Message Logging and Recovery in Wireless CORBA Using Access Bridge

Xinyu Chen and Michael R. Lyu
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{xychen, lyu}@cse.cuhk.edu.hk

Abstract

The emerging mobile wireless environment poses exciting challenges for distributed fault tolerant (FT) computing. This paper proposes a message logging and recovery protocol on the top of Telecom Wireless CORBA and FT-CORBA architectures. It uses the storage available at the access bridge as the stable storage to log messages and checkpoints on behalf of a mobile host. Our approach engages both quasi-sender-based and receiver-based logging methods and makes seamless handoff in the presence of failures. The details of how to tolerate mobile host disconnection, mobile host crash and access bridge crash are described. The normalized execution time of a mobile host engaging our proposed scheme and the handoff effect are evaluated.

Keywords: *Fault tolerance, Wireless CORBA, Message logging, Failure recovery, Mobile computing*

1. Introduction

Advances in wireless networking technology and portable information appliances have brought a new paradigm of decentralized computing, called mobile computing [3]. Mobile computing enables users to access or exchange information while they roam around, so it causes physical damage of mobile hosts (MHs) more probable [7]. MHs inherit slow processors and small memories. The wireless links usually suffer with high bit error rates, little bandwidths, and long transfer delays. MHs even disconnect from the hosting networks intermittently [13]. Wireless systems are more often subject to environmental conditions which can cause loss of communications or data [5]. All these call for a fault tolerant mobile computing system.

A mobile computing system is considered as an extension of distributed systems. In distributed systems, much of the action takes place at the middleware level. CORBA (Common Object Request Broker Architecture) which is specified by Object Management Group (OMG) is one of the most popular middlewares nowadays. CORBA provides portability, location transparency, and interoperability of applications across heterogeneous platforms (hardware architectures, operating systems, and programming languages) [6]. To support wireless access and terminal mobility in CORBA, OMG also have published Telecom Wireless CORBA specification [8].

Recently OMG have specified Fault Tolerant CORBA [9] as a standard to provide fault tolerance in CORBA. FT-CORBA is based on entity redundancy. It employs three replication styles: *cold passive*, *warm passive* and *active* replications. Logging and checkpointing mechanisms record messages and entity states in logs. All these are intended for wired networks. This paper proposes a message logging and recovery protocol on the top of wireless CORBA and FT-CORBA architectures. The storage available at the access bridge (AB) is employed as the stable storage to log messages and checkpoints on behalf of MHs. Both the quasi-sender-based and the receiver-based logging methods are engaged in our approach . The AB hides mobile host disconnection from other network hosts [15] and makes a seamless handoff when fault tolerant properties are called upon. We also discuss how to tolerate mobile host disconnection, mobile host crash and access bridge crash. After that, a simulation model is constructed to evaluate our proposed scheme.

2. Wireless CORBA Architecture

Figure 1 shows the architecture in Telecom Wireless CORBA [8], which identifies three different domains:

- **Terminal Domain**. The terminal domain is an MH which can move around while maintaining network connections by a wireless interface. It hosts a Terminal Bridge (TB) through which the objects on the MH can communicate with objects in other wired or wireless networks.

- **Visited Domain**. The visited domain contains several ABs to provide communications with objects on MHs.

Proceedings of the Sixth International Symposium on Autonomous Decentralized Systems (ISADS'03)
0-7695-1876-1/03 $17.00 © 2003 IEEE

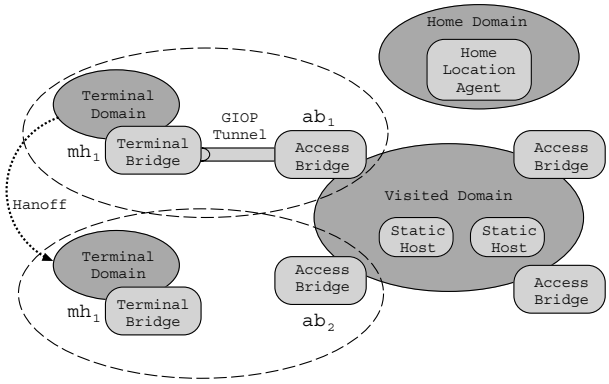

Figure 1. Wireless CORBA architecture

It also contains some Static Hosts (SHs). All communications in the visited domain are via wired links. ABs reside on Mobile Support Stations (MSSs) which have necessary wireless facilities to communicate with MHs and have wired interfaces to communicate with SHs and other MSSs.

- **Home Domain**. The home domain hosts the Home Location Agent (HLA), which keeps track of the ABs that an MH has associated with during its movement.

Each MSS has a geographical area within which it can communicate with MHs directly, plotted as dashed circle in Figure 1. When an MH moves across the border of the geographical area, a handoff occurs between the new AB and the old AB.

All hosts communicate with each other by messages only. The GIOP (General Inter-ORB Protocol) tunnel is the communication channel between an AB and an TB, through which the GTP (GIOP Tunnel Protocol) messages are transmitted. No messages can be exchanged among TBs directly. All messages to and from an MH are relayed by its currently associated AB.

3. Related Work

Neves and Fuchs [7] developed an adaptive checkpointing protocol for mobile environments. It saves consistent global states through local timers and checkpoint number counters piggybacked in application messages. It also piggybacks time to next checkpoint to synchronize those local timers. It logs in-transit messages when taking checkpoints at the sender. Hard checkpoints are saved on stable storage and soft checkpoints are saved locally in the MHs. Instead, our protocol engages uncoordinated checkpointing to reduce message size transmitted through wireless links. No messages will be logged at the MH to decrease the power consumption and storage utilization.

Pradhan et al. [12] described recovery schemes as a combination of state saving strategies and handoff strategies. The state saving strategies include *no logging* and *logging*, the handoff strategies include *pessimistic*, *lazy*, and *trickle* handoffs. It engages "local" base station as the stable storage, which implies that successive checkpoints of an MH may be stored at different base stations when the MH moves around. In our proposal, we also choose the currently connected AB as the stable storage. In the pessimistic strategy, a process checkpoint is transferred to the new base station during handoff. The lazy strategy creates a linked list of base stations visited by the MH but does not transfer checkpoints during handoff. In the trickle strategy, logs and checkpoints are always at a nearby base station, which may be some hops or even one hop further away. In our protocol, we adopt a handoff strategy like the trickle one, but we use an adaptive, dedicated and separate thread to collect the last checkpoint and the successive message logs.

In [15], Yao et al. proposed a proxy-based recovery for applications on MHs. The proxy transparently monitors an MH's interactions with other hosts and maintains a copy of the MH's state. They also described a receiver-based pessimistic message logging protocol in [16].

Park and Yeom developed an asynchronous recovery scheme based on optimistic message logging [10]. This scheme assigns the task of logging to the MSSs. The messages exchanged between the MSSs carry vector clocks for the asynchronous recovery. But traced dependency information may be imprecise that leads to unnecessary rollback of MHs after a failure. We choose pessimistic message logging to avoid large number of control message interchange and rollback propagation.

Ruggaber and Seitz [13, 14] introduced Π^2, a proxy platform for CORBA-based applications in the nomadic environments. It splits the connections between an MH and static server by a proxy to avoid to suffer from sudden disconnections. If an MH detects loss of reply, it will send a retrieval request to retrieve the reply after handoff. In our approach, the AB takes the initiative to forward the reply to the MH after handoff.

4. Fault Tolerance Model

Figure 2 presents an architectural overview of our FT model. Our approach is based on message logging and checkpointing. The message logging mechanism in ABs applies different methods for messages received from and sent to TBs. An AB logs messages after it receives them from an TB (receiver-based), but logs messages which are received from other hosts (ABs or SHs) before it sends them to the TB (quasi-sender-based). The TB may send back an acknowledgement depending on the received message's type. We checkpoint an MH's state periodically by a local

timer or when the amount of the received messages exceeds a predefined threshold.

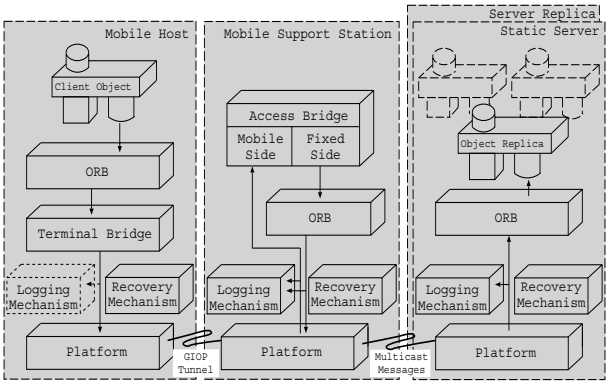

Figure 2. Fault tolerant architecture

An MH may become unavailable due to (i) mobile host disconnection, (ii) mobile host crash, and (iii) access bridge crash. As mentioned before, physical damage becomes more probable to an MH. It is limited to lower processing power, lower memory resources, and lower power supply. It can be disconnected from network intended or unintended. It may often move from one AB to another. So an MH is not suitable to act as stable storage. But when it is disconnected and the user still wants to operate continuously using local information, the FT protocol may save checkpoints in its local disk in order to recover from some transient faults, such as operating system crash and battery discharged [7], etc. So we depict the logging mechanism in MH with dashed lines.

An AB is on the border between wireless and wired network. In Telecom Wireless CORBA architecture, all messages to and from an MH are traversed through ABs. Every message has a local copy in AB. It does not need to send an extra copy of each message elsewhere for logging purpose to tolerate mobile host crash. So we choose the storage at AB as stable storage for the message logging and checkpointing protocol. But the mobile computing environment does not restrict a user's location. When a user moves from one AB to another, the carried MH should change its connected AB. So the location of the stable storage also would be changed during handoff [12]. It is one of the duties of our recovery protocol to find where the last checkpoint is located. An AB contains multiple associated TBs at the same time, but these TBs uses the AB only as a proxy, and there is no dependency between these TBs from the viewpoint of the AB . So an AB keeps different logs for different associated TBs.

The SH and HLA are replicated in *passive* or *active* style according the FT-CORBA standard. An AB communicates with SHs and HLAs by a group communication system, which should detect and suppress duplicate requests and replies, and deliver a single request or reply to the AB. So there is no single point of failures in our architecture. In this paper, we do not discuss this fault tolerance strategy in detail.

4.1. Data Structures

We employ the following data structures in our message logging and checkpointing mechanism.

- *Sequence Number (SN)*. Each message exchanged in GIOP Tunnel has an SN, which identifies the message itself and the order in which the message is sent. An AB ensures the SN is distinct for a dedicated TB, but the SNs may be same between different TBs.

- *Message Record (MR)*. There are two types of MR for two message logging mechanisms. The first type contains a message received by an AB from an TB and the status after the AB processes it. The second type includes an additional SN of the corresponding acknowledgement message, which indicates the order in which the message is received by an MH. The second type is used for messages sent to an TB.

- *CheckpointData* and *CheckpointDataReply* Messages. When an MH takes a checkpoint, it utilizes these two messages to reliably save the checkpoint in the current AB.

- *PurgeCheckpoint* Message. This message is sent out by the HLA to clear old checkpoints when it is informed that a new checkpoint is taken. It needs not be delivered reliably.

- *FetchCheckpoint* and *FetchCheckpointReply* Messages. A *FetchCheckpoint* message is initialized when an MH detects a failure and starts a state recovery procedure. A *FetchCheckpointReply* contains the last checkpoint of the MH.

4.2. Message Logging and Checkpointing

The steps in message logging, illustrated in Figure 3, are (1) A mobile client sends a request message x via a GIOP tunnel to the currently connected AB; (2) The AB logs x in its local stable storage pessimistically, sends an acknowledgement back to the client, and relays x to the remote static server, then the AB waits for a reply; (3) After receiving the reply message y, the AB logs y, and dispatches it to the mobile client; (4) The mobile client sends a message back to acknowledge y and delivers y to the top level; (5) The AB logs the SN in the acknowledgement message with message y.

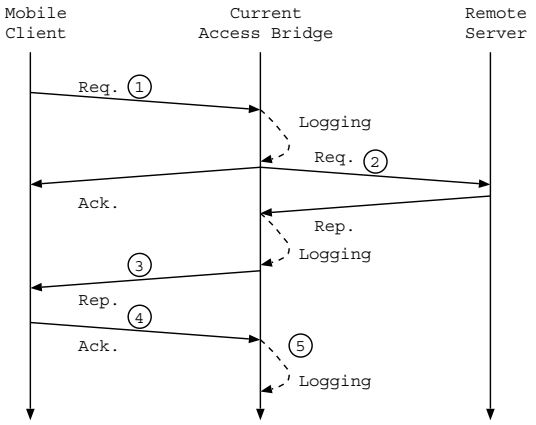

Figure 3. Normal operation sequence

When a local timer in the TB expires or the TB receives a predefined number of messages since its last checkpoint, the TB will initialize a checkpointing procedure. The TB encapsulates the checkpoint in a *CheckpointData* message and sends this message to the current AB. To save wireless bandwidth, the checkpoint may not be sent out immediately, and it can be piggybacked with the next message from TB to AB [10]. The AB logs the checkpoint in its local stable storage and informs the HLA that a new checkpoint of the MH is saved in this AB. This information will be used in the recovery process to fetch the last checkpoint. The checkpointing interval is determined by the application requirements and the failure rate of the MH. It is also determined by the handoff frequency of the MH.

If an MH takes a checkpoint in the currently associated AB, the message logs and checkpoints prior to this checkpoint can be deleted since they are no longer necessary for recovery of this MH. So the AB will send a *PurgeCheckpoint* message to the HLA to delete those obsolete checkpoints and MRs. This message needs not be reliably delivered, so long as any future *PurgeCheckpoint* message for the same MH will be delivered [4]. After the HLA receives the purge message, it will forward this request to the ABs in the itinerary track of the MH so that they can purge the unnecessary checkpoints and messages and collect the stable storage. No MHs and wireless communications are involved during storage collection.

4.3. Mobile Host Handoff

In wireless networks which are organized in cells, a handoff is a mechanism for an MH to seamlessly change a connection from one AB to another. Handoff can be started due to two causes: normal operation and sudden connectivity loss [8]. In normal operation, the MH will create a connection with a new AB. But in the second case, there is another successful outcome of the handoff procedure: connectivity reestablished to the same AB as before. We identify two ABs in a handoff procedure:

- Old Access Bridge (OAB) that was connected by a mobile host before the handoff.
- New Access Bridge (NAB) that would be connected after the handoff, which may be the same as the OAB.

Figure 4 depicts the handoff procedure where an MH reestablish connectivity to a new but different AB.

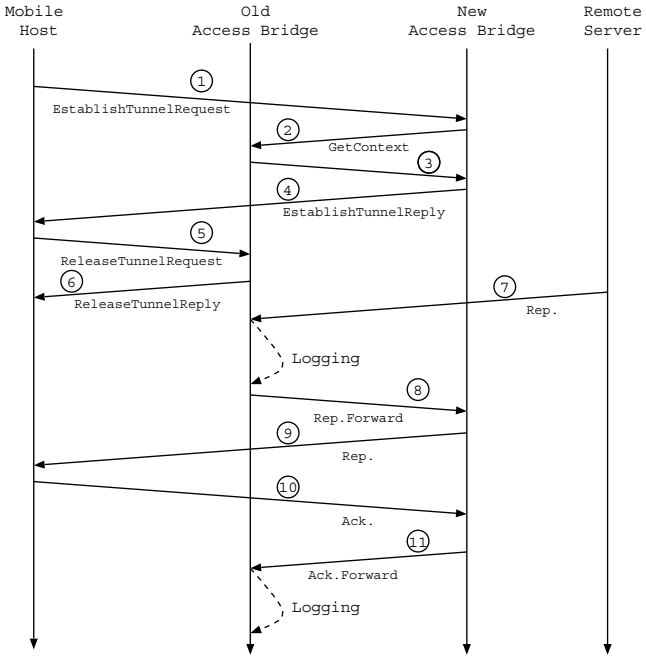

Figure 4. Handoff procedure

In Figure 4, the MH creates a network connectivity (in network layer) with the NAB. Then it sends a request message to establish a tunnel (message 1). The NAB uses information contained in the request message or acquired by querying the HLA to get the OAB of this MH. The NAB sends a message to the HLA to update this MH's location and invokes a handoff operation at the OAB (message 2). The OAB forwards necessary context data, such as *Sequence Number, Last Sequence Number Received, Connection ID*, to reconstruct the execution context in the NAB (message 3). The NAB sends the tunnel establishment reply to the MH (message 4) and the MH breaks the connection with the OAB (message 5 and 6). Afterwards, the MH sends and receives all messages through the NAB. The messages received by the OAB during the handoff (message 7), such as replies to former requests, etc., are forwarded to the NAB (message 8) and the NAB relays them to the

MH (message 9). The acknowledgement messages (message 10) are forwarded to the OAB (message 11) to update the corresponding message status in the logs to keep these MRs integrity. The GIOP requires that a reply should be sent in the same GIOP connection as the request came in [8]. So if a message is a reply for a request that was received through the OAB before the handoff, this message is encapsulated in a forward format and when the NAB receives it, the NAB should relay it to the OAB. All these forwarded messages are logged in the OAB.

4.4. Mobile Host Disconnection

An AB functions as a proxy between an MH and an SH. Its major function is to forward messages to and from the MH. We construct an AB with two parts (see Figure 5): Mobile side and Fixed side [8]. The mobile side connects with the MH by GIOP Tunnel, while the fixed side uses normal IIOP (Internet Inter-ORB Protocol) connections to communicate with remote static servers. The AB keeps different maps between these two parts by *Connection ID* specified in [8] for every associated TB. Using the AB as a proxy, we can hide sudden mobile host disconnection from the remote servers [15].

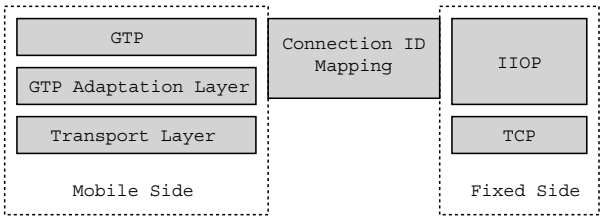

Figure 5. Access bridge ORB

According to the fact that an AB is a proxy for relaying GIOP messages, we define three statuses of a message in the AB.

- *Received*. This is the default status for a message when an AB receives it.
- *Sent*. When a message is relayed to an MH or an SH but before receiving the acknowledgement or the reply, the status of the message is *sent*.
- *Processed*. After an AB receives an acknowledgement message or a reply, it changes the corresponding message's status to *processed*.

If an AB receives a message which does not need to be relayed, the status of this message will be directly changed to *processed* after the AB processes this message.

During a sudden mobile host disconnection, the last connected AB still keeps IIOP connections with remote servers for a predefined time period. When the AB receives messages from the remote servers, it logs messages but does not forward them to the target MH (as message 7 in Figure 4) . When the AB gets a notification that the MH reconnects with the network, it forwards these received-but-not-sent messages to the MH (reconnects with the same AB) or the currently associated AB of this MH (reconnects with a new AB). If the MH recovers the connection with the same AB in the predefined time period, the AB will reuse these IIOP connections for successive communications. Otherwise the AB terminates all IIOP connections established for this MH. If the AB receives all the reply messages sent back from the remote server, it also closes these connections.

4.5. Mobile Host Crash

During disconnection, the state of the MH is kept intact. In case of mobile host crash, the local state is lost and needs to be recovered from the checkpoint and message logs. The MH is assumed to be fail-stop [2], i.e., the associated AB is able to detect the failure of the MH. Each failed MH can perform handoff and recovery procedure independently, which means that no other MHs need to roll back together.

First the MH initiates a handoff procedure as depicted in Section 4.3. After the successful handoff, it starts a state recovery procedure. This procedure includes four phases:

1. The HLA finds the location of the last checkpoint and forwards it to the NAB;
2. The HLA collects all successive MRs from the itinerary track of the MH and forwards them to the NAB;
3. The NAB sends the checkpoint, sorts the *processed* messages by their corresponding acknowledgement SNs, forwards these messages sequentially, and delivers the *sent* messages sequentially in their own SN order;
4. The MH initializes the application using the checkpoint and then executes the application. If a generated message has a counterpart message in the MR set, this message is inhibited and will not be sent out.

After applying the recovery procedure, the state of the MH will be restored to the state just before the failure. (We assume that the application is deterministic.)

A GIOP tunnel is shared by all GIOP connections to and from the TB [8] , so some messages maybe arrive at the TB earlier than the messages sent before them. We adopt a quasi-sender-based message logging mechanism for these messages [4], which means that the AB acts as message sender from the viewpoint of the TB. For reconstructing the same sequence of messages arriving before failures, we use

the SNs of the corresponding acknowledgements in MRs to sort these processed messages before sending them out sequentially.

In our approach, if a user moves from one AB to another, the stable storage for storing checkpoints and messages is changed accordingly. So if the mobile user traverses many times during a checkpointing period, the logged messages are scattered in these ABs. If we want to recover an MH from a failure or to revoke the stable storage for outdated messages, we need a method to find all these messages. Because the HLAs keep one itinerary track for each MH, we can employ the tracks to facilitate the messages collection and storage revocation, as described in Section 4.2.

The recovery period is time consuming because visiting different ABs is required to collect necessary MRs. The reason is that when a failure occurs, the messages are scattered. We can improve this recovery procedure by collecting messages to a stable storage near or in the current AB. A strategy proposed in [12] ensures that the message logs and the checkpoint corresponding to the MH are at the "predecessor" AB. To achieve this, during handoff, a message is sent to the predecessor AB to transfer the checkpoint and logs. But if the MH moves frequently to another AB, this strategy will still create heavy volume of data transfer. We improve this strategy by letting the HLA trigger the transfer of the checkpoint and logs. In the HLA, there is a daemon and an array of timers for each MH. If one timer is expired, the daemon will dispatch a thread to handle the data transfer for the corresponding MH. The thread will collect the last checkpoint and successive message logs and save the data in the current AB of this MH. The timer is adaptive. It will extend the time period if an MH moves frequently and it will shorten the time period if the MH maintains connection with an AB for a long time. If a checkpoint is taken during this message collection period, the HLA stops the related thread. We also can use the number of handoffs or the distance between the currently associated AB and the AB which contains the last checkpoint as the trigger of message collection.

4.6. Access Bridge Crash

An AB facilitates the connection mapping between an MH and an SH, so the AB is in the critical path. For tolerating access bridge crash, normal replication strategies can be adopted [1]. Recognizing the nomadic feature and the handoff mechanism in the mobile computing environment, we utilize a strategy that replicates the execution context and messages in an AB to its *previous* AB for each MH. A *previous* AB for an MH is an AB in its movement track just one hop before its current AB. If there is no movement track for this MH, we choose the HLA as the "previous" AB. This replication strategy is passive. Some messages that do not change the status of the AB will not be replicated. Because each MH has different movement tracks, this strategy generates different AB replicas for different MHs. After an AB failure, different MHs can move to different NABs to start the handoff procedures.

If an AB crashes, the MH will detect this failure and then start a handoff procedure. If in the current location area there is only one AB, the mobile user should explicitly move to another location for handoff. The handoff procedure has some differences from the normal handoff. The NAB first queries the HLA for the location of the replicated message logs and makes a request to the AB. The AB reconstructs the execution context from the message logs and sends this context to the NAB. The NAB initializes a new context for this MH according the received context and resends those messages which have no acknowledgments or replies and whose SNs are not in the vector which contains all the SNs of the messages received by this MH after its last checkpoint. After a successful handoff, the NAB informs the HLA that the recovery procedure is finished and the MH continues to work as in normal condition. The HLA removes the failed AB from the MH track to avoid to select the failed AB as a previous AB. If the MH moves back to the previous AB, the recovery procedure will be more efficient because all messages required to recovery are in the local storage. If the AB restarts after a failure, the MH can create connectivity with this AB just as a normal handoff from the previous AB.

To avoid re-execute resent messages after an access bridge crash, a remote server should do some special work. When the server sends a reply message to the crashed AB, it learns that the AB is not reachable and then logs this reply message locally. After a successful handoff, the NAB reissues the same request through a new GIOP connection, the server identifies this request, retrieves the corresponding reply from its local log, and sends it back. Therefore the server does not process the same request more than once and keeps the data consistent.

5. Simulation and Evaluation

A simulation model is constructed to evaluate our proposed scheme, which consists of one MH, five ABs and two SHs. The MH sends request messages to SHs which are selected randomly and the time interval between two successive messages is exponentially distributed with a mean of γ. The MH moves around in the mobile computing environment with a handoff rate which follows a Poisson process with rate ρ. Each AB has a static route to the SHs. The failure rate of the MH follows a Poisson process with rate λ. We assume that a failure is detected as its occurrence, so an MH performs recovery procedure instantly after a failure. The service rate of an SH is ω. Let α be the ratio of the

average cost transferring an application message or a checkpoint to the average cost transferring a control message over one hop of the wired network and δ be the ratio of the cost transferring a control message over one hop of the wireless network to the cost over one hop of the wired network. The checkpointing rate is τ. Every request has a corresponding reply and the MH completes successfully if it receives all the reply messages.

5.1. Program Execution Time

Figure 6 shows the execution time of the MH engaging checkpointing or not with the following variables : $\gamma = 0.1$, $\lambda = 0.001$, $\omega = 0.1$, $\alpha = 10$, $\delta = 10$, and $\tau = 0.05$. In this measurement, no handoff occurs. The execution time is normalized to the execution time without failures and handoffs.

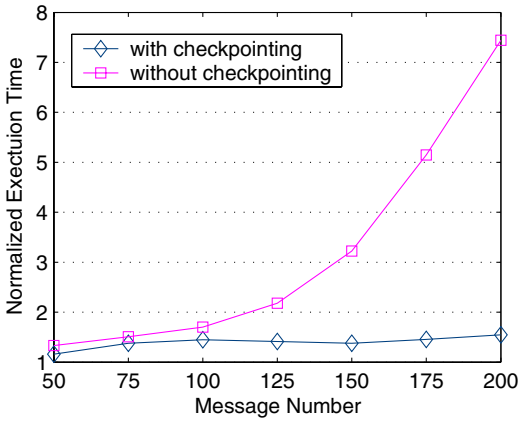

Figure 6. Program execution time

From this figure, we know that without checkpointing the normalized execution time increases dramatically as the message number increases. After engaging checkpointing, the time curve is plotted nearly horizontally. So the execution time with checkpointing has a linear relationship with the message number. We also can see that checkpointing and message logging incurs overheads due to no application message can be sent out during checkpointing and the message logging mechanism delays the message delivery.

5.2. Handoff Effect

To demonstrate how the handoff influences the execution time, we let the MH moves randomly in the ABs. In this simulation, the same parameters values are used and the message number is 100. Figure 7 shows the results which are also normalized to the execution time without failures and handoffs. It implies that the execution time increases linearly as the handoff rate increases.

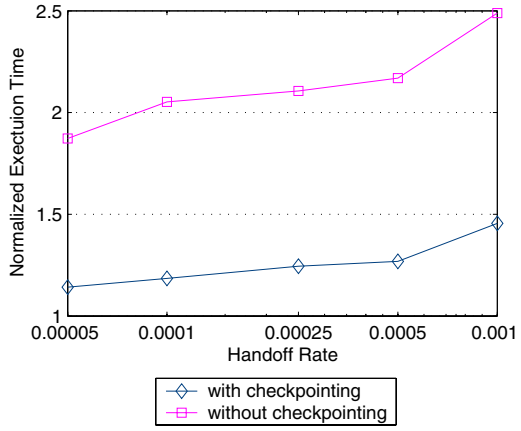

Figure 7. Execution time vs. handoff

As we described, no application message can be transmitted to or from the MH during handoff. After a handoff, the OAB has to forward its received replies during the handoff to the NAB. When a failure occurs, the checkpoints and message logs may be scattered in several ABs. All these increase the total execution time.

6. Conclusions

This paper describes a message logging and failure recovery protocol in wireless CORBA. It employs both quasi-sender-based and receiver-based message logging methods. The protocol can tolerate mobile host disconnection, mobile host crash and access bridge crash. It chooses the storage available at the AB as the stable storage to log messages and checkpoints. To tolerate the access bridge crash, it replicates an AB's state in the previous AB for each MH. It also engages the handoff mechanism as a means to recover from the access bridge crash.

A simulation model is constructed to evaluate the proposed scheme. After engaging checkpointing and message logging, the program execution time increases linearly as the message number increases in the presence of failures. The handoff affects the execution time by delaying message delivery and by scattering checkpoints and message logs in multiple ABs.

Acknowledgement

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4360/02E).

References

[1] S. Alagra, R. Rajagopalan, and S. Venkatesan. Tolerating mobile support station failures. *Proceedings of the 1st Conference on Fault Tolerant Systems*, pages 225–231, December 1995.

[2] M. Barborak, M. Malek, and A. Dahbura. The consensus problem in fault-tolerant computing. *ACM Computing Sruverys*, 25(2):171–220, June 1993.

[3] J. Jing, A. Helal, and A. Elmagarmid. Client-server computing in mobile environments. *ACM Computing Surveys*, 31(2):117–156, June 1999.

[4] D. B. Johoson. Distributed system fault tolerance using message logging and checkpointing. *Ph.D. Dissertation, Rice University*, December 1989.

[5] P. Krishna, N. H. Vaidya, and D. K. Pradhan. Recovery in distributed mobile environments. *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, pages 83–88, October 1993.

[6] L. E. Moser, P. M. Melliar-Smith, and P. Narasimhan. A fault tolearnce framework for CORBA. *The 29th International Symposium on Fault-Tolerant Computing*, pages 150–157, June 1999.

[7] N. Neves and W. K. Fuchs. Adaptive recovery for mobile environments. *Communications of the ACM*, 40(1):68–74, January 1997.

[8] Object Management Group. Telecom wireless CORBA. *OMG Doucment dtc/01-06-02*, June 2001.

[9] Object Management Group. The Common Object Request Broker: Architecture and specification, 2.6.1 edition. *OMG Document formal/02-05-15*, May 2002.

[10] T. Park and H. Y. Yeom. An asynchronous recovery scheme based on optimistic message logging for the mobile computing systems. *Proceedings of the 20th International Conference on Distributed Computing Systems*, pages 436–443, April 2000.

[11] H. Pham. *Software Reliability*. Springer-Verlag, Singapore, 2000.

[12] D. K. Pradhan, P. Krishna, and N. H. Vaidya. Recovery in mobile environments: Design and trade-off analysis. *The 26th International Symposium on Fault-Tolerant Computing*, June 1996.

[13] R. Ruggaber and J. Seitz. Using CORBA applications in nomadic environments. *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, pages 161–170, December 2000.

[14] R. Ruggaber and J. Seitz. A transparent network handover for nomadic CORBA users. *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 499–506, April 2001.

[15] B. Yao and W. K. Fuchs. Proxy-based recovery for applications on wireless hand-held devices. *The 19th IEEE Symposium on Reliable Distributed Systems*, pages 2–10, October 2000.

[16] B. Yao, K. F. Ssu, and W. K. Fuchs. Message logging in mobile computing. *Proceedings of IEEE Fault-Tolerant Computing Symposium*, pages 294–301, June 1999.