American Journal of Applied Sciences 8 (12): 1388-1397, 2011
ISSN 1546-9239

Performance Analysis of a Highly Available Home Agent in Mobile Networks

[1]Abdelgadir Tageldin Abdelgadir, [2]Mohiuddin Ahmed,
[3]Al-Sakib Khan Pathan, [1]Mohd Ariff Abdullah and [1]Shariq Haseeb
[1]Communication Networks and Solutions Lab,
Wireless Communication Cluster, MIMOS Berhad,
Technology Park Malaysia, 57000 Kuala Lumpur, Malaysia
[2]Department of Information Systems,
Jazan University, Kingdom of Saudi Arabia
[3]Department of Computer Science,
International Islamic University Malaysia,
Jalan Gombak, 53100 Kuala Lumpur, Malaysia

Abstract: Problem statement: Network Mobility as a service is provided by the NEMO protocol in IPv6 environments. NEMO is an extension to MIPv6 and thus inherits the same reliability problems of MIPv6. MIPv6 is not reliable because the Home Agent (HA) is a single point of failure. In order to provide real-time services for MIPv6 networks, reliability should be considered as part of any high availability solution used to deploy Mobile IPv6 networks. **Approach:** Many approaches have been taken to solve the problem of HA as a single point of failure. In our proposed solution, failure detection and recovery is handled by the home agent. Therefore, recovery is transparent to the mobile network. **Results:** In this study we opted for using HA redundancy to provide a highly available home agent solution which achieves recovery times suitable for real-time applications. **Conclusion:** The results show that achieving high availability in IPv6 based mobile networks which support NEMO is possible.

Key words: Home agent, real time, high availability, Network Mobility Basic Support (NEMO), MIPv6, Dynamic Home Agent Address Discovery (DHAAD), Virtual Home Agent Reliability Protocol (VHARP), Operating System (OS)

INTRODUCTION

With the rapid growth of the internet and rapid deployment of wireless networks, mobility is becoming an important and required feature as part of the current internet services. Since the introduction of the IPv6 protocol, mobility has been integrated into IPv6 through MIPv6. MIPv6 was standardized by IETF RFC 3775; it was designed to enable mobility for nodes on a network. MIPv6 was intended to allow a Mobile Node (MN) to continuously maintain its ongoing sessions while it changes location between different networks. While MN is at its home network, it functions as a fixed node. When the movement of MN occurs, MN will need to send a binding update to the HA in order to have a tunnel created for communication purposes (Kusin and Zakaria, 2011). When a packet intended for MN reaches the home network of MN, packets will be intercepted by HA and then sent to MN on its new address. It is observed that in this scenario, HA becomes a single point of failure since packets intended for MN when on a different network will be lost if HA is not available for interception. MIPv6 introduced another problem in which many nodes try to connect at the same time when roaming from one network to another, causing what is known as a "binding update storm" (Tazaki *et al*., 2011).

As a solution to the inefficiency of MIPv6, IETF defined a new protocol called Network Mobility Basic Support (NEMO BS) in RFC 3963. The name is derived from Network Mobility and will be referred to by (NEMO) in the rest of this document.

Corresponding Author: Abdelgadir Tageldin Abdelgadir, Communication Networks and Solutions Lab, Wireless Communication Cluster, MIMOS Berhad, Technology Park Malaysia, 57000 Kuala Lumpur, Malaysia

NEMO is designed to allow a mobile network, which consists of a Mobile Router (MR) and Mobile Network Nodes (MNNs) connected to the MR, to continuously maintain its ongoing sessions while roaming from a network to another. In order to achieve such mobility, each MR is identified by its home address, regardless of its location on the internet. While MR is at its home network, it operates like a fixed node by using its home address to communicate with other nodes on the internet. When the mobile network moves to a foreign network, it will inform it's HA of its new address by sending a binding update back to HA. HA should be aware of the Mobile Network Prefixes (MNPs) associated with the MR when there are more than a single network. This information is exchanged with the binding update or it can be preconfigured according to the mode used. There are two modes of operation in the case of MNPs configuration; these are implicit mode and explicit mode. In implicit mode, the information related to MNPs is known to HA, while in explicit mode the information related to MNPs is sent in the binding update. When a mobile network is away from its home network, packets intended to the MNNs are intercepted by HA and then tunneled to MR. It is observed that in this scenario, similar to MIPv6, HA becomes a single point of failure since packets intended for MR when its on a different network will be lost if HA is not available for interception (Abdullah *et al*., 2010).

From the scenarios mentioned above we can conclude that if a failure occurs in HA, MN and MR will not be able to perform a binding update with HA, therefore a tunnel will not be created and communication with MN or MR will not be possible because packets required to pass through HA will be lost. This problem concerns the reliability of MIPv6 and NEMO because HA is a single point of failure in both architectures. The issue of HA reliability has been discussed in (Wakikawa, 2009; McCarthy *et al*., 2009; Zhenkai *et al*., 2011; Ayaz *et al*., 2009; Gay *et al*., 2009; Rathi and Thanuskodi, 2009; Lin and Yang, 2009). Currently, there is no standard RFC or protocol to solve the HA reliability issue in MIPv6 and NEMO, the most recent work in progress can be found in (Wakikawa, 2009).

The mentioned solutions provide HA reliability (Bassek *et al*., 2006) through various ways such as HA redundancy (Wakikawa, 2009; Gay *et al*., 2009; Rathi and Thanuskodi, 2009; Lin and Yang, 2009) Distributed HAs in (McCarthy *et al*., 2009; Zhenkai *et al*., 2011 Ayaz *et al*., 2009) or MN initiated recovery by using mechanisms such as "Home Address Regerneration" in (Lin and Yang, 2009). The time required for the recovery of HA varies as well between different solutions. Wakikawa (2009) and Rathi and Thanuskodi (2009) the recovery time for a HA is reported to be in the range of 3 to 4 seconds, while in (Gay *et al*., 2009) as a partial implementation of (Wakikawa, 2009), the recorded time was 0.7 seconds for transparent mode not involving MN and 2 seconds for a non-transparent mechanism involving MN. Other methods such as those mentioned in (McCarthy *et al*., 2009; Zhenkai *et al*., 2011; Ayaz *et al*., 2009) have not provided figures of HA recovery time.

Frantti and Koivula (2011), it is mentioned that for real-time application such as VoIP and video conferencing, the quality of IP mobility services is affected by packet loss, delay and jitter. It is also mentioned that packet loss above 1% and latencies over 100 m sec can cause distraction to users, so they should be avoided. Therefore, HA recovery time is crucial for real-time applications in environments such as NEMO where HA is not just a single point of failure but a main arbitrator affecting any communication between the mobile network and other nodes in the internet. In order to have seamless mobility, solving the problem of HA reliability should be achieved without causing more delays in the communication path between the mobile network and the correspondent nodes.

In this study, we present a highly available HA reliability solution. We also analyze performance factors affecting HA recovery in order to introduce the least amount of delay to the operation of a network providing real-time services compared to other approaches found in the reviewed literature.

NEMO: NEMO is used as a protocol to handle network mobility. Examples of networks that move include personal vehicular networks, networks on transportation systems and Personal Area Networks (PANs). The benefits of these services can be derived from considering the entire network of nodes as a single unit (Georgopoulos *et al*., 2011). For MR to operate, it must have a home network. A home network hosts the original point of attachment for MR. The prefix used by the mobile network behind the MR is a Mobile Network Prefix (MNP). This prefix is used by the mobile network and is administered as a local network having MR as the gateway. The foreign network in NEMO is any other network which is not a home network. When the MR roams and attaches to a foreign network, it acquires a Care-of-Address (CoA) needed by HA to know the location of MR. A possible way to acquire CoA is through stateless IPv6 configuration or any other method provisioned at the foreign network.

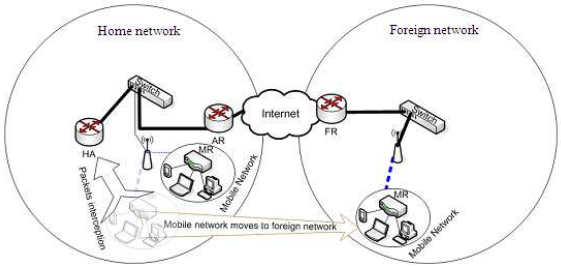

Fig. 1: Interception of packets destined to the mobile network when it moves to another network

These changes in location are exchanged with the HA through the Binding Update (BU). Other information in the BU of NEMO includes the MNPs and an R flag set to define an operating MR. After this, HA replies with a binding update acknowledgment (BU-ACK) if the binding is accepted by HA. After the completion of this procedure a bidirectional tunnel is created between MR and HA to tunnel packets between these two components. The binding update information is stored in a binding cache which is used to map between its Home Address (HoA) and the CoA of MR. When HA intercepts packets destined to the mobile network, it will tunnel the packets to MR and MR will route the packets accordingly.

In Fig. 1 we observe that when packets reach a mobile network which has moved to a foreign network, HA will be responsible for tunneling these packets to the mobile router. A failure in HA will result in losing all those packets. Thus from this architecture, HA is a single point of failure.

HA failure detection and recoverability: Availability of a HA is crucial as previously mentioned. In order to avoid total communication loss, HA failure must be detected and recovered as soon as it happens. This study does not cover the failure of other routers in the network as it's another topic covered in more details in (Suchara *et al*., 2011; Wu and Dong, 2009).

In normal operation of NEMO, MR can only detect the failure of an HA when it tries to refresh the binding or when it moves to a different network and thus sends another BU. Communication is lost between the mobile network nodes and any other nodes during this process. As MR tries to recover connectivity, it attempts to find another HA using a Dynamic Home Agent Address Discovery (DHAAD) message sent to the home network, if another HA exist to serve MR, then a binding will be created with the new HA and communication resumes after the tunnel is created in a similar way as previously mentioned.

The process of using (DHAAD) introduces a total breakdown in communication between the mobile network nodes and other nodes on the internet while it tries to restore connection, thus it's not suitable for real-time applications requiring a continuous and stable communication link. Many approaches have been proposed to overcome the problem of HA failure. We discuss some of these approaches.

Related work on HA reliability: Many reliability protocols have been proposed to solve the problem of HA reliability in (Wakikawa, 2009; McCarthy *et al*., 2009; Zhenkai *et al*., 2011; Ayaz *et al*., 2009; Gay *et al*., 2009; Rathi and Thanuskodi, 2009; Lin and Yang, 2009). Currently, there is no standard RFC or protocol to solve the HA reliability problem in NEMO, the most recent work in progress can be found in (Wakikawa, 2009). MIPv6 and NEMO do not handle HA reliability issues, thus proposed solutions are either extensions to the main protocol or newly proposed approaches. The problem statement draft for HA reliability in (Wakikawa, 2009) influenced many recent protocols by researchers.

The protocol presented in (Wakikawa, 2009) inspired the basic idea of having redundancy as a solution to the HA reliability problem. This approach lacked support for IPv6 based mobility and thus cannot be used as a solution in an IPv6 environment. Another proposed protocol which introduced the concept of virtual HA addresses to the reliability problem in an IPv6 environment is found in (Wakikawa, 2009) and known as Virtual Home Agent Reliability Protocol (VHARP), where HAs in this set-up will share the same global address. VHARP introduces ideas which we opted to adapt such as the use of a virtual address, however a problem with the protocol is that it does not provide an optimal mechanism defined for HA failure detection and the protocol uses modified router advertisements, which is a slow process since these messages are sent within certain intervals suitable for router advertisements, usually between 1 and 3 second according to the implementation used. Another method using distributed HAs to achieve global HA reliability is found in (McCarthy *et al*., 2009), the global scope introduces new issues such as the use of anycast routing on a global scale (Khare *et al*., 2010) as well as overheads due to the synchronization messages involved. Research work based on this protocol is found in (Ayaz *et al*., 2009) it describes that in a global HA to HA network, each HA can be either a primary HA (priHA) or a proxy HA (proHA) for an MR. Each MR has a Home Address (HoA) configured either from the home network prefix or from its MNP depending on

the physical deployment. In a scenario where a packet sent from any CN to MR whose primary HA is a HAP, the packet is first routed to the topologically closest HA. In case the closest HA that intercepted the packet is not HAP, it tunnels the packet to HAP over HA-HA tunnel and HAP decapsulates the packet and then tunnels it to MR via MR-HA tunnel. Final specifications for Global HA to HA are yet to be defined, except for an experimental draft recently published in (Wakikawa, 2009). Zhenkai *et al*. (2011) an implementation of multiple HAs mechanism in IPv6 environments is presented, where a method is introduced, which enables MN to simultaneously have multiple HAs when away from the home network. This method is not transparent to the MNs and introduces a substantial amount of overhead caused by message exchanges between the HAs and MNs. Wakikawa (2009) and Gay *et al*. (2009) the HA reliability protocol is discussed. This protocol introduced the concept of HA virtual switch which enables an HA switch between a main HA and a standby HA to happen without the knowledge of MN. This is possible through the use of a virtual IP address on both HAs. State synchronization must be done between an active HA and a standby HA using information contained in the binding cache of the active HA. Failure detection is done by HA, where the HA is required to continuously check for possible failure of an active HA. For this to happen, periodic "hello messages" are used to detect the failure of an active HA. This causes some traffic on the local link, although negligible but causes more network overheads in such systems (Hernandez-Cons *et al*., 2010). Rathi and Thanuskodi (2009), a Virtual Private Network based Home Agent Reliability Protocol (VHAHA) is presented as part of a framework which deals with the reliability problem in IPv6 environments. The framework deals with reliability by deploying an active HA which is an HA serving MNs. The active HA maintains the binding cache which stores the mobile bindings of all registered MNs. Other components are the backup HA and Inactive HA. The backup HA will provide services when the active HA is down, while the inactive HA's holds no mobility bindings, but provides limited services to backup HA services. The framework also introduces certain amount of overhead in the network to construct the Virtual Network and to provide reliability as reported in (Rathi and Thanuskodi, 2009). McCarthy *et al*. (2009) the proposed approach does not implement hardware redundancy; it is based on a procedure called "Home Address Regeneration" to tolerate HA failure. If an HA fails, the affected MNs cannot continuously perform binding updates to that failed HA; but instead opt to have MN resend a binding update to another HA known by MN through a modified binding update message. This process will introduce significant delay because it involves resending a binding update to a new HA in the case of a main HA failure. This approach is not transparent to MN, since the selection of failure-free HAs is initiated from the failure-affected MN.

MATERIALS AND METHODS

Proposed solution: The solution proposed adapts many features from the related work such as: HA redundancy using virtual HA address in (Wakikawa, 2009) and the home agent reliability protocol state synchronization techniques found in (Gay *et al.*, 2009), adds support for NEMO and provides high availability of HA with recovery time suitable for real time applications such as VoIP and Audio/Video streaming. In the proposed solution it is assumed that multiple HAs operate simultaneously, each HA will have a certain role. In this solution, we describe this as a "Home Agents Group". The Main HA in this solution will have the role of handling all mobility management tasks such as holding the binding cache and creating tunnels used for communication by the MR in a NEMO environment. The MR is only aware of the existence of the Main HA. The Main HA performs critical operations of handling the monitoring procedure as well as synchronizing the binding cache through fast synchronization messages which are triggered once a binding occurs. The Main HA is also responsible-through the MR-for multiple MNNs when operating in a NEMO network. The failure of the Main HA will result in the disconnection of all attached MNNs located throughout the Internet. In our proposed solution, the main components involved in a redundant HA scheme can be summarized in the following:

- Main HA: This is the HA which will serve MR in a NEMO environment
- Standby HA: In case of a main HA failure, a Standby HA takes over and becomes Active HA
- Multiple Home Agents Group (MHAG): A number of HAs, both Main and Stand-by operating as a single entity to serve MR, these appear as a single HA entity to MR
- Shared HA address: The HA address available to MR is used to achieve mode switching between HAs while preserving a single identity known by MR

Failure detection is done through a method of a simple and continuous signaling between the HAs in the HA group.

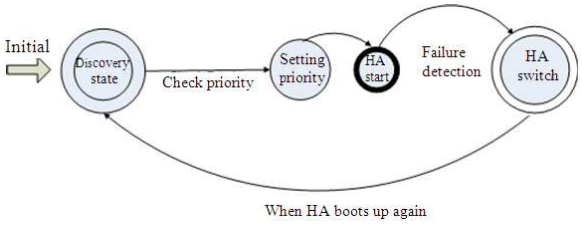

Fig. 2: HA State diagram

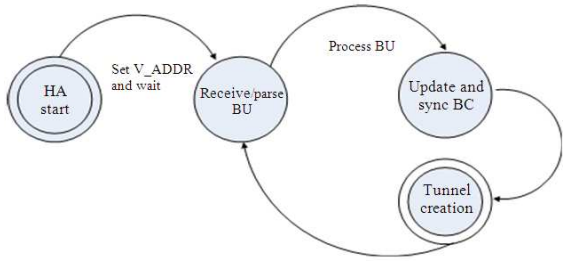

Fig. 3: HA Data flow diagram

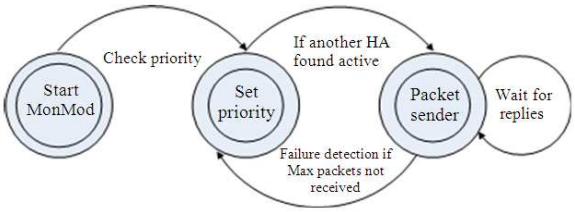

Fig. 4: Monitoring module state flow diagram

This is a simple procedure of sending a signaling packet from a standby HA to a main HA in order to assure its availability.

A reply to the message will determine the availability of the other HA, while a no-reply to a certain number of request signals is considered failure. The mechanism of using signaling to detect failure of a device is a common method in high availability systems to improve the availability of such systems. In this solution, an interval of 62 m sec between signaling packets is used to detect failure. The standby HA will be able to detect the failure of an active HA instantly and then proceed to initiate the operation of switching the mode of service and creating the required tunnels to assure that packets are routed correctly in case of an HA switch resulting from an HA failure. There are two types of switching between active and standby HAs, which can be executed:

- Soft switch: in this mode, after failure detection the HA switches its mode from active to stand-by and vice versa without involving the MR. This is transparent to the MR and depends only on the HA, thus we focus on this type of switching in this study
- Hard switch: another type of switching is a hard switch; in this mode the MR will be triggered to initiate a switch when a failure is detected

In Fig. 2 a state diagram illustrates the procedures in the proposed solution. From Fig. 2, HA initializes and process parameters and values set by the user, a state discovery operation in which HA starts to discover its own state is then started. In this discovery state operation, HA can be a main HA if no other main HA is configured, or it acts as a standby HA if another HA with a higher priority is already running. HA will need to set the priority according to the information gathered from the discovery state process and then start its role as a main HA or standby HA. If a failure is detected, another process will be triggered called “HA switch”, in this process a HA with the highest priority in the HA group will take over the role of main HA after going through the previous procedures such as discovering state and setting priority.

In Fig. 3, the HA data flow diagram depicting the start process is illustrated with some other sub-processes involved. When HA starts, it will set a virtual address known by the mobile node or router in order to be visible on the network as a main HA when required. In Fig. 3, it is assumed that HA has already reached an active state from its initial priority. At this point, HA is ready to receive binding updates from a MR in the case of NEMO. HA will then parse the binding update, reply with an acknowledgment if the binding is accepted, process the required information needed to create tunnels and then synchronize the information with the next HA in priority list if it is running.

In Fig. 4 the monitoring module (MonMod) will check the priority of its host HA as well as if any other HA is active. When a HA having the highest priority is started, signaling packets will only be sent if another HA existed in the group. If no other HA is found, MonMod will not generate signaling packets. When an HA with a lesser priority starts and finds a HA with a higher priority, the two HAs exchange signaling packets continuously to achieve high availability. A failure is detected when the lost packets becomes more than MaxPackets during packets exchange. In Fig. 5, we present an overview of the HA failure procedure. It reflects the implemented logical sequence used in the solution.

```
Procedure : HA operations and recovery
Begin procedure {
        State discovery based on initialization of:
                Node type
                Group
                Priority in the group
        Setting priority:
                Check configuration from static files;
                If ( another HA is found active && with lower priority )
                        Set state as main
                        Acquire virtual address
                        Start services and signaling requests
                If ( another HA is found active && with higher priority )
                        Sync binding cache
                        Start signaling replies
                else
                        Signaling is not started
        In Failure detection scenarios:
                Case of failed HA is detected do
                State Discovery
                Setting Priority
                Swap backup binding cache with active cache
                Stand-by HA acquires virtual address
        }
End of procedure
```

Fig. 5: HA failure procedure

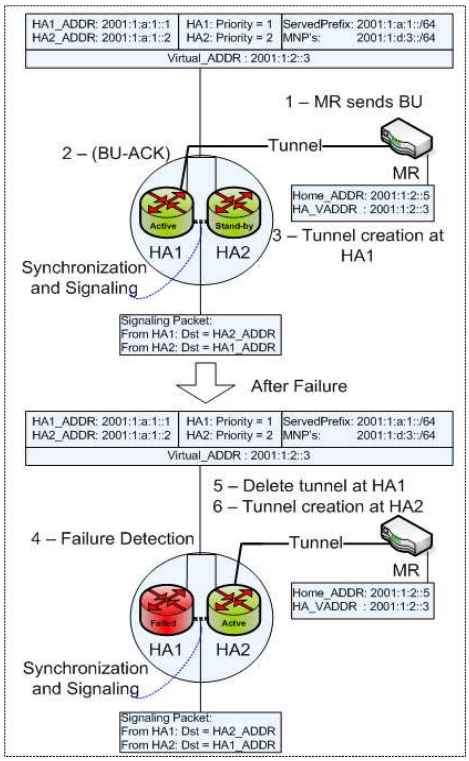

Fig. 6: Main functions of the solution: description of steps executed in the solution

In Fig. 6, a set of two home agents are used. HAs are assumed to be working and HA_1 is assigned an address such as in HA1_ADDR while HA_2 is assigned an HA2_ADDR. The Virtual_ADDR is assigned to both HAs, but only activated at the main HA having the highest priority at any point in time according to HA states. MR on the other hand have knowledge of HA's virtual address stored as HA_VADDR, this will enable MR to communicate with the HA having that address activated. The signaling packets mechanism running as a separate process is supported by the same stored address of HA1_ADDR and HA2_ADDR. The standby HA will monitor the availability of active HA acting as the main HA. Synchronization is done through update messages sent instantaneously when a binding update procedure completes successfully. The following is a summary of events that take place during the operation of mobility management:

- A BU is sent from MR to HA, the active HA will intercept BU
- A binding update acknowledgment (BU-ACK) is generated for MR from HA, a binding happens and the information is synchronized with the standby HA
- A tunnel is created between the active HA and the MR
- If the active HA is down for any reason and detection of failure occurs, the next HA in the list of HAs takes over the virtual address and creates a tunnel to MR. This takeover needs to be done in the most minimal time possible in order not to disrupt services
- The tunnel created earlier must be deleted when a failure occurs
- A new tunnel is created at HA_2 to serve MR through HA_2
- If the set-up is configured to have the failed HA to be active HA again, a similar takeover will take place once the failed HA is active again

By observing the previous points, we conclude that all events execute and complete while the communication between the mobile network and other nodes on the internet is ongoing. Thus, having a very fast recovery time for HA is a requirement for real-time applications which are expected to be met in this architecture. Theoretically, HA recovery time (R_T) is the sum of failure detection time (F_{dt}) added to destruction and establishment of tunnels time as well Eq. 1:

$$R_T = F_{dt} + MR_n (T_D + T_C) \quad (1)$$

Let R_T = Recovery time
Let F_{dt} = Failure detection time
Let T_D and T_C = The tunnel destruction and creation time
Let MR_n = The number of MR communicating in the network

Assuming that HA_1 is running and tunnels are created after successful binding with MR. Thus, from (1) when a failure occurs, F_{dt} should be considered as it increases with the addition of MR's in addition to the value of T_c at HA_2. The number of MRs also affects such system since according to the number of tunnels existing; recreating these tunnels with their routing tables at HA_2 after a failure will require an amount of time T_c in addition to F_{dt}. In our studies we observed a time of approximately 60ms for the process of tunnel creation and routing table manipulation, meaning at a rate of about 12 MR/s in our case. It is obvious from (1) that failure recovery time is proportional to the number of connected MRs in NEMO network. These numbers are rather affected by other variables such as processor power and available memory in the device used. The process of tunnel destruction at one HA and its creation at another HA will affect the solution when the number of MRs increases linearly. In a real test bed, this is related to the availability of computing resources at the site where an HA is being deployed. The process adds substantial milliseconds of time to the general solution and thus we conclude that in order to have a true solution suitable for real-time applications, the number of MRs must also be considered during deployment. The limitation of an HA solution is thus related to the number of MRs as has been observed in this research.

RESULTS

The performance study is focused on the recovery time resulting from an HA switch and its effect on packet loss, throughput and round trip time. In the experiments performed, the time taken for recovery is calculated using the mipv6tester utility and also various (Khare *et al*., 2010) analysis techniques were used in the study.

The same test-bed as in Fig. 7 was used in the experiments. In the experiments, we have tested different implementations based on NEPL on a "Ubuntu 8.04.1" Operating System (OS) running 2.6.24-5 version of the Linux kernel enabled with mobility support. MR, CN and FR are all running the same OS, each equipped with the required network interface card or wireless card needed. Some results are usually affected by wireless interferences and are not expected to have absolute accuracy. The components are connected through a gigabit connection but are using 100Mb/s since some components are limited to that speed.

Performance evaluation quantitatively compares the different factors affecting the performance in the test bed. In this study, parameters used for performance comparison include.

Packet loss caused by HA recovery time: packet loss is presented in relation to HA recovery in which an HA switch happens while packets are transferred. HA recovery time is the time needed for HA switch to happen and how it affects different streams on the network. The less it affects a stream sent, the better the results:

- Throughput: results show the number of packets being forwarded in the network while a failure occurs. Higher and a continuous throughput means better result
- Round Trip Time (RTT): this is a delay measurement on a network between the MNN and CN going through many components in between such as HA, FR.... We measure RTT during HA failure

For comparison purpose, UDP and TCP streams were used to compare the effect of HA recovery time on each stream. The tests were run on a test environment having MNN and CN at both ends. In Fig. 8-11 HA failure points are depicted on the diagrams by the dips created. Results from the packet loss experiments are compared to results obtained from (Lin and Yang, 2009). We can observe that the difference in packet loss is substantial for real-time applications as it reaches up to 65% for experiments done when running normal HARP having a delay time of 0.7 sec. On the other hand, while using our implementation at the HA and having a delay of approximately 20 m sec, we observe that packet loss is very small compared to the HARP implementation, as it introduced packet loss less than 1% when using UDP flows used in real-time applications. This loss is not noticeable in real-time applications and does not cause distraction to the user as mentioned in (Georgopoulos *et al*., 2011).

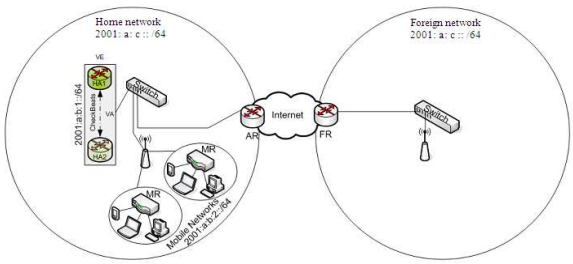

Fig. 7: Network design: Components used in the testbed are shown in this diagram

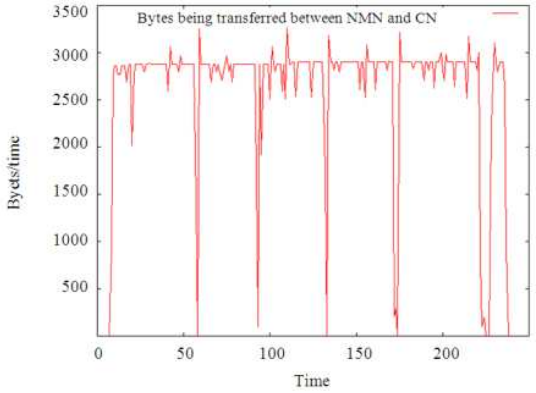

Fig. 8: Packet loss in TCP using normal HARP implementation having a 0.7s HA recovery time, loss is depicted by the dips in the graph

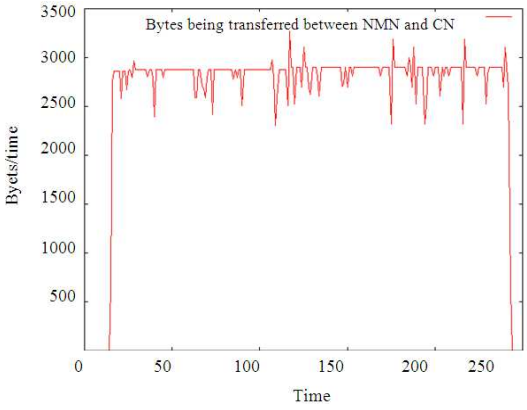

Fig. 9: Packet loss in TCP using normal HARP implementation having a 20ms HA recovery time, loss is depicted by the dips in the graph

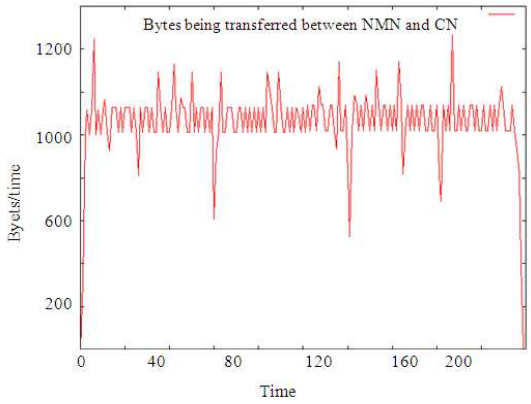

Fig. 10: Packet loss in UDP using normal HARP implementation having a 0.7s HA recovery time, loss is depicted by the dips in the graph

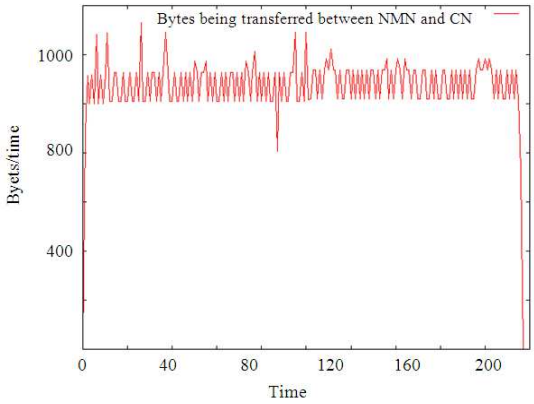

Fig. 11: Packet loss in UDP using our implementation having a 20 m sec HA recovery time, loss is depicted by the dips in the graph

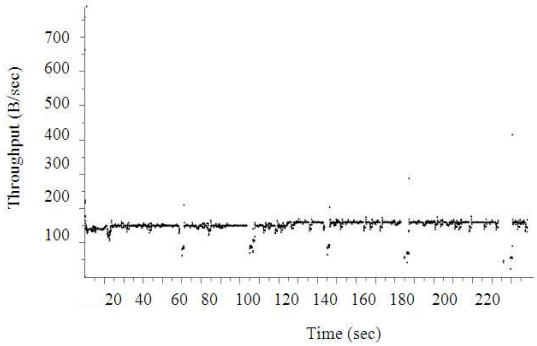

Fig. 12: Throughput using normal HARP implementation having a 0.7 sec HA recovery time

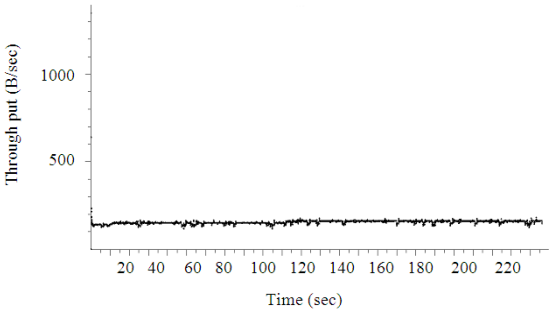

Fig. 13: Throughput using normal HARP implementation having a 20 m sec HA recovery time

From Fig. 12 and 13, similar test environment is used and throughput is illustrated and the failure points are depicted by the discontinuity in the graph.

Table 1: Implementation comparison

Criteria	Our solution	Other
Switch Time	20 m sec	0.7sec
Packet loss	Less than 1% loss	About 65% loss
Throughput	Unnoticeable drop	More than 50% drop
RTT	Average 5 m sec	Average 5 m sec
Overhead	1.4%	0.64%

DISCUSSION

In our experiments, it was found that RTT results have shown similarities when running both implementations.

The similarities are reflected by the RTT value of approximately 5 m sec, since our implementation introduces a negligible amount of overhead to the network used in our test environment (approx. 1.4% of network overhead); therefore RTT similarities is a good sign of network performance when compared to the HARP implementation performance. Moreover, our implementation performs better in terms of HA recovery time and throughput as summarized in Table 1.

CONCLUSION

In this study, we have presented the results of a redundant HA solution to the HA reliability problem in MIPv6. The implemented solution supports NEMO and outperforms available solutions in terms of HA recovery time to suit real-time applications. We have also shown that less packet loss is incurred in our solution, while the effect on the network overhead affecting round trip time is similar to other solutions. These factors are important when deploying a highly available MIPv6 network targeting real-time applications as they directly affect the performance of such networks.

ACKNOWLEDGMENT

The researcher would like to thank MIMOS Berhad for providing the infrastructure needed to run these experiments and all anonymous reviewers for their valuable comments.

REFERENCES

Abdullah, I. and M. Othman, 2010. A simulative study on the performance evaluation for simultaneous and successive mobility for mobile IPv6. J. Comput. Sci., 6: 1511-1517. DOI: 10.3844/jcssp.2010.1511.1517

Ayaz, S., C. Bauer and F. Arnal, 2009. Minimizing end-to-end delay in global haha networks considering aeronautical scenarios. Proceedings of the 7th ACM International Symposium on Mobility Management and Wireless Access (MobiWAC '09), ACM, New York, USA., pp: 42-49. DOI: 10.1145/1641776.1641784

Bassek, C.K., S. Pierre and A. Quintero, 2006. Redundancy Schemes for High Availability Computer Clusters. J. Comput. Sci., 2: 33-47. DOI: 10.3844/jcssp.2006.33.47

Frantti, T. and M. Koivula, 2011. Fuzzy packet size control for delay sensitive traffic in ad hoc networks. Exp. Syst. Appli., 38: 10188-10198. DOI: 10.1016/j.eswa.2011.02.079

Gay, V., T. Hof, V. Perrier and E. Robert, 2009. Design, implementation and performance evaluation of a home agent redundancy solution. Proceedings of the 5th International Conference on Networking and Services, Apr. 20-25, IEEE Xplore Press, Valencia, pp: 69-75. DOI: 10.1109/ICNS.2009.78

Georgopoulos, P., B. McCarthy and C. Edwards, 2011. A collaborative AAA architecture to enable secure real-world network mobility. Lecture Notes Comput. Sci., 6640: 212-226. DOI: 10.1007/978-3-642-20757-0_17

Hernandez-Cons, N., S. Kasahara and Y. Takahashi, 2010. Dynamic Hello/Timeout timer adjustment in routing protocols for reducing overhead in MANETs, Comput. Commun., 33: 1864-1878. DOI: 10.1016/j.comcom.2010.06.011

Khare, V., D. Jen, X. Zhao, Y. Liu and D. Massey *et al*., 2010. Evolution towards global routing scalability. IEEE J. Selected Areas Commun., 28: 1363-1375. DOI: 10.1109/JSAC.2010.101013

Kusin, Z. and M.S. Zakaria, 2011. Mobile node speed detection mechanism in hierarchical mobile Internet Protocol (IPv6). J. Comput. Sci., 7: 1432-1438. DOI: 10.3844/jcssp.2011.1432.1438

Lin, J.W. and M.F. Yang, 2009. Fault-tolerant design for wide-area Mobile IPv6 networks. J. Syst. Software, 82: 1434-1446. DOI: 10.1016/j.jss.2008.07.021

McCarthy, B., C. Edwards and M. Dunmore, 2009. Using NEMO to support the global reachability of MANET nodes. IEEE, 2097-2105. DOI: 10.1109/INFCOM.2009.5062133

Rathi, S. and K. Thanuskodi, 2009. A secure and fault tolerant framework for mobile IPv6 based networks. Int. J. Comput. Sci. Inform. Security, 5: 46-55.

Suchara, M., D. Xu, R. Doverspike, D. Johnson and J. Rexford, 2011. Network architecture for joint failure recovery and traffic engineering. Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '11), ACM, New York, USA., pp: 97-108. DOI: 10.1145/1993744.1993756

Tazaki, H., R. Meter, V. Wakikawa, R. Uehara and K. Murai, 2011. NAT-MANEMO: Route optimization for unlimited network extensibility in MANEMO. J. Inform. Process., 19: 118-128. DOI: 10.2197/ipsjjip.19.118

Wakikawa, R., 2009. Home Agent Reliability Protocol. 75th IETF, MEXT Working Group.

Wu, X. and L. Dong, 2009. Research and design of the Pseudo-VRRP based high availability mechanism in the forCES router. Proceedings of the 8th International Conference on Networks, Mar. 1-6, IEEE Xplore Press, Gosier, Guadeloupe, pp: 440-444. DOI: 10.1109/ICN.2009.49

Zhenkai, Z., Z. Lixia and R. Wakikawa, 2011. Supporting mobility for internet cars. IEEE Commun. Maga., 49: 180-186. DOI: 10.1109/MCOM.2011.5762816