.10, the average cost consumed in MAEC decreases about 14.7% than that of SPUPA. However, when λ_i^m / λ_i =1(shown in Fig.11), there is little difference between the two mechanisms. The reason is in this case all the operations to the Web servers are performed by the mobile client only, so they are certainly accepted by Web servers. Thus, the user can prolong L_x freely without worrying about performing forced updates.

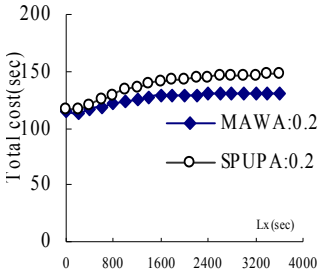

Figure 9. λ_i^m / λ_i =0.3

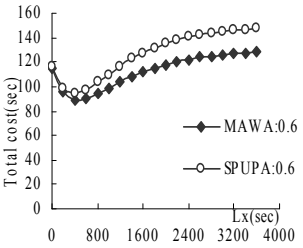

Figure 10. λ_i^m / λ_i =0.6

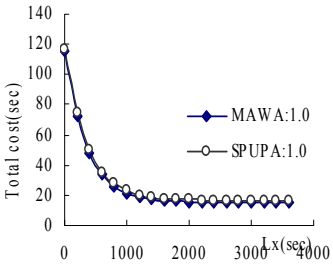

Figure 11. λ_i^m / λ_i =1.0

Figures 9-11 Effect of update rate on two mechanisms

Fig.12-14 shows the effect of parameter C_{rm} on the two mechanisms. We fix λ_i^m / λ_i =0.6, λ_i =10 updates/ hour, C_w =1.5 second, C_m =0.01second, λ_f =0.2 and C_{rm} varies in the range of [30, 150]. Seen from the figures, the curves in two mechanisms both get deeper with C_{rm} increased. This means the user has to pay more cost for renewing his assembly profile. When C_{rm} =30, the average cost of MAEC is about 26.7% lower than that of SPUPA. As C_{rm} increases, the efficiency of MAEC decreases. However, even for C_{rm} increase up to 150, the average cost of MAEC is still 9.4% lower than that of SPUPA. And this is because the mobile client in our scheme first sends UsAg a message to inquire the state of the Web servers. Thus, some unnecessary operation propagation performed by the client is avoided.

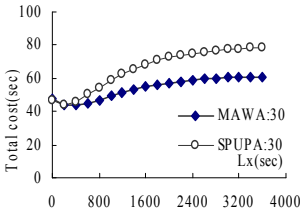

Figure 12. C_{rm}=30

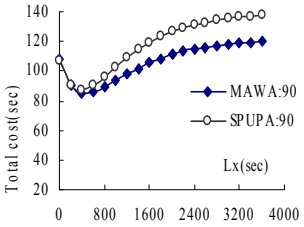

Figure 13. C_{rm}=90

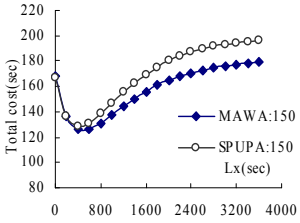

Figure 14. C_{rm}=150

Figure 12-14. Effect of C_{rm} on the two mechanisms

Fig.15-17 show the effect of C_w on the two mechanisms. We fix λ_i^m / λ_i =0.6, λ_i =10 updates/hour, C_{rm} =100 seconds, λ_f =0.2 and C_m =0.01second. The value of C_w varies from 1 to 5 seconds. The communication cost increases when the value of C_w being increased. Indicated by the figures, when C_w =1, the cost of MAEC is only 6% lower than that of SPUPA. However, when C_w increases up to 5, the cost of MAEC is nearly 25% lower than that of SPUPA. We can see that as the cost of transferring the data packet increases, MAEC shows an obvious advantage than SPUPA.

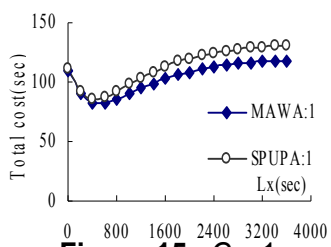

Figure 15. C_w=1

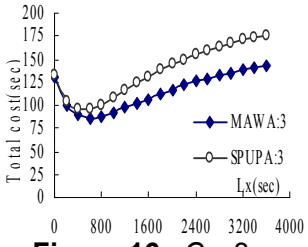

Figure 16. C_w=3

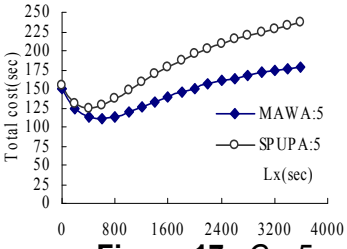

Figure 17. C_w=5

Figure 15-17. Effect of C_w on the two mechanisms

In the e-commerce application of multiple Web servers accessing, the mobile client are assumed to access ten Web servers in each reintegration phase and the updated rate λ_i of each Web server i is selected from the range of [0,15] randomly. We fix C_w=1.5 second, C_{rm}=30 seconds, C_m=0.01 second. Seen from figure 14, showing its obviously efficiency, the cost of MAEC is about 24.6% higher than the cost of MPUPA. The reason is that, in MAEC, the mobile client first sends UsAg a message to inquire the state of the Web servers and then decides to take the following actions. In this way, much wireless communication cost on unnecessary operations propagation is avoided. And as a result, it saves the wireless communication cost greatly. In MAEC, the data packet is sent to the mobile client by UsAg not by Web server, so the communication cost on wired network is also saved in some way.

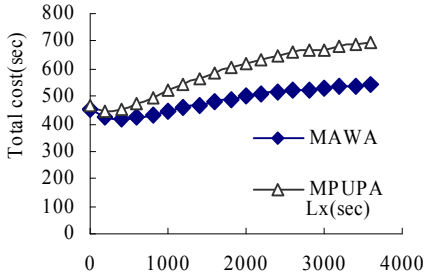

Figure 18. MAEC compared with MPUPA in multiple Web servers access

6 Related works and Conclusion

Until now, most existing schemes for wireless Web access just support read-only Web access during mobile device'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.

In [5], the system uses a cache manager named Venus on the mobile client. During disconnection, all updates to the Web server 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.

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

In [14], the authors integrate the concept of coherency interval of supporting disconnected Web accessing as 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 mobile device suffers a failure. And in these works, during the reintegration phase, the mobile client has to communicate with Web server for many times, thus, improving the burden of the Web server. This paper presents a reliable and effective mechanism for the mobile client to perform e-commerce in wireless mobile environments. This mechanism uses mobile agents as brokers between Web server and the mobile to avoid the unnecessary operations propagation performed by the user, so that the mobile client can save his communication cost. The mobile agent can also help the mobile client to perform e-commerce process seamlessly, when his mobile device suffers a failure. And the simulation results show the behavior of this mechanism.

Reference

1. Chander Dhawan.: Mobile Computing: A systems Integrator's Handbook.McGraw Hill, USA,1998.
2. J. Jing, A.S. Helal.: Client-Server Computing in Mobile Environments. *ACM Computing Survey*, vol. 31, no. 2 June 1999:117-157.
3. E. Pitoura and G. Samaras.: Data Management for Mobile Computing. Kluwer Ac-ademic Publishers. New York,1998.
4. H. Chang.: Web Browsing in a Wireless Environment:Disconnected and Asynch-ronous Operation in ARTour Web Express. *Proc. Third ACM/IEEE Conf. Mobile Computing and Networking*, Sept. 1997:260-269.
5. J.J. Kistler, Satyanarayanan.:Disconnected Operation in the Coda File System *ACM Trans. Computer Systems*, vol. 10, no. 1, Feb. 1992:3-25.
6. Z. Jiang, L. Kleinrock.:Web Prefetching in a Mobile Environment, *IEEE Personal Comm*. vol. 5, no. 5, Oct.1998.:25-34
7. M.S. Mazer, C.L. Brooks.:Writing the Web while Disconnected, *IEEE Personal Comm*. vol. 5, no. 5, pp. 35-41, Oct. 1998.
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*, 1997:101-107.
9. R. Floyd, R. Housel, C. Tait.:Mobile web access using eNetwork Web Express.*IEEE Personal Communications*, Vol. 5, No. 5, Oct. 1998(47-52)
10. M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, K. Raatikainen. Optimizing World Wide Web for weakly connected mobile workstations: An indirect approach. *Proceeing of the 2nd International Workshop on Services in Distributed and Networked Environments*, June 1995:153-161.

11. E.J. Whitehead Jr, M.Wiggins. WEBDAV: IEIF Standard for Collaborative Authoring on the Web. *IEEE Internet Computing*, Vol.2, no.5, 1998:34-40.
12. M.S. Mazer,C.L. Brooks. Writing the web while disconnected. *IEEE Personal Communications*, Vol. 5, No. 5, Oct. 1998:35-41.
13. M.F. Kaashoek, T. Pinckney, J.A. Tauber. Dynamic documents: mobile wireless access to the WWW. *IEEE Workshop on Mobile Computing Systems and applications*,Santa Cruz, CA, Dec. 1994.:179-184.
14. I.R.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*, VoL.1, No.1. 2002:46-58.
15. D VanderMeer, A Datata, K Dutta. Mobile User Recovery in the Context of Internet Transactions. *IEEE Transactions on Mobile Computing*, Vol.2, No.2, 2003:132-146.
16. Taesoon Park, Namyoon Woo, Heon 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 Syst em. *IEEE Transactions on Computers*. Vol.51, No.10, October 2002:pp1219-1224
18. Z.jiang, L.Kleinrock. Web Prefetching in a Mobile Environment. *IEEE Personal Communication*, Vol.5, No.5, Oct.1998:25-34.

Authors

Haiyang Hu born in October, 1977, male, Ph.D, senior membership of China Computer Federation. He received BS, MS and Ph.D. in computer science and technology from Nanjing University in 2000, 2003 and 2006 respectively. His current research interests include mobile agents, software middleware, mobile computing.

Hua Hu born in November, 1964, male, Ph.D, Professor. His current research interests include
autonomous computing, pervasive computing.

Jie Chen born in September, 1965, female, Professor. Her current research interests include
distributed computing, artificial intelligence.