HyperMIP: Hypervisor controlled Mobile IP for Virtual Machine Live Migration across Networks

Qin Li, Jinpeng Huai, Jianxin Li, Tianyu Wo, Minxiong Wen
School of Computer Science and Engineering, Beihang University, Beijing, China
{liqin,lijx,woty,wenmx}@act.buaa.edu.cn, huaijp@buaa.edu.cn

Abstract

Live migration provides transparent load-balancing and fault-tolerant mechanism for applications. When a Virtual Machine migrates among hosts residing in two networks, the network attachment point of the Virtual Machine is also changed, thus the Virtual Machine will suffer from IP mobility problem after migration. This paper proposes an approach called Hypervisor controlled Mobile IP to support live migration of Virtual Machine across networks, which enables virtual machine live migration over distributed computing resources. Since Hypervisor is capable of predicting exact time and destination host of Virtual Machine migration, our approach not only can improve migration performance but also reduce the network restoration latency. Some comprehensive experiments have been conducted and the results show that the HyperMIP brings negligible overhead to network performance of Virtual Machines. The network restoration time of HyperMIP supported migration is about only 3 *second. HyperMIP is a promising essential component to provide reliability and fault tolerant for network application running in Virtual Machine.*

Key words: *Hypervisor*, *Virtual Machine*, *Live Migration*, *Mobile IP*

1. Introduction

In recent years, the Virtual Machine (VM) technology has got widespread attention. Through the use of Virtual Machine Monitor, or Hypervisor, application runtime environment can be deployed on-demand over dynamic, distributed and heterogeneous computing resources [1-3]. VMware and Xen proposed VM live migration technologies [4][5], and using live migration, runtime state of a VM can be transferred from one host to another while the application in VM keeps running during the migration. Current live migration technology can achieve minimum service down-time, it can be used to eliminate the impact of hardware failure and maintenance to enhance the stability and reliability of software, as well as a dynamic load balancing mechanism for hardware resources provision.

Since network interface card of a Virtual Machine is a virtual hardware device emulated by Hypervisor, VM has no direct connection to the physical network through its network interface. Hypervisor intercept network packets on the physical link for VM, and transfer the packets to the VM via the virtual link, and vice versa. In this manner, Hypervisor can be considered as the network attachment point to the VM. Therefore when a VM migrates from one host to another, the network attachment point for the VM is also changed.

The IP address of the VM will most-likely become inaccessible to all of the hosts in the network after migration, because the network attachment point for the VM will change. This problem caused by Virtual Machine migration is similar to IP mobility problem[6] caused by wireless device roaming between different networks, except that there is no physical movement of any device and just movement of software runtime state in the live migration scenario. In order to maintain IP connectivity and keep TCP connection alive after Virtual Machine migration, either Hypervisor, or VM, or other network nodes must renew its network topology information, in order to contact the VM at its new network attachment point.

Selecting an appropriate node responsible for re-establishing network connectivity for VM is crucial to solve the IP mobility problem in live migration. There are three kinds of different approaches according to the node being selected:

VM controlled Mobility: In this approach, the mobility node, or the VM discovers the network attachment point changed, and update network topology through some special nodes in the network. Mobile IP uses this kind of approach to solve wireless

device roaming problem. However this approach is not suitable here, since live migration is transparent to VM. VM cannot use wireless signal strength to determine the timing of migration as in device roaming scenario. Therefore it is inflexible for the VM to discover the migration, and controls the mobility.

Network controlled Mobility: In this approach, the connection restoration is performed by some of the network nodes. The mobility is transparent to the migrated VM, but may or may not be transparent to the correspondent nodes for the VM. Some network controlled mobility mechanisms designed to solve mobility problem in VM live migration require that the correspondent nodes update the network topology information at first in order to restore the network connections with the VM.

Hypervisor controlled Mobility: This approach can be seen as a special case of the Network controlled Mobility, yet more suitable in VM migration scenario. The mobility of Virtual Machine is discovered and managed by the attaching Hypervisor, and fully transparent to VM and other network nodes, which is similar to Proxy Mode Mobile IP solution proposed by 3GPP forum. Since Hypervisor is perfectly capable of predicting exact time and destination host of Virtual Machine migration, it can makes use of these two factors to improve migration performance, as well as to reduce the network restoration latency.

This paper, architecture of Hypervisor controlled Mobile IP system (HyperMIP) is proposed to solve the IP mobility problem in live migration. HyperMIP provides IP mobility service to Virtual Machine completely transparent to VM's OS and applications. According to the evaluation result, using HyperMIP brings negligible overhead to network performance of Virtual Machine.

In the next part of this paper, Section 2, introduces related work on this topic, and discusses them according to aforementioned criteria. The following content is organized as follows. Section 3 defines basic concepts and describes the architecture of Hypervisor control Mobile IP system, namely HyperMIP. Section 4 provides detailed design of HyperMIP and discusses possible improvement in different scenarios. Section 5 gives on quantitative experiment results of the performance of our implementation of HyperMIP, measured by frequently used benchmark applications. Finally, Section 6 summarizes the paper.

2. Related work

This paper mainly focuses on solving IP mobility problem caused by VM live migration. As discussed in Section 1, VM uses virtual network interface device emulated by Hypervisor, therefore Virtual Machine must connect to physical network through the underlying host machine. When a VM migrates to a destination host, the destination host will become the new network attachment point for the VM, and the IP address may become unavailable after migration according to different network configuration on the host of VM.

The following figure shows two typical network configurations scheme for the host of VM. In both schemes, the host serves as a proxy between VM and the physical network. In the left scheme, the host serves as a layer 2 gateway. When other nodes on the physical link request for link layer address of VM's IP address, the host replies for VM and forward packets to VM via virtual link afterward. The host can reply its own link address as discussed in [7], or it can forward the request to VM and forward the reply back to the requester. In the scheme VM can be considered virtually connected to the Switch.

In the right scheme, the host serves as a layer 3 gateway, or a network router. All network packets sent to or received from the VM pass through the host, because it is the next hop according to rules in route table. If the VM uses a private address, network address translation must be used to connect the VM to physical network.

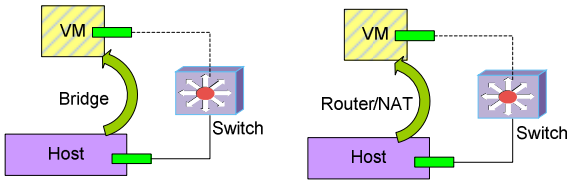

Figure 1: Different VM Networking Schemes

As far as live migration implemented in Xen hypervisor [5], a Gratuitous ARP [8] message will be broadcasted on the physical link after migration, to redirect network packet designated to the IP address of VM from source host to destination host. This mechanism is only effective if the host uses Bridge Networking for the VM and two hosts locate in the same link or network, because VM resides in the same link after the migration. If the host uses Route Networking or NAT Networking for the VM, then VM will roam to a different link after any kind of migration. In this case, network traffic cannot be redirected by layer 2 protocol, like ARP.

It is also discussed in [5] that the possibility to use Mobile IP[6] to solve this problem. But in traditional Mobile IP scenario, a Mobile IP stack pre-installed on mobile node is used to discover the movement and predict the movement destination by wireless signal strength. This assumption of Mobile IP does not apply

in the case of live migration of VM. In live migration, VM is in deed the mobile node, but it is not preferable for VM to pre-install a Mobile IP stack, also it is infeasible for VM to predict the time and destination of migration, as live migration is designed to be transparent to VM.

R. Bradford et al proposed an enhancement to Xen live migration[9], enabling live migration of both runtime state and local persistent state for VM. As to mobility problem, it also proposed a method to combine IP tunnel and dynamic DNS to maintain network connectivity for VM. Virtual Machine uses two IP address after migration. Network packets designated to the old address are transferred through IP tunnel established between source host and destination host of VM. New connection will be designated to new address according to dynamic DNS. This method is not transparent to VM because it requires Virtual Machine to use two addresses at the same time. Therefore it requires VM to discover the migration and restore network connection itself. Furthermore, the source host cannot free all resources used by the VM after migration, because it is responsible for forwarding the packets designated to the old address through tunnel. It is not mentioned for how long the source host should wait before it removes the tunnel, and what happens if some nodes still used the old address after the tunnel is removed.

F. Travostino et al described a solution [10] to implement seamless live migration over wide-area network. This solution can be considered as a Network Controlled Mobility solution, which requires other network peers to setup an IP tunnel to the host of the VM, before communicating with the Virtual Machine. When VM migrates to a new host, a tunnel to the new host will be established on the correspondent node. This solution does not require VM to involve in the live migration process. But correspondent nodes must participate in the process, which has several disadvantages. Firstly, if the correspondent node needs to be unmodified, it cannot benefit from this solution. Secondly, if each correspondent node creates a tunnel on the host, the number of client that a VM can serve is greatly limited by the network tunnel resources consumed on the host, which is also a huge burden on the host itself.

3. Hypervisor controlled Mobile IP

In this section, we will introduce our new architecture, HyperMIP, to solve the mobility problem during live migration. HyperMIP can enable IP Mobility for VMs without any modifications to Virtual Machine and other network nodes.

Firstly, this section discusses why Hypervisor controlled Mobile IP is suitable live migration between different networks. Secondly, we introduce some basic concepts in Mobile IP. Thirdly, we describe the architecture of HyperMIP system and how it works. Lastly, it explains the HyperMIP message interactions in several typical VM Operations.

3.1 HyperMIP Overview

The idea of Hypervisor controlled Mobile IP is inspired by Proxy Mode Mobile IPv4[11]. HyperMIP is compatible to Mobile IP standard with special design to gain better performance in VM live migration scenario. It has several advantages:

Firstly, mobility is merely controlled by Hypervisor, and assisted by special network device called Home Agent, which makes live migration transparent to all other nodes in the network, including the Virtual Machine itself. After migration, previous attaching host of VM can free all the resources allocated for the VM without disconnecting the network of the VM.

Secondly, the fact that Hypervisor is able to precisely predict the timing and the destination of live migration helps to reduce the network restoration latency and minimize the impact on the network performance after migration.

Finally, the adoption of standard protocols allows reusing existing standard network devices to solve the network related issues in live migration, without creating another similar but non-standard solution. It also makes this solution more scalable, and less problematic to integrate or inter-operate with other systems.

3.2 Basic Concepts

Since this paper re-uses standard Mobile IP technology in the solution, following discussion also follows the terminology defined by Mobile IP standard. Before further discussion, it is important to define some of the terms selected from [5] here:

Mobile Node, or MN, is a network device or entity that changes its point of attachment from one network to another. Mobile Node is in fact the VM in the case of live migration.

Home Agent, or HA, is a special router on a mobile node's home network which tunnels packets for delivery to the mobile node when it is away from home, and maintains current location information for the mobile node.

Home Network, a network with a network prefix matching mobile node's home address. In standard IP

routing, packets destined to a mobile node's Home Address will be delivered Home Network.

Foreign Network, any network other than the mobile node's Home Network.

Correspondent Node, or CN, is a peer with which a mobile node is communicating.

Foreign Agent, or FA, is a special router on a mobile node's visited network which provides routing services to the mobile node. The foreign agent de-tunnels and delivers packets to the mobile node that were tunneled by the mobile node's home agent.

Hypervisor Mobile IP controller, or HyperMIP, is new module on Hypervisor, which has built-in Foreign agent Functionality. When invoked by Hypervisor, HyperMIP register VM's IP address at home agent. The term HyperMIP also refers to the solution proposed in this paper.

3.3 Architecture of HyperMIP

There are two key entities in architecture of Hypervisor controlled Mobile IP. The first is called Home Agent, which is fully compatible to standard Mobile IP, with optional enhancement to improve the performance in VM migration scenario, which will be discussed in section 4. The latter is called Hypervisor Mobile IP controller, namely HyperMIP. HyperMIP is integrated with Hypervisor, when VM is created, destroyed, migrated, or any operations involved with network connectivity, HyperMIP module will be invoked by Hypervisor to change the network topology for VM.

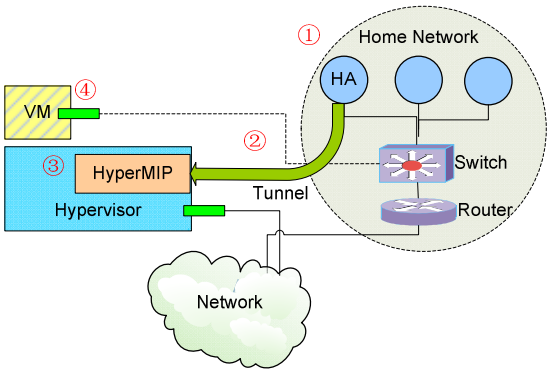

Figure 2: Architecture of Hypervisor controlled Mobile IP

As depicted in Figure 2, a VM is virtually connected to a remote home network through a tunnel established between HyperMIP and home agent. All other nodes, including VM itself, recognize the VM as always residing in home network. Even it may have been migrated several times. As shown in the figure, packets sent to VM are delivered through following steps:

1. Home Agent receives the packets on behalf of the registered VM on the home network.
2. The intercepted packets will be delivered to current attaching host of the VM through the tunnel.
3. HyperMIP receives packets from the tunnel, then sends to the VM through virtual link
4. VM receives the packets from its virtual network device, as if it was received directly in home network.

The packets sent from VM are of course delivered in the reverse order of the previous process.

3.4 Virtual Machine Operations

If an administrator takes some management operations to the VM modifying its network connectivity state, such as creating or migrating VM, HyperMIP will send appropriate message to home agent reflecting the adjustment. The message format is conformant to standard Mobile IP, including information like VM's IP address and preferred lifetime before deregistration. The message may include optional extensions to reduce network restoration time after live migration.

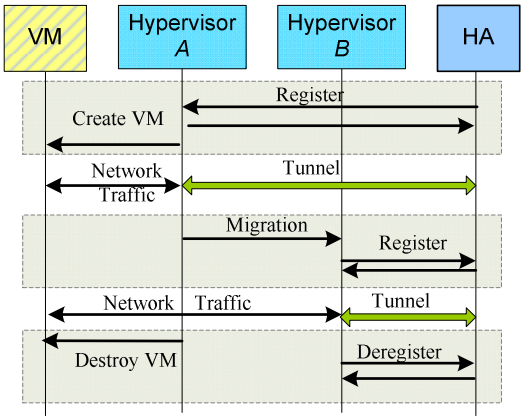

Figure 3: Message flow in VM Operations

The above figure describes the message interaction between Hypervisor and Home Agent in three operations.

Create VM, before VM is created, Hypervisor must register VM's address to home agent. Packets for VM are delivered through tunnel established between home agent and the attaching Hypervisor.

Migrate VM, after VM migrates from source host to destination host, Hypervisor on the destination host must register itself to the home agent as the new receiver of VM's address. When new tunnel is established on home agent, old tunnel will be destroyed automatically. In the mean time, source host can free all resources used by VM including the tunnel.

Destroy VM, after Hypervisor destroys a VM, it invokes HyperMIP to de-register VM's address from home agent.

4. Detailed Design and Implementation

We implemented the aforementioned HyperMIP and Home Agent module under Linux in C++ language. The code can be downloaded from website http://code.google.com/p/proxymip4/. Section 4.1 introduces the implementation details of the bi-directional packet forwarding, the core function of HyperMIP. Section 4.2 discusses how to enable IP address dynamic allocation for VM. Section 4.3 discusses how to reduce restoration latency. Section 4.4 introduces security issues in HyperMIP. Section 4.5 provides details on integrating HyperMIP with existing systems.

4.1 Bi-directional Packets Forwarding

Bi-directional packets forwarding is a core function of our system.

All packets destined to the VM are delivered to home network first. Home agent will capture the packets on behalf of the VM. Packet capture can be realized by Proxy ARP: when some host uses ARP to resolve link layer address of the VM's IP address, home agent will reply its own link layer address to the requester. Then home agent can receive the packets destined to the VM.

All packets sent by the VM are received by Hypervisor first. Because the VM is physically disconnected to the home network, when the VM uses ARP to resolve link layer address of any other home nodes, especially that of the VM's default gateway, Hypervisor must reply its own link layer address on the virtual link using Proxy ARP. This is similar to the home agent, except that home agent only works as a proxy for VM, Hypervisor works as a proxy for all other home nodes.

After home agent or Hypervisor receives the packet, it delivers the packets through the tunnel according to the rules in routing tables respectfully.

In our implementation, packet capture mechanism is implemented using Linux ProxyARP[7] functionality; packet delivering is realized by manipulating traditional route table and policy route table of Linux; Tunneling and de-tunneling of IP datagram is done natively using Linux's iptunnel support. These functions are all dealt in Linux kernel to gain maximum performance.

4.2 Dynamic Address Allocation

In previous discussion, it shows that Hypervisor must register the IP used by VM to home agent before creating the VM, otherwise the network is disconnected. This requires that VM uses a fixed IP address known by Hypervisor previously, which brings more management work to administrators.

Mobile IP allows dynamic address allocation, but enforces mobile node to request a new address using Mobile IP message. In HyperMIP scenario, no Mobile IP stack is previously installed on VM. Virtual Machine will most likely use DHCP[12] to request for new address.

To support dynamic address allocation using DHCP, HyperMIP should delay the address registration. After it captures DHCP request from VM, it can request address allocation from home agent, then sends DHCP reply containing the allocated address to the VM.

4.3 ARP Table Updating

Each network node has an ARP table containing link layer addresses of other nodes located in the same link with the owner of ARP table. This ARP table is used as a cache of ARP reply to avoid frequent broadcasting of ARP request during network communicating.

In Virtual Machine migration, after HyperMIP establishes the tunnel on destination host, the network layer or layer 3 can be consider reconnected. But it is very likely that the VM will still suffer from packet loss for a period of time. In order to fully restore network connection, some nodes need to update its ARP table first. Otherwise, network will be restored only after the concerning ARP table entries become invalidate after several tens of seconds.

Considering that the correspondent node itself can also be a VM connected to home network via HyperMIP, there are four kinds of nodes which should update its ARP table.

Correspondent Nodes at Home Network: The ARP table contains an entry concerning VM' IP, which must refer to the link layer address of the current attaching point of the VM. If the VM attaches to the home network via Bridge Networking, current attachment point is the attaching host of the VM. If the VM is away from Home Network, current attachment point is the home agent. After migration, the ARP table must be updated accordingly.

Virtual Correspondent Nodes at Source Host: Since the correspondent nodes and the migrating VM are on the same host before migration, the correspondent node's ARP table should contain the link layer address of VM, which must be changed to the source host's after migration.

Virtual Correspondent Nodes at Destination Host: Since the correspondent nodes and the migrating VM

are on the same host after migration, the correspondent node's ARP table should contain the destination host's link layer address, which must be changed to the VM's link address after migration.

The VM being migrated: The ARP table of the migrating VM contains entries of the Correspondent Nodes in the same network. When VM attaches to the home network via Bridge Networking, the entries should refer to actual link address of the correspondent node. If the VM is away from host, the entries should refer to the link address of current attaching host. After migration, the ARP table must be updated accordingly.

ARP table can be updated using Gratuitous ARP[8]. Since Gratuitous ARP is widely supported by most of the Internet nodes without special configuration, it brings no additional requirements to the correspondent nodes and the VM. The ARP table of correspondent nodes at home network is updated by home agent. The ARP table of VM, including both migrating VM and virtual correspondent nodes, are updated by HyperMIP module runs on their current attaching hypervisor.

Our implementation of HyperMIP support ARP table updating in live migration. It brings significant performance improvement for VM's network restoration, which is measured in the next section.

4.4 Security Considerations

To prevent other malicious hosts from stealing network packets sent to VM, message integrity protection must be applied to messages between HyperMIP and home agent. The standard mandates that all Mobile IP message be authenticated using a secret key shared between the mobile node and home agent. This assumption is not applicable in HyperMIP since the mobile node here is the VM, which is not involved with Mobile IP message interaction. The shared key between mobile node and home agent, even if it exists, cannot be used to protect the message.

HyperMIP uses a secret key sharing between the attaching host and the home agent to authenticate the message. Since HyperMIP enables VM migration across different administrative domains, a more effective way like automated trust negotiation [13][14] are preferable to establish trust relationship between the attaching host and the home agent.

4.5 System Integration

HyperMIP is highly independent from Hypervisor, the interface between HyperMIP and Hypervisor contains only two operations, connect and disconnect. This interface is called by Hypervisor when network adjustment was made to the VM. We integrated our HyperMIP with Xen version 3.0.3[15], requiring only a few lines of modification to the Xen daemon source code. We further integrated HyperMIP enabled live migration in our resource sharing infrastructure[16][17], to support reliable, trusted, on-demand application instantiation over heterogeneous, dynamic, and distributed computing resources.

5. Evaluation

We designed three experiments to evaluate the performance of HyperMIP from different perspectives. The first experiment evaluates network performance of a VM connecting to a remote Home Network using HyperMIP. The second compares the performance of two modes of VM live migration, one with HyperMIP enabled and the other disabled. The third experiment evaluates the optimization contributed by ARP table updating.

All experiment results were measured by popular network benchmarking tools. We use apache's *ab*[18] to measure the Web application performance under the use of HyperMIP. We use *tbench*[19] to measure network throughput under the use of HyperMIP t.

Experimental environment includes eight PC machines with Intel Core Duo 7500 CPU and 4G memory. We uses these eight PCs together with three 100Mbps LAN Switches to establish different network topology for each experiment.

5.1 Packet Forwarding

To support VM migration across distributed networks, a VM must be configured to connect to Home Network via HyperMIP. This experiment measures network performance of a VM running in HyperMIP packet forwarding mode. The test bed consists of four physical machines connected via three Switches as depicted in Figure 4. Two Virtual Machines are created on one machine with different network configuration. One VM uses normal Bridge Networking to connect to Switch 2, the other uses HyperMIP to connect to Home Network. Figure 4 shows the packet route path of each VM.

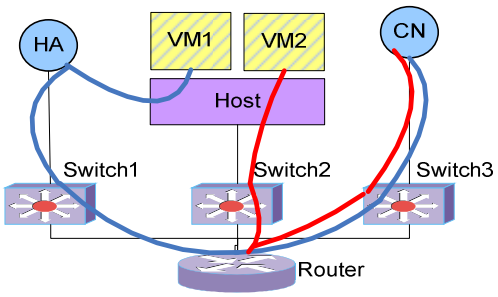

Figure 4: Packet Forwarding Experiment Scenario

The result is generated by *ab* and `tbench`. The benchmark tool runs on CN, the server runs on both VM1 and VM2. Figure 5 depicts 10 seconds of the network throughput of VM1 and VM2 generated by each benchmark tools. It shows that the overhead introduced by HyperMIP is negligible for *ab*, and less than 3% for *tbench*.

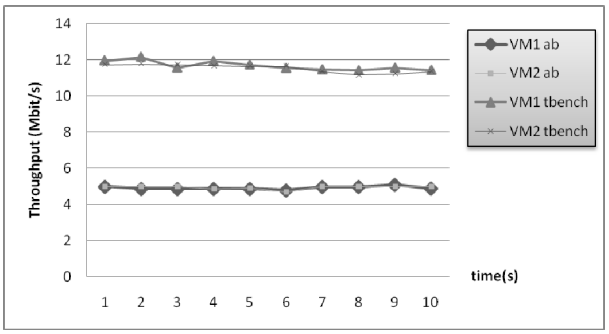

Figure 5: Result of Packet Forwarding Experiment

5.2 Live Migration

HyperMIP enables VM live migration across networks. This experiment compares the performance of two modes of live migration, one with HyperMIP enabled and the other disabled. It uses eight physical machines connected via three Switches, as depicted in Figure 8. The migrating VM is called VMN.

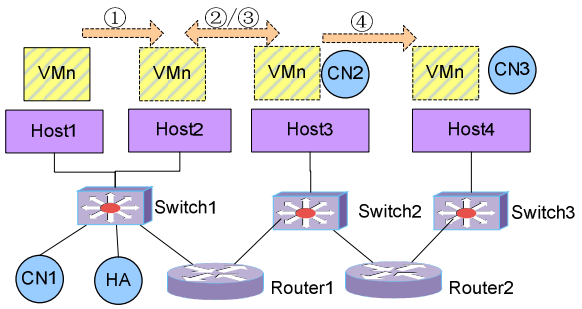

Figure 6: Live Migration Experiment Scenario

This experiment measures four kinds of VMN live migration (Mig1-4). In Mig1, VMN migrate among hosts in the same network. In Mig2 and Mig3, VMN migrate away from Home Network and migrate back respectfully. Mig4 migrate VMN among hosts in foreign networks. Note that, HyperMIP is disabled in Mig1, and it is enabled in Mig2, Mig3 and Mig4. Experiment result is generated by ab. The client of benchmark tool runs on CN1, and server runs on VMN.

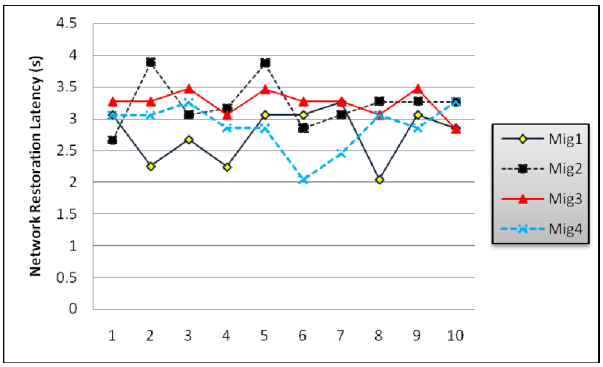

Figure 7: Result of Live Migration Experiment

Figure 7 shows the benchmark result of doing Mig1 to Mig4 for 10 times. All data are within the range of 2s to 4s. With HyperMIP disabled, the average network restoration time after migration in the same network is 2.75s (Mig1). The restoration time among different networks are 3.23s (Mig2), 3.24s (Mig3), 2.85s (Mig4). It reveals that in order to support live migration across networks, HyperMIP introduces an overhead of about 100ms to 400ms to live migration.

Figure 8 shows the traffic data during one HyperMIP enabled migration selected from the experiment. The migration starts at the fifth seconds, and last for about 38 seconds. At the end of migration, no traffic is sent or received by the VMN for about 3 seconds. After VM migration completes, the quality of services is restored to normal level rapidly.

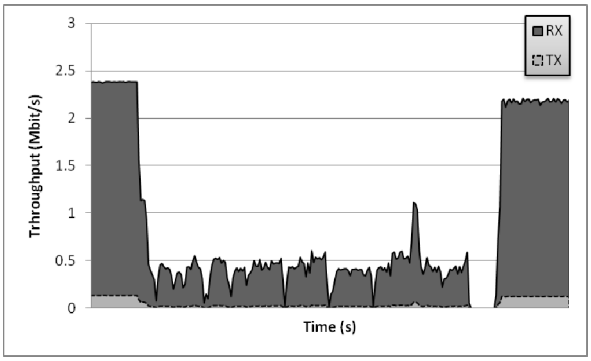

Figure 8: ApacheBanch Traffic in Live Migration

5.3 ARP Table Updating Optimization

In order to evaluate effectiveness of ARP table updating mechanism, we did another live migration experiment after we disabled ARP table updating feature in HyperMIP. This experiment uses the same test bed topology with that of the experiment 2. Note that there are three correspondent nodes deployed in the test bed (CN1, CN2, CN3), each representing one of

the three kinds of Correspondent Nodes, as discussed in section 4.3. There are three set of results, generated by running *ab* from CN1 to CN3 while VMN is migrated four times. As listed in Table 1, it shows that with ARP table updating disabled, network restoration time varies from tens of seconds to 3 second, which also shows that ARP table updating mechanism is effective for the all three kinds of Correspondent Nodes.

Table 1: Result of Experiment 3

Migration Type	CN1	CN2	CN3
Mig1	18.2	3.2	3.0
Mig2	24.1	19.2	3.4
Mig3	19.3	3.2	15.3
Mig4	2.8	20.7	16.8

6. Conclusion and future work

When a VM migrates from one a source host to another host which resides in a different network, the network attachment point of the VM changes, thus creates the mobility problem for VM live migration. To solve this problem, this paper introduces an approach called Hypervisor controlled Mobile IP, namely HyperMIP, to support live migration of VM across different and distributed networks. HyperMIP provides IP mobility to VM completely transparent to the OS and applications that run within the VM. According to the evaluation result, the use of HyperMIP brings negligible overhead to network performance of VM.

Hypervisor is capable to predict exact time and destination host of Virtual Machine migration. To further support seamless mobility for VM live migration, HyperMIP makes use of these two factors to improve migration performance, as well as to reduce the network restoration latency. According to the evaluation result of our HyperMIP implementation, the ideal network restoration time can be reduced to about 3 seconds average, only 100ms to 400ms more than the network down-time introduced by Xen VM live migration itself. Using HyperMIP, VM has the ability to lively migrate across largely distributed computing resources, which can be used to further ensure reliability and fault tolerant of network application in VM.

Our future work includes several directions.

Firstly, we have only implemented Hypervisor controlled mobility for IPv4 now, with IPv6 in the future plan. Supporting IPv6 is almost the same with IPv4 in concept and procedure, but different in implementation.

Secondly, although this paper does not discuss migration of local persistent data of VM, yet it must be combined with HyperMIP to fully support live WAN migration. Further research will be focused on transferring local persistent data more efficiently to reduce service down-time in live migration.

Last but not least, VM migration across distributed resources from different administrative domains brings severe security problems from resource management and monitoring perspective, secure and trusted Virtual Machine migration should be studied in the future.

7. Acknowledgements

This work is partially supported by grants from China 863 High-tech Program (Project No. 2006AA01A106, Project No. 2007AA01Z120), China 973 Fundamental R&D Program (No. 2005CB321803) and National Natural Science Funds for Distinguished Young Scholar (Project No. 60525209). We would also like to thank members in Distributed Computing Research team in Institute of Advanced Computing Technology of Beihang University for their hard work on CROWN grid middleware.

References

[1] K. Keahey, et al., "Virtual Workspaces in the Grid," in *11th International Euro-Par Conference* Lisbon, Protugal, 2005,

[2] P. Ruth, et al., "Virtual Distributed Environments in a Shared Infrastructure," in *IEEE Computer*. vol. 38, 2005, pp. 39-47,

[3] S. Adaballa, "From Virtualized Resources to Virtualized Computing Grid: The In-VIGO System," in *Future-Generation Computing System*,

[4] M. Nelson, et al., "Fast Transparent Migration for Virtual Machines," in *USENIX Annual Technical Conference*, 2005,

[5] C. Clark, et al., "Live Migration of Virtual Machines," in *2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)* Boston, MA, 2005, pp. 273-286,

[6] C. Perkins, et al, "IP Mobility Support for IPv4", 2008 http://tools.ietf.org/html/rfc3344

[7] J. Postel, et al, "Multi-LAN Address Resolution", RFC 925, 1984

[8] W. Stevens, et al, "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley, Reading, Massachusetts, 1994.

[9] R. Bradford, et al., "Live Wide-Area Migration of Virtual Machine Including Local Persistent State," in *Proceedings of the 3rd international conference on Virtual execution environments (VEE07)*, New York, NY, 2007, pp. 169-179,

[10] F. Travostino, et al., "Seamless Live Migration of Virtual Machines over the MAN/WAN", in *Elsevier Future Generation Computer Systems*, 2006, pp. 901-907

[11] K. Leung, et al, "WiMAX Forum/3GPP2 Proxy Mobile IPv4", 2008,

http://tools.ietf.org/html/draft-leung-mip4-proxy-mode-08
[12] R. Droms, et al, "Dynamic Host Configuration Protocol", RFC 2131, 1997.
[13] Jianxin Li, et al., TOWER: Practical Trust Negotiation Framework for Grids. In *2nd IEEE International Conference on e-Science and Grid Computing (eScience 2006)*, Amsterdam, 2006.
[14] Jinpeng Huai, et al., ROST: Remote and hot service deployment with trustworthiness, in CROWN Grid Journal of Future Generation Computer Systtem. Vol.23 No.6.825-835, Elsevier Science Publishers, 2007.
[15] P. Barham, et al., "Xen and the Art of Virtualization," in *ACM Symposium on Operating Systems Principles*, 2003,
[16] J. Huai, et al., "CROWN: A service grid middleware with trust management mechanism," in *Science in China Series F-Information Sciences 2006 Vol.49 No. 6 pp.731-758*,
[17] J. Huai, et al., CIVIC: a Hypervisor based Virtual Computing Environment, In *International Conference on Parallel Processing Workshops 2007*, 10-14 Sept. 2007 Page(s):51 - 51
[18] Apache HTTP server benchmark tool, 2008, http://httpd.apache.org/docs/2.0/programs/ab.html
[19] Dbench – Emulating NetBench benchmark tools http://samba.org/ftp/tridge/dbench/