An Efficient Fault Tolerance Protocol with Backup Foreign Agents in a Hierarchical Local Registration Mobile IP

Choong Seon Hong, Ki-Woon Yim, Dae-Young Lee, and Dong-Sik Yun

A Mobile IP allows IP hosts to move between different networks without changing their IP addresses. Mobile IP systems supporting local registration were introduced to reduce the number of times a home registration with the remotely located home agent was needed. The local registration Mobile IP scheme enhanced performance by processing registration requests of mobile nodes at a local agent. The local registration approach may affect other aspects of the Mobile IP systems such as fault tolerance. In this paper, we briefly review previous solutions for supporting fault tolerance in local registration Mobile IP systems and propose a fault tolerance protocol with a backup foreign agent in a hierarchical local registration mobile IP to enhance the efficiency of such systems against foreign agent failures. We also describe the specification of the proposed protocol using LOTOS and perform its validation using MiniLite. Finally, we analyze the performance of our proposed fault tolerance protocol through simulation.

Manuscript received May 3, 2001; revised Oct. 26, 2001.

Choong Seon Hong (phone: +82 31 201 2532, e-mail: cshong@khu.ac.kr), Ki-Woon Yim (e-mail: iki77@unitel.co.kr), and Dae-Young Lee (e-mail: dylee@khu.ac.kr) are with the School of Electronics and Information, Kyung Hee University, Yongin, Korea.

Dong-Sik Yun (e-mail: dsyun@kt.co.kr) is with the Telecommunications Network Laboratory, KT, Daejeon, Korea.

I. INTRODUCTION

The IETF Base Mobile IP [1] allows an IP host to move between different subnetworks without having to disrupt established transport layer sessions. The mobility of the IP host is supported by the home agent (HA) and foreign agent (FA). The HA keeps track of the current location of its mobile nodes (MNs) and tries to keep the correspondent nodes (CNs) that are communicating with those MNs updated with the current location information. The FA forwards packets to and from the MNs that are currently being serviced in its area. One of the main features of the IETF Base Mobile IP protocol is that for location management, it specifies the minimum number of administrative messages using a simple implementation. However, the IETF Base Mobile IP suffers from some performance problems. One of them, high data latency, is caused by indirect triangular routing. The CN may need to communicate with an MN that is currently located in the same network. In such a situation, all packets from the CN to the MN travel from this network to the HA of the MN and then back to the original network where the MN and CN are located. This doubles the data path from the CN to the home network of the MN. The IETF Mobile IP with route optimization extension draft [2] attempted to resolve this problem, but it introduced additional overhead for administrative messages and more processing issues in the mobility support elements.

Another well-known problem is the overhead for frequent movement of an MN. This kind of overhead is due to the registration message exchange between the HA and FA. There may be many hops between the HA and the FA or the CN. Thus, it

may take a long time for the home agent to be aware of the mobility and to adjust to the new environment when the MN moves to a new foreign network. Much research on local registration Mobile IP systems has been devoted to solving this problem [3]-[7]. The basic idea of local registration Mobile IP is to allow an MN to send registration requests to the regional FA that processes such requests. The regional FA tracks regional movements but it does not forward the MNs request to the HA. In Mobile IP Local Registration with a Hierarchical Foreign Agent Approach [4]-[6], FAs are hierarchically arranged in a regional topology. The MN is also allowed to move from one service area of the regional topology to another service area of the same topology without requiring approval by the HA or the need to bind location information at the HA. However, the local registration approach needs to consider other aspects of the Mobile IP systems, such as the fault tolerance.

This paper describes the issues related to the operation of the local registration Mobile IP and its effect on FA fault tolerance as well as two previous approaches to implementing FA fault tolerance in such environments [8], [9]. We propose an FA fault tolerance protocol with a backup FA in a hierarchical local registration Mobile IP. The proposed protocol enhances the efficiency of the systems when the FA experiences failures.

To improve readability, we define the terms related to Mobile IP, using RFC 2002 [1], as follows:

Mobile Node: A host or router that changes its point of attachment from one network or subnetwork to another.
Home Agent: A router on a mobile node's home network which tunnels datagrams for delivery to the mobile node when it is away from home and maintains current location information for the mobile node.
Foreign Agent: A router on a mobile node's visited network which provides routing services to the mobile node while registered. The foreign agent de-tunnels and delivers datagrams to the mobile node that were tunneled by the mobile node's home agent.
Agent Advertisement: An advertisement message constructed by attaching a special extension to a router advertisement message.
Care-of Address: The termination point of a tunnel toward a mobile node for datagrams forwarded to the mobile node while it is away from home.
Correspondent Node: A peer with which a mobile node is communicating. A correspondent node may be either mobile or stationary.
Visitor List: The list of mobile nodes visiting a foreign agent.

The rest of the paper is organized as follows. Section II discusses related work on dealing with FA failure in hierarchical local registration Mobile IP. The design of our protocol architecture is presented in Section III. In Section IV, we describe the specification of our protocol using LOTOS and validation using MiniLite. In Section V, we compare our proposed scheme with previous proposals. Finally, we make a conclusion.

II. RELATED WORK

1. IETF Base Mobile IP

The goal of a Mobile IP is to provide mobility support for hosts who need to remain connected to the Internet without changing IP addresses. When an MN moves to a new location, it registers its current care-of address (CoA) with its home agent, which is attached to its home network. When a packet for the MN arrives at its home network, the HA intercepts the packet and forwards it to the CoA by encapsulation [10]. The FA then decapsulates the packet and delivers it to the MN. A detailed description can be found in [1].

To track the MN, whenever the MN changes its attachment to the Internet, a registration request needs to be sent back to the HA. While the MN can send out packets through the FA and along an optimal path, incoming packets have to travel through the HA. If the current location of the MN is close to the sender's location and the HA is far away from the MN, the packets have to take a long detour. Such a triangular routing scheme suffers from some performance problems.

2. Hierarchical Local Registration Mobile IP

As mobile devices are becoming smaller and more convenient, they often require reduced power consumption to avoid carrying larger and heavier batteries. Less powerful wireless transceivers cannot reach very long distances. On the other hand, smaller and denser cells provide a higher aggregate bandwidth and can locate a mobile device more accurately.

The downside of a wireless network with small cells is that a moving host may cross cells very often, resulting in frequent handoffs. If the MN leaves one registration area before registering its next CoA, some packets may be lost, and frequent handoffs aggravate the performance problem. The hierarchical local registration Mobile IP (HLRM-IP) [4]-[6] is meant to alleviate this problem. The FAs in a domain are organized into a hierarchy to handle local movements of the MNs within their domain (Fig. 1). An FA includes in its Agent Advertisements a vector of care-of addresses, which are the IP addresses of all its ancestors as well as its own. When an MN arrives at an FA, it registers with its HA not only the FA as the CoA but all its ancestors. A registration goes through the FA with all its ancestors and the HA and is processed by them.

When a packet for the MN arrives at its home network, the HA tunnels it to the root of the FA hierarchy. When an FA receives such a tunneled packet, it re-tunnels the packet to its next lower-level FA. Finally, the lowest-level FA delivers it directly to the MN. Therefore, any FA that processes a registration must record the next lower-level FA as the other end of the forwarding tunnel.

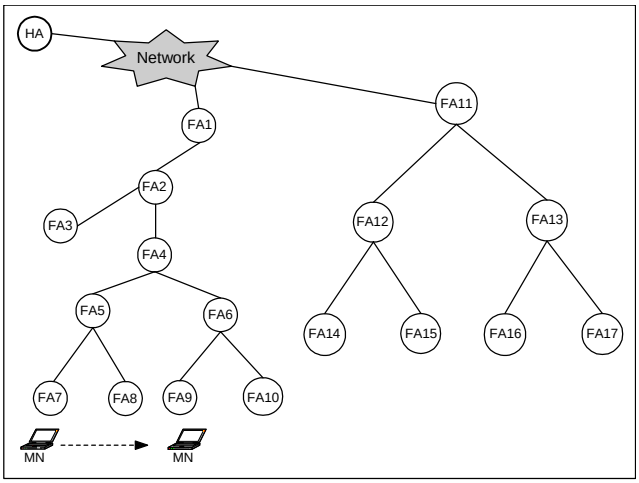


Fig. 1. Hierarchical FAs.

As an example, in Fig. 1 when the MN first arrives at FA7, it registers FA7, FA5, FA4, FA2, and FA1 as its care-of addresses. The registration request goes through this path to the HA, and the registration reply travels the same path in the opposite direction. The packet for the MN is intercepted by the HA and tunneled to FA1, which in turn re-tunnels it to FA2, which again re-tunnels it to FA4, which again re-tunnels it to FA5, which again re-tunnels it to FA7. And finally, FA7 delivers it directly to the MN.

When a handoff occurs, the MN compares the new vector of the care-of addresses with the old one. It chooses the lowest-level FA that appears in both vectors and sends a Regional Registration Request to that FA. A higher-level agent does not need to be informed of this movement since the other end of its forwarding tunnel still points to the current location of the MN. In Fig. 1, when the MN moves from FA7 to FA8, FA5 is the target of the regional registration, and FA4, without knowledge of this movement, still correctly re-tunnels the packet to FA5. When the MN moves from FA8 to FA9, the registration target is FA4. In the meantime, the HA has no knowledge of the local movements and none of these registrations reaches the HA; hence, registration overhead is reduced.

3. Fault Tolerance in HLRM-IP

A characteristic that distinguishes HLRM-IP from other Mobile IP approaches [4]-[6] is that the failure of any of the FAs along the path between the root FA and the leaf FA will cause the MN located at the leaf FA to lose its connectivity. Figure 2 shows the situation where there is no faulty FA, and FA7 is the current service area where the MN is located. The data packet generated from a CN and destined to the MN will be forwarded to the HA. Then it will tunnel the packet to the root FA (FA1) and FA1 will tunnel the packet to FA2. After that, FA2 will tunnel the packet to FA4 and FA4 will tunnel it to FA5. Finally, FA5 will tunnel the packet to FA7. The failure of any of those FAs will break the path between the root and the leaf FA (FA7). This situation occurs even if there is a route bypassing the faulty FA (FA4 in Fig. 2), for example, the path FA2-R1-FA5, because FA4 is needed to point to the next lower FA in the hierarchy and not to be just a routing node. Considering the case of Non-Hierarchical Local Registration systems and assuming the failure of FA4, we can see that FA4 is not needed for its tunneling function. The other routing node R1 will be able to route the packet and deliver it to the FA serving the MN.

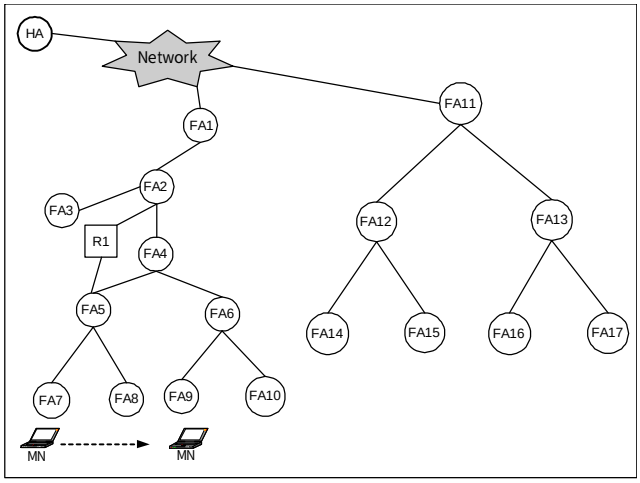


Fig. 2. HLRM-IP.

H. Omar et al. presented two solutions to solve this problem [8], [9]. The first solution reverts the MNs affected by the failure of the FA to a Non-HLRM-IP Mode (Revert to Non-HLRM-IP Mode). The other approach attempts to heal the broken segment in the encapsulation-decapsulation chain caused by the failure of the FA by removing this faulty FA from the hierarchy (Self-Healing Mode). The basic idea of Revert to Non-HLRM-IP is that if the failure of the FA occurs in HLRM-IP, the MNs affected by the faulty FA request home registration to overcome the failure of the FA just like that of the IETF Base Mobile IP. The basic idea of the Self-Healing Mode is to heal the breakage in the path caused by the faulty FA. This is accomplished by bypassing the faulty FA so that the FA in the hierarchical level just above the faulty FA will remove the faulty

FA from its copy of the hierarchy and consider the FA in the level just below the faulty FA as its tunnel end. For example, when the failure of the FA4 occurs in Fig. 2, the new tunneling path becomes FA1-FA2-FA5-FA7.

III. FAULT TOLERANCE PROTOCOL WITH BACKUP FA IN HLRM-IP

1. Background

In section 2.3, we described two solutions that provide fault tolerance in the HLRM-IP system. The second solution (Self-Healing Mode) retains more of the hierarchical structure than the first solution (Revert to Non HLRM-IP Mode). However, in the second solution, if the number of faulty FAs increases, the removed FAs increase, so the hierarchical tree structure tolerating the FA fault has no meaning. To overcome this problem, we propose an efficient fault tolerance protocol that maintains the hierarchical architecture of the FA with a backup FA in HLRM-IP.

2. Configuration of the Backup FA

The backup FA is configured so that it uses a router that has the same routing path as applicable FAs. The backup FA has to support some special mechanism such as explicit routing to send the same destination packet in both directions. For example, in Fig. 3, the backup FA of FA2 is R2 and it has the same routing path as FA2. R1 cannot be the backup FA of FA2 because R1 has no path to FA3. In the same way, the backup FA of FA4 is R3. It is necessary to support more than one FA fault, so an additional routing path, the path between R2 and R3 in Fig. 3, is required between the FA just above the faulty FA and the FA just below the faulty FA. The backup FAs of all the FAs in the hierarchical structure are constructed as described above. We will consider two elements, called the *FA Registry* and *Backup FA Registry*, which are located with the root FA that keeps information regarding the FAs and backup FA on the different levels of the hierarchy. In Fig. 3, the backup FAs of the FAs in the hierarchical level below the FA4 were omitted to make the figure simpler.

The Backup FA that receives the message activates the process to act as an FA, performing the role of the FA in which the fault occurred. A more detailed description of the operation follows. Let's assume that MNs are connected to FA7, FA8, FA9, and FA10 and receive data in the FA hierarchical structure that has the backup FA as shown in Fig. 3. When the fault occurs in FA4, the following scenario is processed to recover the fault.

a) The FA (FA2) that is just above the level of the faulty FA (FA4) in the hierarchical level detects the failure of FA4.

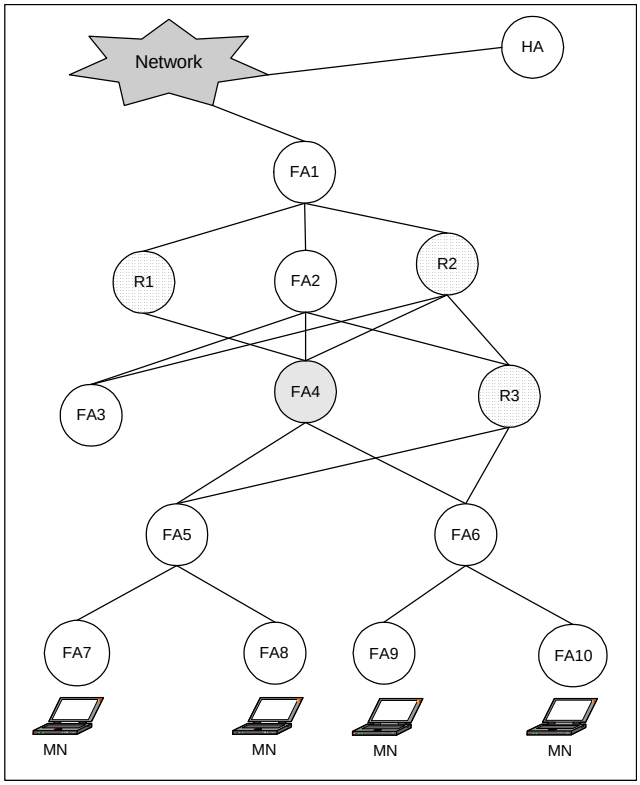


Fig. 3. Hierarchical FAs topology with backup FAs.

b) FA2 obtains the list (FA5, FA6) of the FAs at the next level below the level in which the fault occurred. The information about the backup FA (R3) is obtained through the Backup FA Registry in FA1.

c) FA2 updates the level below it from FA4 to R3 and sends a message to activate R3 to perform FA4's functions. FA2 also informs the FAs at the level below FA4 (FA5 and FA6) of the fault at FA4. FA2 obtains this information from the FA Registry and reports that R3 will perform FA4's functions.

d) After being informed by FA2 of FA4's faulty state, FA5 and FA6 update their information about the next level above them from FA4 to R3 and notify R3 that they are providing services.

e) After adding the MN list from FA5 and FA6 to its own visiting list, R3 sends its modified MN list to FA2.

f) When FA2 receives the MN list from R3, it updates its own visiting list.

g) FA1 stores the change in the hierarchical structure in the FA Registry and the Backup FA Registry that is located in FA1. (The FA Registry updates R3 instead of FA4 and the Backup FA Registry updates FA4 instead of R3)

h) FA2 periodically checks the faulty FA (FA4). If FA4 has recovered from the fault, FA2 detects it. Then, in order to reset FA4 into the original FA hierarchical structure, FA2 acts as if the fault had occurred in R3. (In the backup FA registry, FA4

was stored as the backup FA of R3.) Therefore, the hierarchical structure of the original state is restored through the process from step b) to step g).

Using this operation scenario, we can recover a fault by replacing a faulty FA with a backup FA.

Figure 4 shows the UDP field that is added for the backup FA protocol proposed in this paper. The data format in Fig. 4(a) is the request message that the FA that detects a failure sends to the root FA, notifying it to search for a backup FA to replace the faulty FA. The reply message format from the root FA is illustrated in Fig. 4(b). Figure 4(c) shows the message format that is sent from the FA that detects a failure to the backup FA for activating the agent function. The reply message format corresponding to the activation message by a backup FA is shown in Fig, 4(d). Figure 4(e) is the message format for the FA that detects a failure to notify FAs which will be affected, and for giving the backup FA the information about the faulty FA. Figure 4(f) shows that FAs (FA5, FA6 in Fig. 3) that are affected by the failure send registration request messages to the backup FA to register the information with the MNs that they serve. Using this registration message, the affected FAs register with the backup FA all the information of the MNs that they manage.

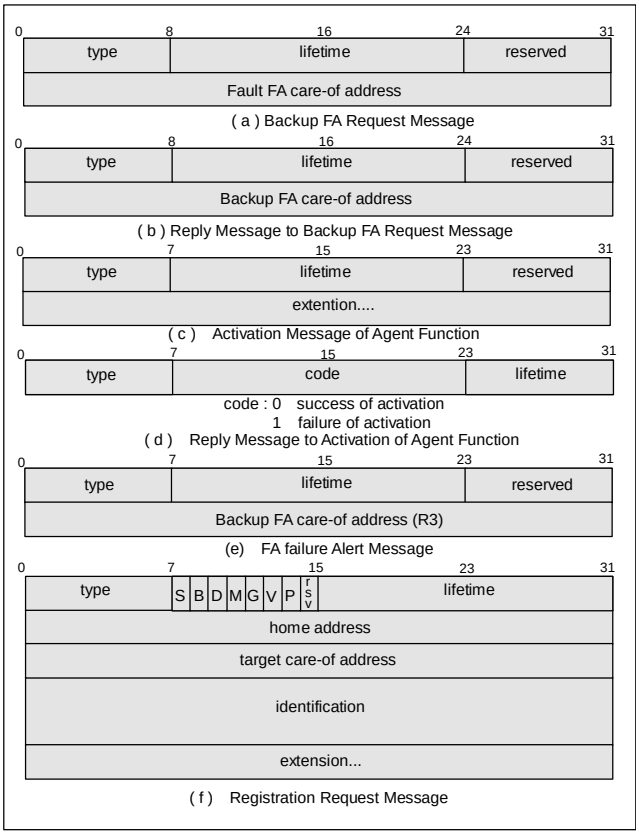


Fig. 4. Message format.

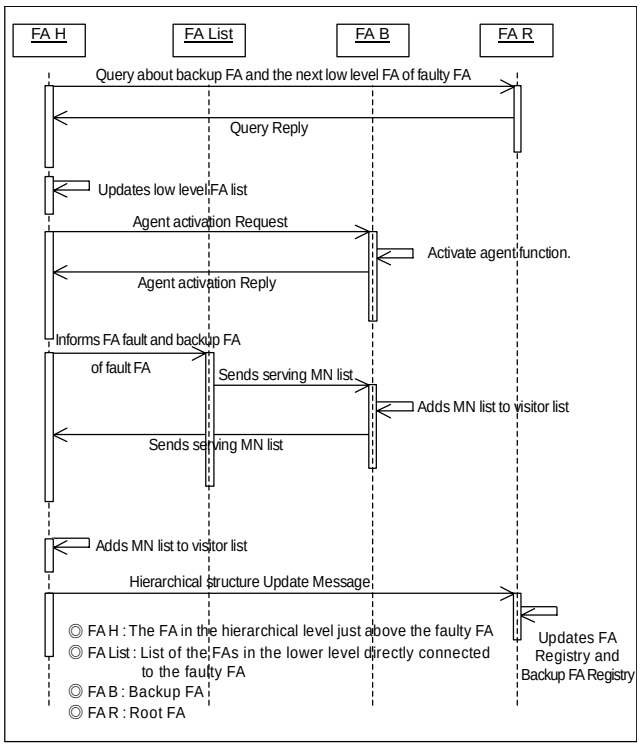


Fig. 5. Sequence diagram of proposed protocol.

The sequence diagram of the proposed protocol is shown in Fig. 5.

Figure 6 shows the comparison of the proposed protocol with the Self-Healing Mode that H. Omar et al. proposed. For the case when a fault does not occur and the MN has been connected to FA7, Fig. 6(a) shows the packet tunneling path toward the MN. Figures 6(b) and (c) show the tunneling routes of the protocol that is proposed in the Self-Healing approach and our proposed protocol for when FA2 and FA4 are in a fault state.

In the Self-Healing Mode, when a fault occurs, the hierarchical structure is modified from five levels to three levels. This is because of the removal of FA2 and FA4 from the hierarchical structure. The proposed protocol (Fig. 6(c)) keeps the five levels of the original hierarchical structure by using the backup FA, that is, the proposed process does not change the hierarchical structure. Let's examine the transmission process of the data packet between FA1 and FA7 in Figs.(b) and (c). With the proposed protocol, the data packet is transmitted through the designated tunneling path, but with the Self-Healing Mode, packets are transmitted through the normal router using a universal routing algorithm. In addition, in the proposed protocol, if an MN moves to FA9, the MN sends a registration request to B4, but in the Self-Healing Mode, because the MN must send the registration request to FA1, the registration time becomes longer. Also, because there are many requests for an MN's reg-

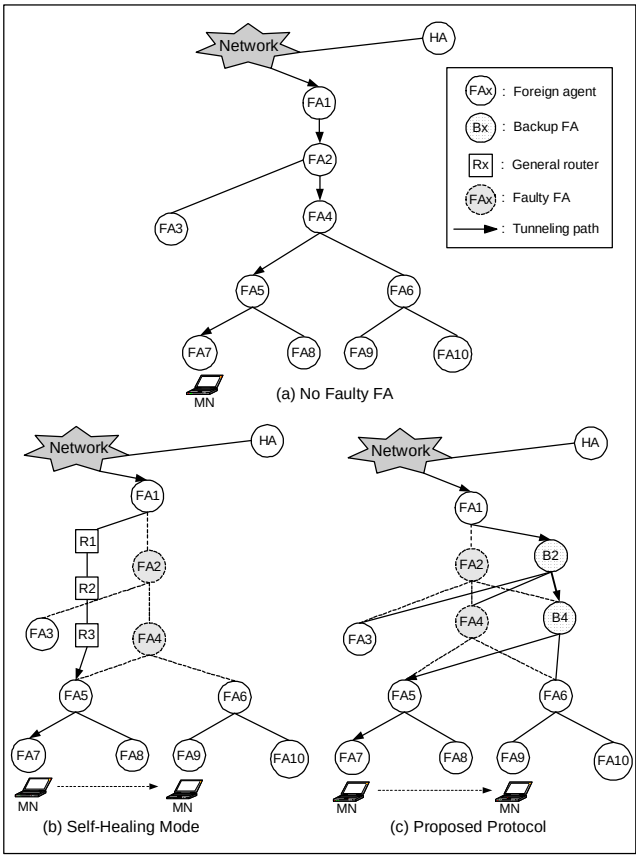


Fig. 6. A comparison of proposed method with Self-Healing Mode when a fault occurs to FA.

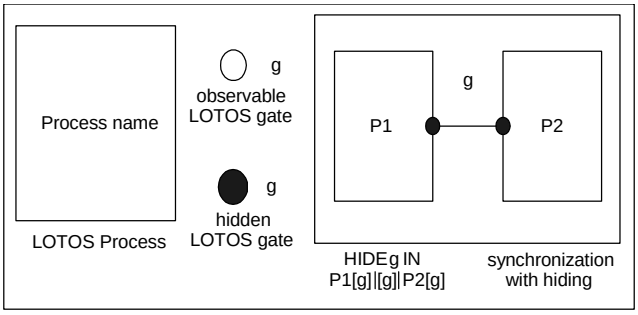


Fig. 7. Graphical notation.

istration to FA1 when there are frequent handoffs, overhead can occur in FA1. However, since the original hierarchical structure is preserved, these problems do not occur with the proposed protocol.

IV. FORMAL DESCRIPTION AND VALIDATION OF THE PROPOSED PROTOCOL

We describe the specification of our protocol using LOTOS [11] in this chapter. We also describe the validation of the protocol specification using MiniLite toolset [12], [13].

1. Specification of the Protocol

In a LOTOS specification, the abstract data type section defines the data types that can be used by the behavioral section. The abstract data type section of the proposed protocol specification describes the data structures, such as FA_H, FA_R and FA_B. The abstract data types also include basic and parameterized data. For simplicity we refrain from discussing these abstract data types here.

Figure 7 summarizes the graphical notation used in this paper for representing LOTOS processes.

Our graphical notation allows the representation of the structure of the process. Each process is represented in terms of its sub-processes and their interconnection through gates. For processes with a more complex structure, or with behavior that contains events, we simply present the LOTOS behavior specification.

In the behavioral section of the specification, our proposed protocol has four main processes.

Figure 8 presents the high level structure of the protocol LOTOS specification. This structure was constructed according to a fault tolerance protocol software architecture.

Process FA_H (Fig. 9) consists of three subprocesses: RequestFAList, Activation and RequestUpdateFAList. The process RequestFAList consists of two subprocesses: Detect FAFailure and BackupFA&FAListRequest. Detect FAFailure detects an FA failure through gate *'detection'* and sends the data to the BackupFA&FAList through gate *'trans_fault'*. BackupFA&FAList Request sends a request for a query about the backup FA and the level below the faulty FA through gate *'query'*. Process Activation consists of three subprocesses: ReceiveBackupFA&FAList, FAFailureAlert&AgentActivation and HandleException. ReceiveBackupFA&FAList receives a list (the backup FA, the lower level FA list) transfer request through gate *'query_out'* and receives the state of the backup FA through gate *'code'*. If the state of the backup FA is available, FA_R sends the list to FAFailureAlert&AgentActivation through gate *'trans_list'*. FAFailureAlert&AgentActivation sends activation data to the backup FA through gate *'activation'* and sends notification of the FA failure to the lower level FA through gate *'alert'*. If the state of the backupFA is not available, ReceiveBackupFA&FAList sends the serving MN list to HandleException through gate *'trans_exception'*. HandleException sends the FA fault alert (FFA) message to all the MNs being served through gate *'exception'*. All the MNs who receive the FFA message from the FA_R send the home registration request to recover the FA fault using Revert to Non-HLRM-IP Mode.

The process *UpdateFAList* consists of two subprocesses: Re-

```
SPECIFICATION  Protocol [detection, registration ] (...) : noexit

BEHAVIOUR
 HIDE        trans_req, trans-fault, trans_list, trans_alert IN
      (FA_H [detection, query, query_out, code, activation, alert, exception, registration,
             update](...)
       | [ query,qurey_out, activation, alert, registration] |
       FA_R [query, query_out, code update, ack](...)
       FA_B [activation, list, registration](...)
       FAList[alert,list](...))

WHERE
    PROCESS  FA_H  [detection, query, query_out, code,activation, alert, exception,  registration,
                   update]: noexit :=
          (detection; query; RequestFAList[detection, query]
          [] query_out; code; activation; alert; exception; Activation[query_out, code, activation, alert]
          [] registration; update; UpdateFAList[registration, update]
    )
  WHERE
      PROCESS   RequestFAList[detection, query]: noexit:=
                    DetectFAFailure[detection]
            >>    BackupFA&FAListRequest[query]
        WHERE
            PROCESS    DetectFAFailure[detection]: :noexit::=
                           detection; BackupFA&FAListRequest[query]
          ENDPROC
            PROCESS    BackupFA&FAListRequest[query]:noexit:=
                           query;
          ENDPROC
        ENDPROC
        PROCESS  Activation[query_out, code, activation, alert, exception]: noexit:=
                     ReceiveBackupFA&FAList[query_out, code]
                  >>  ( FAFailureAlert&AgentActivation[activation, alert]
                      [] HandleException[exception] )
        WHERE
            PROCESS ReceiveBackupFA&FAList[query_out, code]: :noexit::=
                        query_out; code; (FailureAlert&AgentActivation[activation, alert]
                                    [] HandleException[exception]  )
          ENDPROC
            PROCESS   FAFailureAlert&AgentActivation[activation, alert] ::noexit:=
                        activation; alert;
          ENDPROC
            PROCESS   HandleException[exception] ::noexit:=
                        exception;
          ENDPROC
        ENDPROC
        PROCESS   UpdateFAList[registration, update] :noexit:=
                      ReceiveRegistrationRequest[registration]
                  >>  HandleReceivedRequest[update]
        WHERE
            PROCESS   ReceiveRegistrationRequest[registration]:noexit::=
                        registration; HandleReceivedRequest[update]
          ENDPROC
            PROCESS   HandleReceivedRequest[update]:noexit:=
                        update;
          ENDPROC
        ENDPROC
      ENDPROC
    PROCES
        S             FA_R [query, query_out, code, update, ack]: noexit :=
                          (query; query_out; code; SendFAList[query, query_out, code]
                          [ ] update; ack;  UpdateFAList[update, ack]
                          )
    WHERE
        PROCESS         SendFAList[query, query_out, code]: noexit:=
                            ReceivedBakcupFA&FAListRequest[query]
                        >>    HadleReceivedRequest[query_out, code]
        WHERE
            PROCESS       ReceivedBakcupFA&FAListRequest[query]:noexit::=
                            query; HadleReceivedRequest[query_out, code]
          ENDPROC
            PROCESS       HadleReceivedRequest[query_out, code] :noexit:=
                                query_out; code;
          ENDPROC
        ENDPROC
       PROCESS      UpdateFAList[update, ack]: noexit:=
                        ReceiveUpdateRequest[update]
                    >>  HandleReceivedRequest[ack]
        WHERE
            PROCESS     ReceiveUpdateRequest[update]:noexit::=
                            update; HandleReceivedRequest[ack]
          ENDPROC
            PROCESS     HandleReceivedRequest[ack] :noexit:=
                            ack;
          ENDPROC
      ENDPROC
    ENDPROC

    PROCESS       FA_B [activation, list, registration]: noexit :=
                        (activation; Activate[activation]
                          []list; registration;  Registration[list, registration]
                    )
     WHERE
        PROCESS     Activate[activation]: noexit:=
                        activate;
        ENDPROC
          PROCESS   Registration[list, registration]: noexit:=
                        ReceiveMNList[list]
                    >>  SendMNList[registration]
        WHERE
            PROCESS     ReceiveMNList[list]:noexit::=
                            list; SendMNList[registration]
          ENDPROC
            PROCESS     SendMNList[registration]  :noexit:=
                            ack;
          ENDPROC
        ENDPROC
    ENDPROC

    PROCESS    FAList [alert, list]: noexit :=
                   ReceiveFAFailure[alert]
               >>  SendMNList[list]
     WHERE
        PROCESS    ReceiveFAFailure[alert]: noexit:=
                       alert;  SendMNList[list]
          ENDPROC
            PROCESS   SendMNList[list]: noexit:=
                          list;
        ENDPROC
    ENDPROC
ENDSPEC
```

Fig. 8. High-Level Specification.

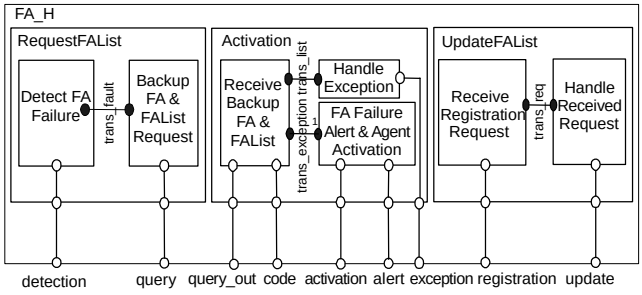


Fig. 9. Structure of process FA_H.

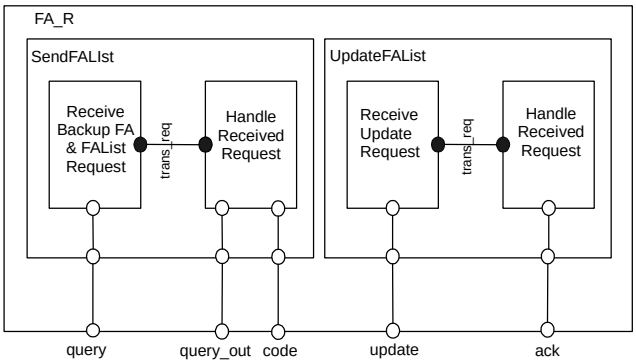


Fig. 10. Structure of process FA_R.

ceiveRegistrationRequest and Send UpdateFAList. Receive RegistrationRequest receives a registration request through gate *'registration'* and sends the data transfer parameters to HandleReceivedRequest through gate *'trans_req.'* HandleReceivedRequest handles the request and sends the data to update the Registry in the root FA through gate *'update.'*

Process FA_R (Fig. 10) consists of two subprocesses: SendFAList and UpdateFAList. Process SendFAList consists of two subprocesses: ReceivedBackup FA&FAList Request and HandleReceivedRequest. ReceivedBackupFA&FAList Request receives the request through the gate *'query'* and sends the data to HandleReceivedRequest through gate *'trans_req.'* HandleReceivedRequest handles the request, sends a list (backup FA, the lower level FA list) through gate*'query_out'* and sends information on the state of the backup FA through gate *'code.'* Process UpdateFAList consists of two subprocesses: ReceiveUpdateRequest and Handle ReceivedRequest. ReceiveUpdateRequest receives a request for updating the Registry in root FA and sends the request to HandleReceivedRequest through gate *'trans_req.'* HandleReceivedRequest handles the request and sends an acknowledgement through gate *'ack.'*

Process FA_B (Fig. 11) consists of two subprocesseses: Activate and Registration. Activate receives the activation request to activate the FA function through the gate *'activation.'* Process Registration consists of two subprocesses: ReceiveMNlist and SendMNlist. ReceiveMNlist receives a serving MN list

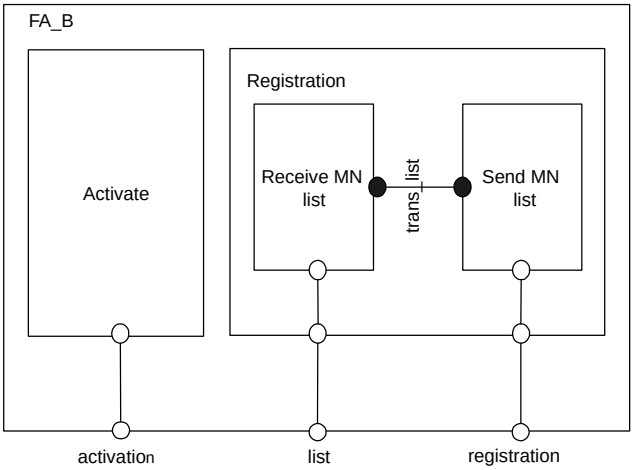


Fig. 11. Structure of process FA_B.

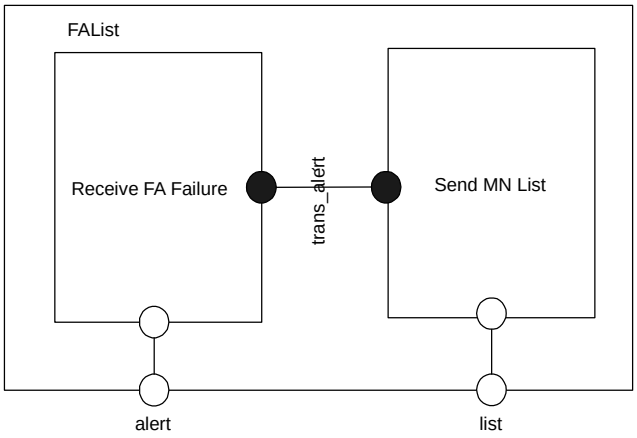


Fig. 12. Structure of process FAList.

from the level below the faulty FA through gate *'list'* and sends the list to SendMNlist through gate *'trans_list.'* SendMNlist sends the MN list through gate *'registration.'*

Process FAList (Fig. 12) consists of two subprocesses: ReceiveFAFailure and SendMNlist. ReceiveFAFailure receives the FA failure alert message through gate *'alert'* and sends the FA failure alert message to register serving MNs through gate *'registration.'*

2. Validation of Protocol

This section presents the validation of the protocol specification. We used the toolset MiniLite. Our approach combines simulation, testing and verification. Smile was used for the validation of data types. This tool was intensively used for the analysis of the most important data types of the specification. For the validation of the behavioral section we used Smile to simulate pieces of the specification. After the simulation of the processes, we executed the whole specification with a test process. The test process contained several test sequences that had to be performed manually and aimed to cover specific execution scenarios.

Finally, we generated and analyzed the Extended Finite State Machine (EFSM) of the specification. By using Smile, we generated the EFSM of each process defined in the specification. These state machines were transformed into an automata representation code called FC2, which was the input for the verification tools.

V. SIMULATION RESULTS

In this section, we compare and evaluate the performance of the Revert to Non-HLRM-IP, Self-Healing Mode with the fault tolerance protocol with a backup FA that is proposed in this paper.

The Network configuration for simulation is as shown in Fig. 13. Even though there are backup FAs for all the FAs in the simulation network topology, for simplicity we omitted them in the figure. In the real simulation they were considered. The delay over a link among FAs on the hierarchy was set at 3 ms, that between any FA and a routing element at set at 3 ms, that between the HA of the MN and the root FA at 50 ms, and that between the CN and the HA at 30ms.

We simulated the MN talking to a distant fixed host (CN). The CN sends one 200-byte packet every 20 milliseconds. We set three different simulation conditions in the configuration and assumed that FA failures randomly occur in the given duration.

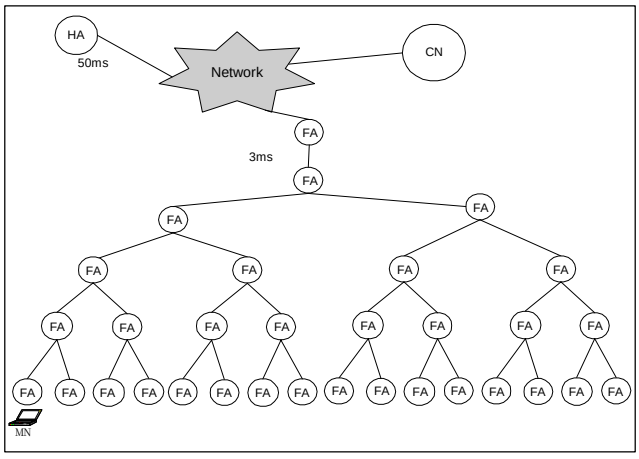


Fig. 13. Simulation network topology.

Figure 14 shows the packet loss rate by the number of times that FAs experience a fault when 50 MNs move with an average of two handoffs per minute in the hierarchical structure shown in Fig. 13.

Figure 15 shows the packet loss rate by the number of MN handoff occurrences in a hierarchical structure in which two FA

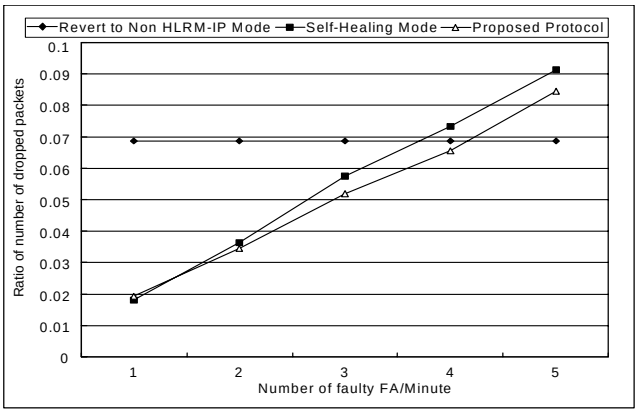


Fig. 14. Packet loss rate versus number of faulty FA.

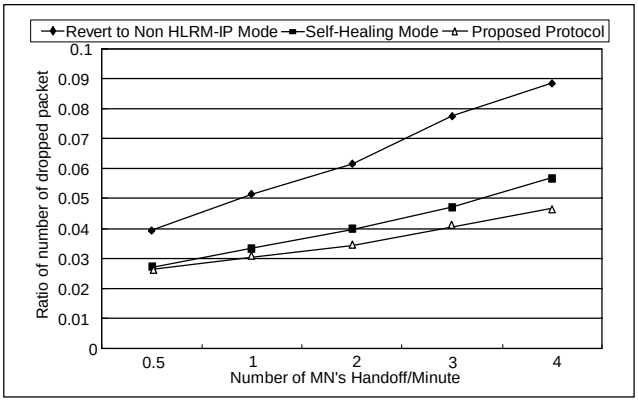


Fig. 15. Packet loss rate by number of MN's Handoff.

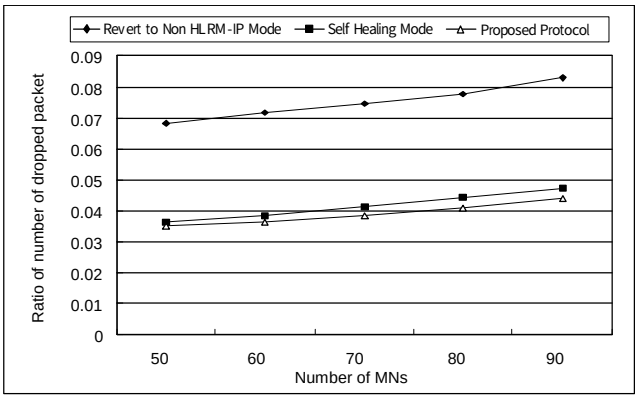


Fig. 16. Packet loss rate versus the number of MNs.

faults per minute occur.

Figure 16 shows the packet loss rate versus the number of MNs when the fault occurs two FAs per minute and MNs move with two handoffs per minute in the hierarchical structure.

Let's look at the result of Fig. 14. Revert to Non-HLRM-IP Mode has a constant packet loss rate even if the number of FAs when the fault occurs increases. In the Revert to Non-HLRM-IP Mode, even though the number of the FA faults increase, it does not need to recover the fault, because this protocol ignores the hierarchical structure when a fault occurs and all MNs send their registration requests to the HAs. However, because all MNs that are affected by the faulty FAs send a registration request to their HA, Revert to Non-HLRM-IP Mode has a higher packet loss rate than other methods even if the number of faults occurring in FAs is not large. Figure 14 shows that the method proposed in this paper performs better than the Self-Healing Mode except when one FA fault occurs. Revert to Non-HLRM-IP Mode performs better than other methods when faults occur in five FAs, but it is not an important result because the case in which many faults occur simultaneously is very rare. In addition, because all MNs send registration requests to HAs, Revert to Non-HLRM-IP Mode is highly dependant on the location of the HAs at the time of the fault recovery. Revert to HLRM-IP Mode may be inappropriate for the proposed fault tolerance protocol when the MN's HA is located comparatively far away.

The packet loss rate versus the number of handoffs is shown in Fig. 15. If the number of handoff occurrences increase, the increase of packet loss of Revert to Non-HLRM-IP Mode is dominant. The protocol that is proposed in this paper has the lowest increase. The reason for the proposed method having the best performance among the three methods may be because the delay in the registration process is less than that of other methods at the handoff occurrence. This result is derived from the fact that the proposed method keeps the hierarchical structure using the backup FA at the time of the fault recovery.

Figure 16 shows the increase of the packet loss rate in accordance with the increase of the number of MNs. The Revert to Non-HLRM-IP method shows the highest increasing rate according to the number of MNs. Other methods show similar results.

According to the simulation results, Revert to Non-HLRM-IP protocol performs best among the three methods when the location between the HA and MN is close and when there are many faulty FAs. However, it is seldom efficient, because of the overhead from the MN's registration message, and because it is rare for many FA faults to occur simultaneously. Our simulation results show that the proposed protocol outperforms the other two methods in all simulation environments.

VI. CONCLUSIONS

We proposed a fault tolerance protocol with backup FAs to solve problems in previous studies for fault recovery in the hierarchical structure of FAs. The fault tolerance protocol with backup FAs maintains the advantage of the local registration Mobile IP protocol because it keeps the hierarchical structure even if a fault occurs. But the method that Omar et al. proposed

cannot keep the hierarchical structure.

Accordingly, our simulation results demonstrated that our proposed fault tolerance protocol with a backup FA is a more efficient fault recovery protocol.

Our future work will include the implementation of the proposed scheme in the wireless network environment so that the proposed fault tolerance protocol can be realized. In addition, because the backup router can suffer traffic concentration, we will measure and analyze the traffic in the backup router.

REFERENCES

[1] Charles Perkins, "IP Mobility Support," IETF RFC 2002, Network Working Group, Oct. 1996.

[2] Charles Perkins and David Johnson, "Route Optimization in Mobile IP," IETF Internet Draft, http://search.ietf.org/internet-drafts/draft-ietf-mobileip-optim-11.txt, Sep. 2001.

[3] Chu-Sing Yang, Kun-da Wu, and Chun-wei Tseng, "Support an Efficient Connection for Mobile IP," *Proc. of the Ninth Int'l Workshop on Database and Expert Systems Application*, Aug. 1998, pp.514-519.

[4] Eva Gustafsson, Annika Jonsson, and Charles E. Perkins, "Mobile IP Regional Registration," IETF Internet draft, http://search.ietf.org/internet-drafts/draft-ietf-mobileip-reg-tunnel-05.txt, Sep. 2001.

[5] Charles Perkins, Kuang-Yeh Wang, "Optimized Smooth Handoffs In Mobile IP," *Proc. of the 4th IEEE Symposium on Computer Communications*, June 1999, pp. 340-346.

[6] Karim El Malki, Pat R. Calhoun, Tom Hiller, James Kempf, Peter J. McCann, Ajoy Singh, Hesham Soliman, and Sebastian Thalanany, "Low Latency Handoffs in Mobile IPv4," IETF Internet Draft, http://search.ietf.org/internet-drafts/draft-ietf-mobileip-lowlatency-handoffs-v4-03.txt, Nov. 2001.

[7] K.E. Malki, P.R. Calhoun, T. Hiller, J. Kempf, P.J. McCann, A. Singh, H. Soliman, and S. Thalanany, "Low Latency Handoff in Mobile IPv4," draft-ietf-mobileip-lowlatency-handoffs-v4-01.txt, May 2001.

[8] H. Omar, T. Saadawi, and M. Lee, "Support For Fault Tolerance In Local Registration Mobile-IP Systems," *Proc. of the 1999 Milcom*, vol. 1, Oct. 1999, pp.126-130.

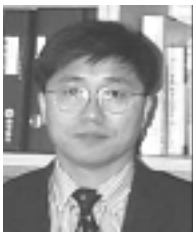
[9] H. Omar, T. Saadawi, and M. Lee, "Support Reduced Location Management Overhead and Fault Tolerance in Mobile IP", *Proc. of the 4th IEEE Symposium on Computer Communications*, June 1999, pp. 347-354.

[10] Charles Perkins, "IP Encapsulation within IP," Network Working Group, Request for Comments 2003, Oct. 1996.

[11] ISO/IEC. LOTOS "A formal Description Technique Based on Temporal Ordering of Observational Behaviour," International Standard 8807, International Organization for Standardization – Open Systems Interconnection, Geneve, 1989.

[12] T. Bolognesi, E. Najm, and P.A.J. Tilanus. "G-LOTOS: A graphical Language for Concurrent Systems," *Computer Networks and ISDN Systems*, vol. 26, no. 9, June 1994, pp.1101-1127.

[13] Clever R. Farias, Luis F. Pires, Wanderley L. Souza, and Celio E. Moron, "Specification and Validation of a Real-Time Parallel Kernel Using LOTOS," *Proc. of MASCOTS 2001*, Aug. 2001.



Choong Seon Hong received his BS and MS degrees in electronic engineering from Kyung Hee University, Seoul, Korea, in 1983 and 1985. From 1986 to 1987, he served in the military as an engineer. In 1988 he joined Korea Telecom, where he worked on N-ISDN and Broadband Networks as a Member of the Technical Staff. In September 1993, he joined Keio University in Japan. He received his PhD degree from the Department of Information and Computer Science at Keio University in March 1997. After obtaining his doctorate, he rejoined Korea Telecom as a Senior Member of Technical Staff and the Director of the Networking Research Team in the Telecommunications Network Laboratory until August 1999. Since September 1999, he has worked as a Professor of the School of Electronics and Information, Kyung Hee University. His research interests include the service and network management architecture for a distributed processing environment, fault-tolerant management for distributed components and next generation Internet protocols. He is a member of IEEE, IEICE, IPSJ, KICS, KISS and KIPS.

Ki-Woon Yim received his BS and MS degrees in electronic engineering from Kyung Hee University, Yongin, Korea, in 2000 and 2002. His research interests include mobile IP, cellular IP and mobile multicast. He is a member of KICS and KISS.

Dae-Young Lee received his BS degree in physics at Seoul National University, Korea, in 1964. In 1971, he received his MS degree from the Department of Computer Science, California State University at Los Angeles, USA, and his PhD degree from the Department of Electronic Engineering at Yonsei University, Korea, in 1979. Since 1971, he has been working as a Professor in the School of Electronics and Information, Kyung Hee University. He was President of the Graduate School of Technology and Information Science at Kyung Hee University from 1990 to 1993 and Chairman of KICS from 1999 to 2000. His research interests include data compression, image processing, mobile IP, network security, and traffic control in ATM networks. He is a member of IEEE, KICS, KIPS, and KISS.

Dong-Sik Yun received his BS in electrical and electronic engineering from Hankook Aviation College and his MS in electrical and electronic engineering from Korea Advanced Institute Science and Technology (KAIST), Korea, in 1986 and 1988. In 1983 he joined Korea Telecom, where he worked on N-ISDN and B-ISDN field trial projects. From 1993 to 1994, he researched TINA-C service management architecture as a visiting researcher of core-team member at NJ. US. From 1996 to 1997, he was involved in the project of development of KT's next generation service and network management system covering the DAVIC (Digital Audio Video Integration Council) VoD (Video on Demand) service control system and the ATM network management system. From 1998 to 2001, he was Director in the development of KT's broadband multi-service (ATM, FR, and IP) service/network control and management system. He is now Director and Coordinator of the project for design and development of KT's integrated service/network management system. His research interests include ATM, ADSL and IP services and network management architecture in a distributed processing environment, fault-tolerant management for distributed components in the inter-domain and heterogeneous networks, Wireless Internet Access Technology, Active Network Management, Service Level Agreement (SLA), and Policy-based Network Management.