Int. Computer Symposium, Dec. 15-17, 2004, Taipei, Taiwan.

FAULT TOLERANCE FOR HOME AGENTS IN MOBILE IP

Yin-Fu Huang and Min-Hsiu Chuang

Institute of Electronic and Information Engineering
National Yunlin University of Science and Technology
Touliu, Yunlin, Taiwan 640, R.O.C.
Email: huangyf@el.yuntech.edu.tw

ABSTRACT

In the conventional IP protocol, whenever a mobile device moves to a different network, it must change its IP address to communicate with other nodes in the Internet. Mobile IP protocol keeps mobile nodes online without changing theirs IP addresses while changing the attachment points. The packets destined to MNs are relayed by their HAs. However, once the only one HA fails, all MNs managed by the HA will not receive packets normally. In the paper, we propose a novel protocol with multiple MAs where only double mobility bindings are maintained in the whole system. When an HA is failed, its backup HA can take over it in a short time without fetching the bindings from other places. Besides, we also consider the load balancing between these HAs during HA takeover and recovery. Through the simulation, we observe that our method has less registration overheads, better MN-scalability and less sensitivity on MN mobility than others.

1 INTRODUCTION

Due to the development of the wireless technology, many personal information products such as laptops, personal digital assistants (PDA), and cell phones are equipped with a wireless communication interface, thereby bringing the convenience for people. However it has some problems when the current TCP/IP protocol [8, 9] works on these portable devices, since the TCP/IP protocol was designed under the assumption that the end-points are stationary. When a mobile node (MN) moves to another network without changing its IP address, it will not receive the packets destined to it. These packets still route to the home network of the MN, but not to the current attachment point. Thus, the work group of IETF (Internet Engineering Task Force) develops the Mobile IP protocol [5, 6, 10] to overcome the problems.

In a single MA system, it will face to the challenges such as efficiency and robustness when the supervised MN number increases dramatically. Thus some methods with multiple MAs were proposed to solve this problem [1, 2, 4], and even concerned about the load balancing between these MAs [11]. In [4], each HA in the home network must maintain mobility bindings of all MNs registered with the network, even if it only manages a portion of these MNs. In [1], there exists a stable storage to keep all mobility bindings in the network. However, the stable storage forms a single point failure. To solve the problems above, we propose a novel protocol with multiple MAs where only double mobility bindings are maintained in the whole system; i.e., only one backup for an MA. When an HA is failed, its backup HA can take over it in a short time without fetching the bindings from other places. Besides, we also consider the load balancing between these MAs. We always select the lightest loading HA as a new backup when the old backup HA takes over the failed HA. In summary, our method has less registration overheads, better MN-scalability, and less sensitivity on MN mobility than others.

The remainder of the paper is organized as follows. In Section 2, we briefly introduce the concepts of Mobile IP. In addition, the system model with assumptions and problem definitions is described in the section. Then the fault-tolerant protocol involving failure detection, HA takeover, and backup selection is proposed in Section 3. In Section 4, we describe HA recovery when an HA comes up from crash. A simulation model is presented to evaluate the system performance in Section 5. Finally, we make conclusions in Section 6.

2 SYSTEM MODEL

2.1 Mobile IP

A mobile IP protocol consists of the following components: 1) mobile nodes (MN), 2) home agents (HA), 3) foreign agents (FA), and 4) correspondent nodes (CN). Basic mobile IP operations are shown in Fig. 1. A mobility agent (HA or FA) transmits agent advertisements periodically to advertise its services on a link. Mobile nodes use these advertisements to determine their current point of attachment to the Internet. When an MN stays in a home network, it can receive and send packets according to the conventional IP protocol, just like the other stationary nodes in the network. Whenever the MN moves away from the home network into a foreign network, the MN will obtain a care-of-address (COA) from the foreign network. There are two ways to obtain a COA. First, if the MN finds an FA in the foreign network, it can register with the FA and acquire a care-of-address using the agent discovery protocol. Second, if the MN finds no FA, it can obtain a collocated care-of-address assigned by using the DHCP protocol [3]. After getting a COA, the MN must register with its HA. If the HA accepts its registration, the HA will update the mapping between the home address and the newest COA of the MN, called mobility binding, and then send a registration reply to the MN.

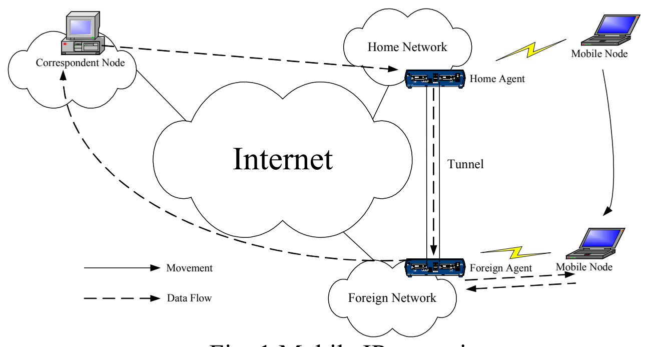


Fig. 1 Mobile IP operations

2.2 The Assumptions and Problem Definitions

When the number of MNs registered with the home network increases, the workload of managing and relaying packets will fall on an HA. Once the only one HA fails, all MNs managed by the HA will not receive packets normally. Therefore, multiple mobility agents (MA) can be used to deal with this problem. In our model, we have some assumptions outlined as follows: 1) not more than one MA of a home network will fail at the same time, 2) whenever an MA fails, any data in the volatile media of the MA will be gone, and can not be restored anymore, 3) the control signals used in the proposed protocol will be reliable in the network, and 4) when an MA recovers from the failed status, no MA fails during the recovery.

Here for a home network with *n* MAs, our goal is that if the MA memory is large enough, we can still service all MNs registered with the home network, even when *n-1* MAs are failed. Besides, in order to save the memory usage, we only need to maintain double mobility bindings in the whole system; i.e., only one backup for an MA. The protocol proposed here is transparent to MNs such that MNs are not aware of any MA failures, and no upgraded software should be installed within them.

3 FAULT-TOLERANT PROTOCOL

3.1 HA Table and Backup Table

In order to still service all MNs registered with a home network even when more than one HA is failed, we must maintain double mobility bindings in the whole system; i.e., one backup for an HA. A logical system framework is illustrated as Fig. 2. A bi-directed graph is used to show the takeover and backup relationships between HAs. Each node represents a physical HA in the home network. The underlined number inside a node indicates that it is acting as a supervising HA (also a logical HA), whereas the other numbers inside the node represent logical HAs that the physical HA is taking over. Besides, the numbers labeled outside a node represent logical HAs that the physical HA backs up. Initially, the logical system framework as depicted in Fig. 2 shows that only one logical HA acts on a physical HA, and it is also a supervising HA. Besides, each physical HA also backs up mobility bindings of the MNs registered with another logical HA. However, after several alternate failures and recoveries, a physical HA might take over or back up more than one logical HAs at the same time. To maintain the takeover and backup relationships between HAs, two tables called HA table and Backup table are used in each physical HA. HA table records 1) the logical HAs (including the supervising HA) being taken over by the physical HA, and 2) their backup locations. Besides, Backup table records the backup HAs, thereby enabling the physical HA to take over them when the logical HAs fail.

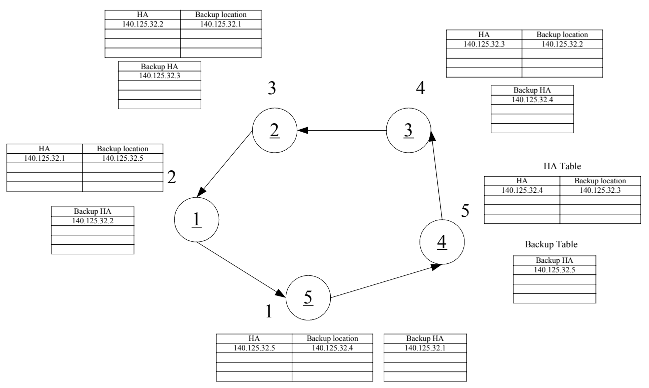


Fig. 2 Logical system framework

3.2 Failure Detection and HA Takeover

During the system operation, each physical HA in the home network must monitor whether all related logical HAs are alive or not. These related logical HAs includes the supervising HAs where the logical HAs being taken over by the physical HA are backed up (i.e., "Backup location" of HA table), and the logical HAs of which bindings are backed up here (i.e., "Backup HA" of Backup table). Here agent advertisement messages sent by logical HAs are used to check whether these logical HAs are alive or not. If a physical HA receives an agent advertisement from a logical HA persistently, the logical HA can be considered alive; otherwise, it could be failed.

In a physical HA, each logical HA monitored here has a timer ***Failure_timer***. Whenever receiving an advertisement message from a logical HA, the physical HA would call procedure ***Adv_monitor( )*** to reset the corresponding ***Failure_timer*** with value ***Max_failure_time***. Besides, each physical HA would call procedure ***Failure_detect( )*** periodically to decrease all the timers by one. If the timer of one logical HA expires, the HA is considered failed. If the failed HA has the backups of the logical HAs being taken over by the physical HA (i.e., "Backup location" of HA table), new backups should be found for the takeover HAs. On the other hand, if the backup of the failed HA is in the physical HA (i.e., "Backup HA" of Backup table), the physical HA will take over the failed HA. Then, it also needs to find a new backup for the failed HA. Procedure ***Adv_monitor( )*** and ***Failure_detect( )*** are given as follows:

```
Procedure Adv_monitor(i){
   Failure_timer[i]←Max_failure_time;
}// Adv_monitor end

Procedure Failure_detect( ){
   For all monitored HA_i
      Failure_timer[i]←Failure_timer[i]-1;
   For all monitored HA_j with timer=0
      If (HA_j exists in "Backup location" of HA table)
         /* Find new backups for all corresponding
            "HA"s of HA table */
         For all HA_k in "HA" of HA table
            Select_backup(k);
      Else Takeover(j);
         /* Find a new backup for "Backup HA" of
            Backup table */
         Select_backup(j);
}// Failure_detect end
```

If a physical HA takes over a failed HA, it must perform **Gratuitous ARP** mapping the IP address of the failed HA to its physical address [7], and then fills the IP address into its HA table. Afterward, the failed HA can be operational on the physical HA, including sending agent advertisements for the failed HA and forwarding packets to the MNs registered with the failed HA. Procedure ***Takeover( )*** is given as follows:

```
Procedure Takeover(j){
   Perform Gratuitous ARP for HA_j;
   Add HA_j into 'HA' of HA table;
   Send agent advertisements for HA_j;
   Forward packets to the MNs registered with HA_j;
}// Takeover end
```

3.3 Selecting the Backup HA

After the failure of a physical HA, new backups should be found for the takeover HAs or for the failed HA, as mentioned in Section 3.2. Each backup action must be performed individually. All physical HAs alive could be the backup candidates as long as their remaining memory is large enough to back up the HA. The physical HA selected here should be the one with the lightest loading in the home network. What we call the loading is based on the following factors ordered by their priorities, such as 1) the numbers of logical HAs backed up at the physical HA, 2) the number of logical HAs acting on the physical HA, 3) the number of the MNs which from other home networks, now are being served by the physical HA, and 4) the remaining memory size at the physical HA. If more than one new backup should be found on a physical HA, the "Backup" message for the takeover HA requiring more memory is issued before the one requiring less memory, since the former has less chances to find a new backup than the latter. Procedure ***Select_backup( )*** is described as follows:

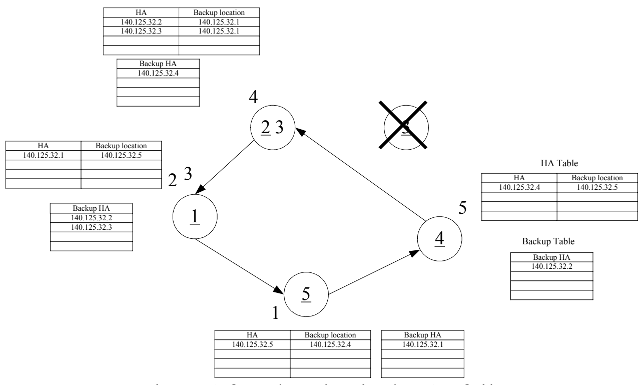
```
Procedure Select_backup(j){
/* Each logical HA has a Tbk and a Backup_ACK_count */
   Tbk←0;
   Backup_ACK_count←0;
   Send Backup messages to all physical HAs in the home
   network;
   Wait Backup_ACK messages for a period of time Tbk;
   Tbk←1;
   If (Backup_ACK_count=0)
      Fill "fail" into "Backup location" of HA table;
      Keep on waiting for other Backup_ACK messages;
/* Receiving multiple Backup_ACK messages during Tbk */
   Else Compare the information from these Backup_ACK
        messages and choose one physical HA_k with the
        lightest loading;
        Send Confirm message to the physical HA_k;
        Send Release messages to the other physical HAs;
        Occupied_flag←1;
        Fill HA_k into "Backup location" of HA table;
        Transmit bindings of the MNs registered with HA_j to
        the physical HA_k;
        Occupied_flag←0;
}// Select_backup end
```

In order to avoid that more than one logical HA selects the same physical HA as the backup at the same time, a semaphore is used to synchronize their backup actions. A physical HA can reply to a ***Backup*** message only when it can satisfy the memory requirement of the ***Backup*** message and is not being "occupied" by other ***Backup*** message. The synchronization among the physical HAs is accomplished with message passing. Whenever a physical HA receives a message, it would call procedure ***Message_handler( )*** to execute the corresponding action using multi-thread techniques. As mentioned, since a physical HA can process only one ***Backup*** message at a time, a semaphore is used to synchronize these message-trigger threads. Procedure ***Message_handler( )*** is given as follows:

```
Procedure Message_handler(msg){
   /* Occupied_flag: global flag for a physical HA,
      S: semaphore initialized to 1 */
   Switch(msg){
   Case Backup
      wait(S);
      Occupied_flag←1;
      /* Back_ACK contains loading information */
      Reply Backup_ACK message;
   break;

   Case Backup_ACK
      If (Tbk=0)
         /* Receiving ACK before timeout */
```

```
            Record the loading information in the message;
            Backup_ACK_count ← Backup_ACK_count+1;
        Else /* Receiving ACK after timeout */
            If (the corresponding "Backup location" of HA
               table="fail")
                /* Choose the first sender as the backup */
                Reply Confirm message;
                Occupied_flag ← 1;
                Fill the sender into "Backup location" of
                HA table;
                Transmit bindings of the MNs registered
                with the takeover HA to the sender;
                Occupied_flag ← 0;
            Else Reply Release message;
    break;

    Case Confirm
        Occupied_flag ← 0;
        signal(S);
        Fill the sender into "Backup HA" of Backup table;
        Receive bindings of the MNs registered with the
        sender;
    break;

    Case Release
        Occupied_flag ← 0;
        signal(S);
      break;

    Case Balance
        Occupied_flag ← 1;
        /* Balance_ACK contains loading information */
        Reply Balance_ACK message;
    break;

    Case Balance_ACK
       Record the loading information in the message;
    break;
    }// Switch end
}// Message_handler end
```

Here the example as shown in Fig. 2 is used to explain the HA takeover and backup. When the physical HA_3 fails, the physical HA_2 will take over HA_3 and also need to find a new backup for HA_3. Besides, since the physical HA_3 backs up HA_4, the physical HA_4 must find a new backup for HA_4. After the physical HA_2 selects the physical HA_1 as a new backup for HA_3 and the physical HA_4 selects the physical HA_2 as a new backup for HA_4, the statuses become as shown in Fig. 3.



HA	Backup location
140.125.32.2	140.125.32.1
140.125.32.3	140.125.32.1

Backup HA
140.125.32.4

HA	Backup location
140.125.32.1	140.125.32.5

Backup HA
140.125.32.2
140.125.32.3

HA	Backup location
140.125.32.4	140.125.32.5

Backup HA
140.125.32.2

HA	Backup location
140.125.32.5	140.125.32.4

Backup HA
140.125.32.1

Fig. 3 After the physical HA_3 fails

4 HA RECOVERY

When a physical HA comes up from the failed state, it immediately listens to whether any agent advertisement on the link contains its IP address. If none (i.e., the physical HA starts up for the first time), it performs **Gratuitous ARP** mapping the IP address to its physical address, and start functioning with no mobility bindings. On the contrary, in addition to perform **Gratuitous ARP**, it must restore the mobility bindings accumulated by the physical HA that took over it when it failed. Procedure ***Recovery( )*** is given as follows:

```
Procedure Recovery(j){
  If (no agent advertisement on the link, sent by $HA_k$,
     contains the IP address of $HA_j$)
      Perform Gratuitous ARP for $HA_j$;
      Add $HA_j$ into 'HA' of HA table;
      Send agent advertisements for $HA_j$;
      Choose the precedent physical $HA_i$ as the backup of
      $HA_j$;
      Fill $HA_i$ into "Backup location" of HA table;
      Transmit bindings of the MNs registered with $HA_j$ to
      the physical $HA_i$;
  Else Perform Gratuitous ARP for $HA_j$;
      Add $HA_j$ into 'HA' of HA table;
      Send agent advertisements for $HA_j$;
      Get backup location and mobility bindings of $HA_j$
      from $HA_k$;
      Fill the backup location into "Backup location" of HA
      table;
      Forward packets to the MNs registered with $HA_j$;
  Balance( );
}// Recovery end
```

After a physical HA recovers from crash, the HA loading in the home network, such as the number of takeover HAs and backup HAs, will be balanced. The recovered physical HA is responsible for issuing balance signals to all physical HAs in the home network. After gathering the loading information from other physical HAs, the recovered HA can start the deducing process to reassign the takeover HAs and backup HAs in the home network.

Procedure ***Balance( )*** is omitted here.

5 PERFORMANCE EVALUATIONS

5.1 Simulation Model

The simulation model is illustrated in Fig. 4. Five packet generators are used to generate MN_pk, MN_re_reg, MN_mv_reg, FMN_pk, and FMN_reg packets, respectively. These packets are submitted to waiting queues of MAs in the home network, and their flow paths are described in Fig. 5. Each MA would process different packet types with different processing costs and flows. The simulation was done using GPSS World developed by Minuteman Software, Inc.

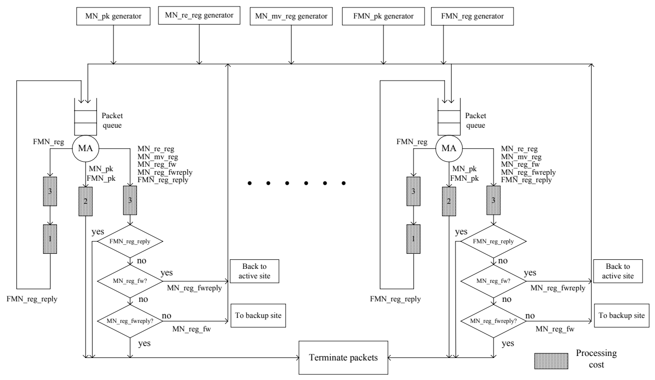
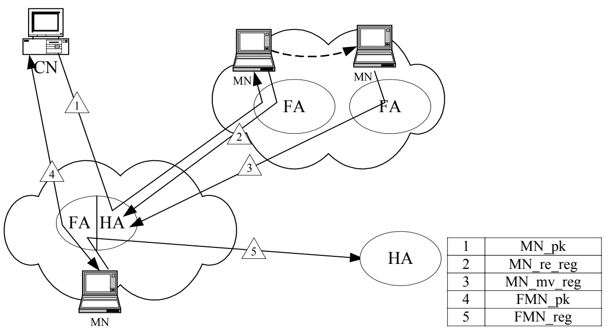


Fig. 4 Simulation model



1	MN_pk
2	MN_re_reg
3	MN_mv_reg
4	FMN_pk
5	FMN_reg

Fig. 5 Flow paths of generated packets

5.2 Experimental Results

Experiment 1: overheads of different MN numbers

In the experiment, we observe the overheads of our method and FTMIP [4]. Those include registration delay, extra registration messages (i.e., registration forwarding and registration forwarding reply), and average numbers of bindings maintained per HA. The registration delay is the interval between the time an HA receives a registration request and the time it finishes processing the corresponding reply from its backup site. As shown in Fig. 6(a) and Fig. 6(b), our method has less registration delay and less extra registration messages than FTMIP. The reason is that, in FTMIP, each HA maintains all bindings of MNs registered with other peers, and thus an HA must wait all other peers to complete binding synchronization when it receives a registration request from its MN. The situation would be worse especially when there are more MNs registered in the network. Besides, we also plot the binding numbers maintained per HA in both methods, as shown in Fig. 6(c).

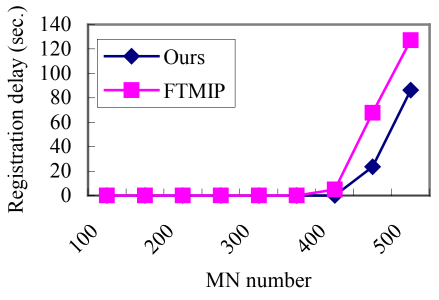


Fig. 6(a) Registration delay of different MN numbers

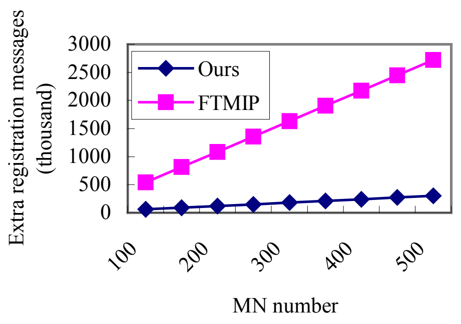


Fig. 6(b) Extra registration messages of different MN numbers

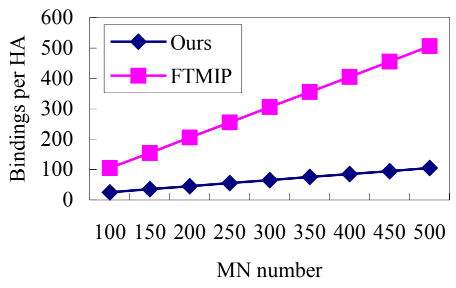


Fig. 6(c) Bindings per HA of different MN numbers

Experiment 2: overheads of different mobility rates

In the experiment, we observe the registration delay and registration forwarding number for total 300 MNs with different mobility rates. As shown in Fig. 7(a) and Fig. 7(b), we found that our method has smoother overheads than FTMIP, regardless of the registration delay and registration forwarding number, when considering different mobility rates.

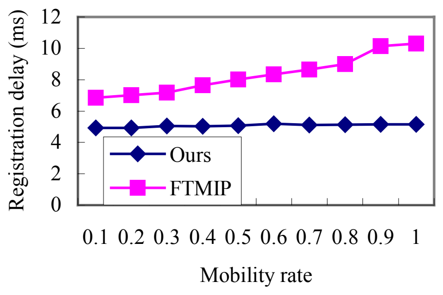


Fig. 7(a) Registration delay of different mobility rates

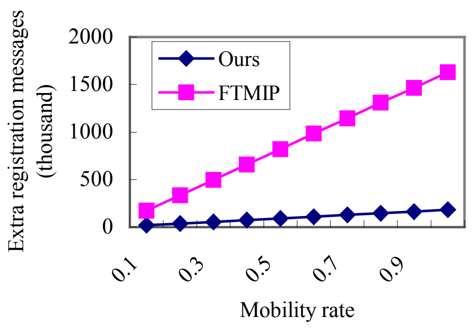


Fig. 7(b) Extra registration messages of different mobility rates

Experiment 3: effects with/without doing balance during HA recovery

In the experiment, after eight HAs are failed, we make the HAs recover from failure one by one in order to observe the effect on the system performance with/without doing balance during HA recovery. As shown in Fig. 8, we found that the registration delay without doing balance are obviously much longer than that with doing balance unless the number of alive HAs is more than half of the total HA number.

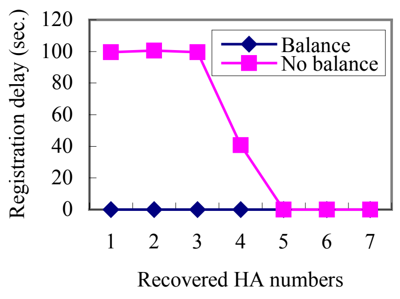


Fig. 8 Registration delay of different recovered HA numbers

6 CONCLUSIONS

In this paper, we point out some drawbacks of previous fault tolerant protocols for Mobile IP, such as longer registration delay, maintaining all bindings of MNs in each HA, and a single point failure or even a performance bottleneck based on centralized managements. Here we propose a novel distributed protocol only maintaining double mobility bindings in the whole system. Our method issues less extra registration messages in order to prevent long registration delay. Besides, we also consider the load balancing during HA takeover and recovery to make the system performance more efficient. Through the experiments, we found that our method has less registration overheads, better MN-scalability, and less sensitivity on MN mobility than others.

REFERENCES

[1] JinHo Ahn and ChongSun Hwang, “Efficient fault-tolerant protocol for mobility agents in mobile IP,” Proc. 15th International Conference on Parallel and Distributed Processing Symposium, 2001, pp. 1273-1280.

[2] B. Chambless and J. Binkley, “HARP – home agent redundancy protocol,” IETF Draft, 1997.

[3] R. Droms, “Dynamic host configuration protocol,” IETF RFC 1541, 1993.

[4] R. Ghosh and G. Varghese, “Fault-tolerant mobile IP,” Technical Report WUCS-98-11, Washington University, 1998.

[5] C. E. Perkins, “IP mobility support,” IETF RFC 2002, 1996.

[6] C. E. Perkins, *Mobile IP: Design Principles and Practices*, Addison-Wesley Longman, Reading, Mass., 1998.

[7] D. C. Plummer, “An Ethernet address resolution protocol-or-converting Network protocol address to 48 bit Ethernet address for transmission on Ethernet hardware,” IETF RFC 826, 1982.

[8] J. Postel, “Internet protocol,” IETF RFC 791, 1981.

[9] J. Postel, “Transmission control protocol,” IETF RFC 793, 1981.

[10] J. D. Solomon, *Mobile IP: The Internet Unplugged*, Prentice-Hall, Upper Saddle River, NJ, 1998.

[11] A. Vasilache, Jie Li, and H. Kameda, “Load balancing policies for multiple home agents mobile IP networks,” Proc. 2nd International Conference on Web Information Systems Engineering, 2001, pp. 178 –185.