Multihoming Support for Mobile IPv6 Networks

Weiwei Xu

in fulfilment of the requirements for the degree of

Master of Science
in
Telematics

Dr.Ir. Sonia M. Heemstra de Groot
Neill Whillans, M.Eng
Malohat Kamilova, P.D.Eng
Dr.Ir. Geert Heijenk
Dr.Ir. Irene de Bruin

Design and Analysis of Communication Systems
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente

August 2005
Enschede, the Netherlands

Abstract

Network Mobility is a research field being widely explored as it has potential implementation possibilities in daily life, such as ubiquitous Internet communications. Network Mobility (NEMO) Basic Support has been proposed for the purpose of managing the mobility of a network, which changes its attachment to the Internet, through a Mobile Router.

This thesis addresses the multihoming issues of MIPv6 Networks on the basis of NEMO Basic Support, analyzes the benefits of multihoming and discusses implementation issues of all classes of multihoming possibilities. Additionally, Policy-based routing, as one of the multihoming benefits, is studied in particular. A policy-based routing framework is proposed for handling both the inbound and the outbound traffic on a mobile network, under specified policies which consider packet characteristics, current network situation and user preferences. A policy protocol is designed for policy messages communication between the Mobile Router and the Home Agent. The link between NEMO Basic Support and the Policy-based routing framework is also discussed.

The framework is simulated in NS2, and simulation results show that the performance of this framework is influenced by the network situation. Furthermore, other multihoming benefits, for instance, fault redundancy and load-sharing, are also revealed through some simulations.

Acknowledgements

Since I started my master project, many people have given me invaluable help from far and near. I want to say a thousand thanks to all of them.

My advisor, Dr. Sonia Heemstra de Groot, is not only a constant source of expert counsel, but also the provider of big confidence and wireless help. She is so friendly to create good discussion environments. I am especially grateful to another advisor, Neill, who devoted a lot of time on my project, particularly, on the simulation part. His detailed explanation and angelic patience gave me a great help on understanding and employing the simulation tool.

I thank Malohat for her many insightful comments on policy-based routing. Thanks also to Geert and Irene, who gave me all kinds of suggestions during my project.

Special thanks go to WMC for offering me the opportunity and support of finishing my thesis.

Finally, I dedicate this thesis to my family and friends without whom none of this would be possible.

Contents

List of Figures

List of Tables

1. Introduction

As wireless networking products and services proliferate, users expect to be connected to Internet from anywhere at anytime. Owning more than one mobile device or multiple access technologies helps it come true. With the development of Bluetooth technology, **Personal Area Network (PAN)** appears. A PAN is a computer network composed of computer devices (including telephones and personal digital assistants, etc.) close to one person. The reach of a PAN is typically a few meters, and the devices in a PAN may or may not belong to the person in question. PANs can be used for communication among the personal devices themselves (intrapersonal communication), or for connecting to a higher level network. Some devices in a PAN may be able to access the Internet directly, while others may not. However, those less powerful devices could communicate with remote nodes by first communicating with those powerful devices inside the PAN.

A new concept, **Personal Network (PN)**, is proposed by extending the concept of PAN. Unlike the present PAN which has the limitation of coverage, PN has an unrestricted geographical span and incorporates devices into the personal environment regardless of their locations.

The ongoing project, **My personal Adaptive Global NET (MAGNET) [MAGNET]**, aims at enabling commercially viable PNs that are attractive, affordable and also beneficial for the end-users in their everyday life. In this project, a PN is a distributed personal environment consisting of a core PAN, which is extended on-demand and in an ad-hoc fashion with personal resources or resources belonging to others. This extension will physically be performed via infrastructure network, e.g., the Internet, an organization's intranet, a PAN belonging to another person, a vehicle area network or a home network. A PN is configured to support the application and takes into account context and location information. **Figure 1** presents a PN example.

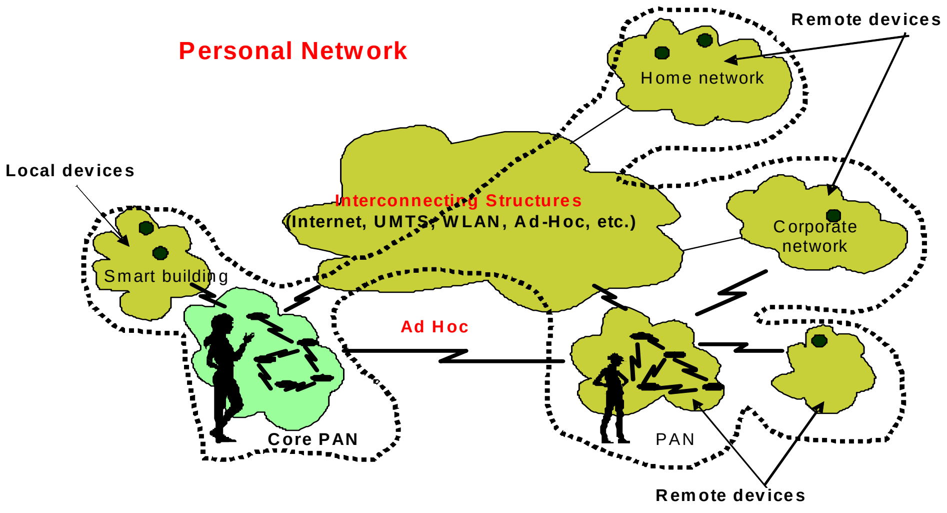


Figure 1: An example of Personal Network

The motivation of creating a PN comes from the belief that new technologies should be centred on the user to improve the quality of life without the need for the user to be aware of any technical details, for instance, the radio access technology that it currently uses. The communication environments should be also smart, responding to and accommodating the users' needs. Ubiquitous communications and access to information are also essential for providing continuous and high quality services.

One implementation scenario of a PN is: a salesman, who has to visit several clients on one day, carries several devices, like a camera, mobile phone and laptop etc. During consulting, he might record some information in his laptop or retrieve some information from his computer in his office. After making a deal, he might scan the contract. The contract would be sent to his laptop directly from the camera and then sent back to his office, where the printer in his office network would print the contract out directly. Other colleagues might start processing the contract before the salesman comes back.

In order to fulfil such kind of implementation, a flexible PN that supports resource-efficient, robust, ubiquitous service provisioning in a secure, heterogeneous networking environment for nomadic users, needs to be designed.

A **cluster**, which is defined as a collection of PN nodes that are connected to each other without any help of foreign nodes[1] which are out of the cluster, is a significant concept. It means that two PN nodes in the same cluster could form a path only by using nodes inside the cluster. A PAN, office network and car network etc, can also be considered as a cluster. It is actually a mobile network.

Two types of nodes are defined in a cluster, Gateway Node and Inner Node. Gateway Nodes are those who have links to foreign nodes. These nodes could be used by the cluster to set up connections with foreign nodes, Internet or other clusters. However, being a Gateway node requires some extra functionality, which might/should not be offered by all nodes. Meanwhile, due to the mobility of nodes in a PN, the structure of a cluster is also dynamic, one node might move out of one cluster and into another one or be switched off/on, and the cluster could split when a person takes some devices and leaves the rest behind. Thus, gateway nodes should be selected according to the real time situation, for example, the availability of devices and the network. Gateway node selection can be considered as a kind of routing for the mobile nodes in the mobile network, as from the perspective of a mobile node, one specific gateway node determines one connection to the infrastructure. [ANAPN] discusses the network architecture for a PN.

1.1. Problem Statement and Thesis Structure

The mobility issues discussed in IP layer by NEMO group in IETF fits the domain of PN. Some mobility issues include nested mobility, multihomed mobile networks, route optimization and mobile devices from different administrative domains (security issues), etc.

To ensure continuous connectivity, with a desirable Quality of Service, it is preferable for a mobile network to be connected via several interfaces, several access technologies and distinct access networks. In this thesis, we concentrate on multihoming supports in a mobile network. We study the potential benefits of multihoming supports, the diverse problems for each multihoming class and propose

[1] A foreign node is not part of the PN. It can either be trusted or not.

a policy-routing framework for multihomed networks, since a cluster in a PN is normally a multihomed network, which is equipped with multiple access interfaces to the infrastructure. Our policy-routing framework is implemented on a publicly available network stack to allow the simultaneous use of multiple interfaces. It is designed to control both inbound and outbound traffic. The handoff decisions are based on explicitly defined policies. Events include the interface availability, connection situation, cost, and user preference. The whole framework is based on the NEMO Basic Support.

The rest of this report is structured as follow:

Section 2 presents the mobility issues of a mobile node and related solutions, including IP mobility problems and Mobile IP. Then we have a look at Network Mobility issues, which is an extension of the basic Host Mobility.

In Section 3, we study one important aspect of Network Mobility: Multihoming. We begin by defining the multihoming concept and exploring its benefits. After that, we examine eight multihoming scenarios and conclude this section with multihoming related issues.

Then in Section 4, we move on to policy-based routing, which is one solution for multihoming issues. We introduce a common policy framework proposed by the IETF, including its structure and problems.

Based on discussions of NEMO and policy-based routing, in Section 5, we come up with our policy-based routing framework within the NEMO architecture. Through a policy example, we illustrate several framework candidates, analyse their effects on the whole system and conclude with one framework.

Section 6 addresses the design of a policy protocol for the policy-based routing framework; while in Section 7, we continue our work by implementing our framework. We design the communication messages format and simulate it with NS2. Simulation results are also presented to prove the benefits of multihoming support listed in Section2.

Finally, in Section 8, we present the conclusions of this thesis and recommend areas for future work stemming from this research.

2. Mobility within an IP Network

In this section, we are going to investigate Mobile IP technology, which is the solution for Host Mobility. After that, we will highlight the limitations of Mobile IP for a mobile network and then describe a possible solution, Network Mobility (NEMO).

2.1. Issues of IP Mobility

With the proliferation of laptops, PDAs and mobile phones, today an increasing number of devices are on the move. A mobile node therefore likely changes its point of attachment to the infrastructure over time.

As we all know, an Internet application needs to know the IP address and port number of the remote entity with which it is communicating. Users want to keep the connection while on the move. From a network layer perspective, a user is not mobile if the same link is used, regardless of location. If a mobile node can maintain its IP address while moving, it makes the movement transparent to the application, since mobility becomes invisible.

2.2. Mobile IP

Mobile IP [MIP] is designed to allow a user to maintain the ongoing connections while moving from one network to another by maintaining its IP address all the time.

In Mobile IP, each mobile node has two IP addresses. One is a **home address (HoA)**, which is used to identify the mobile node. The other is a **foreign address**, which is used to identify its location. A Home address is the permanent address of a mobile node and obtained at its home link. A Foreign address, also called **Care of Address (CoA)**, is obtained in a foreign network using a protocol such as **DHCP (Dynamic Host Configuration Protocol)** [DHCP] when the mobile node moves to a foreign link.

The entity within the home network that performs the mobility management functions on behalf of the mobile node is called the **Home Agent (HA)**.

In Mobile IPv4 [MIPv4], a **Foreign Agent (FA)** is named as the entity in the foreign network which helps the mobile node with the mobility management functions. However, in Mobile IPv6, FA is not used any more since a mobile node itself could perform the mobility management functions. A FA/mobile node has to register itself onto a HA by sending a Binding Update message, so that the HA could track the location of the mobile node and manage the delivery of packets.

The entity which is communicating with the mobile network is called a **Correspondent Node (CN).**

Now, let us have a look at the routing methods used in Mobile IP, both Indirect Routing and Direct Routing. In book [CN-ATDAFI], detailed discussions could be found.

2.2.1. Indirect Routing to a Mobile Node

With indirect routing, a HA is responsible for interacting with a FA/MN to track the mobile node's CoA and looking out for all packets destined for a mobile node that is currently in a foreign link.

When a mobile node sends a datagram to a CN, the procedure is quite easy, since the mobile node knows its CN's IP address, it could address its datagram directly to the CN without passing by any special entities.

However, when a CN wants to send a datagram to a mobile node, the situation becomes a little more complex. Since the CN only knows the mobile node's home address, its datagram could only be sent to the HA from which the home address is advertised. However the mobile node is now in a foreign link, and taken care through its CoA. As the HA knows where the mobile node is currently located by receiving a Binding Update message, the received datagram can be encapsulated by the HA and delivered to the mobile node's CoA. Finally, the mobile node would receive its data successfully.

The whole procedure and explanation of indirect routing are also depicted in **Figure 2**. As we can see, a triangular route is created with Indirect Routing.

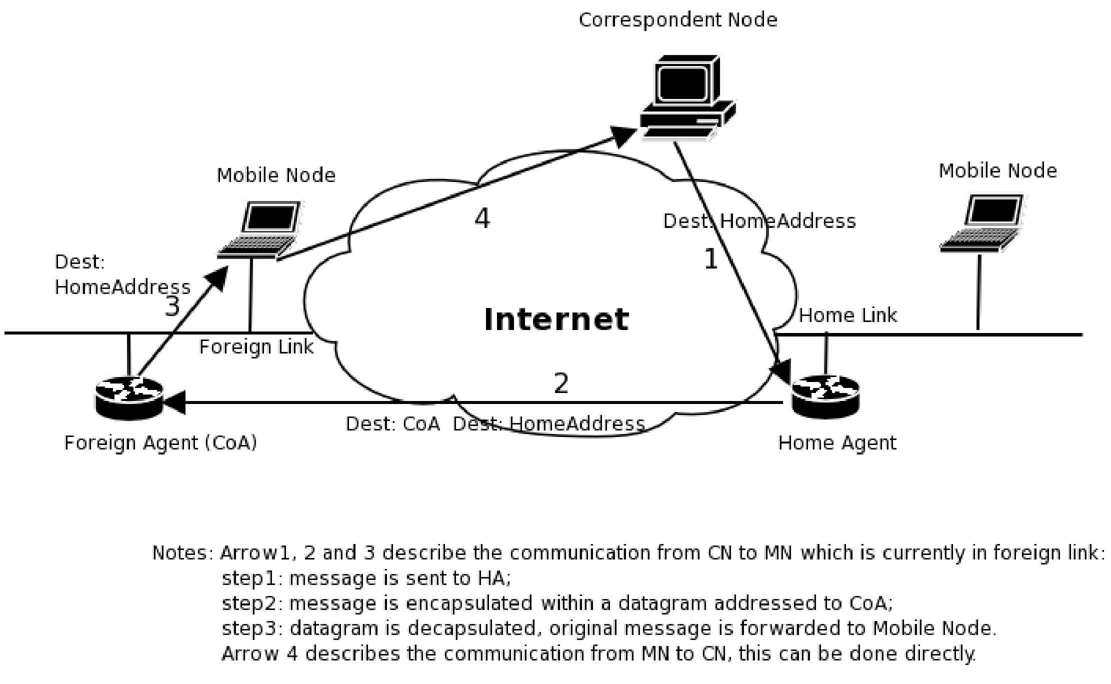

Correspondent Node
Mobile Node
Mobile Node
4
Dest: HomeAddress
1
Dest:
HomeAddress
3
Internet
Home Link
Foreign Link
2
Dest: CoA Dest: HomeAddress
Foreign Agent (CoA)
Home Agent
Notes: Arrow1, 2 and 3 describe the communication from CN to MN which is currently in foreign link:
step1: message is sent to HA;
step2: message is encapsulated within a datagram addressed to CoA;
step3: datagram is decapsulated, original message is forwarded to Mobile Node.
Arrow 4 describes the communication from MN to CN, this can be done directly.

Figure 2: Indirect Routing

The IP-in-IP message passed between the FA and the HA has the format shown **Figure 3**.

HA's Address	CoA	CN's Address	MN's HoA	Data

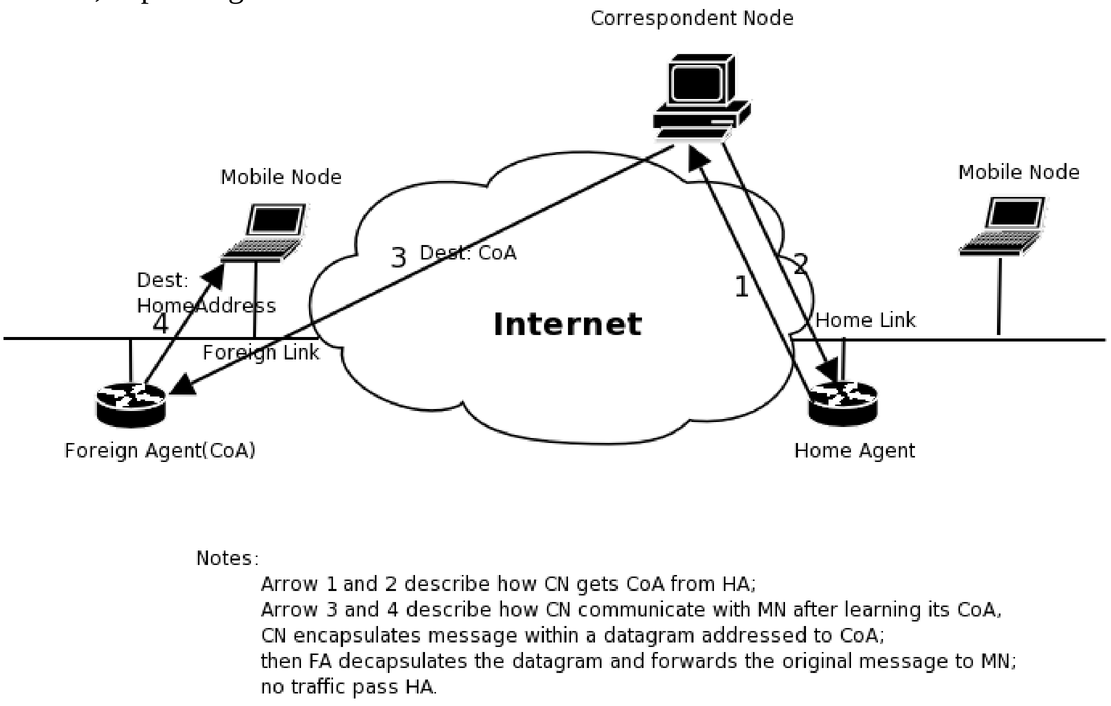
Figure 3: IP-in-IP Format

2.2.2. Direct Routing to a Mobile Node

We notice that with indirect routing, because of the triangular route problem, traffic has to be sent to the HA first, and then forwarded to the mobile node, even when a much more efficient route exists between the CN and the mobile node.

Direct routing is proposed to solve the problem. In direct routing, packets could be sent to mobile nodes without passing the HA. To do that, a FA/MN would register the CoA to all CNs as well, so that CNs could know the new address and create IP messages with this new destination address.

Figure 4 describes the procedure of Direct Routing, in which no triangular route problem exists any more. However, a disadvantage is that more traffic might be created, depending on the number of involved CNs.



Notes:

Arrow 1 and 2 describe how CN gets CoA from HA;
Arrow 3 and 4 describe how CN communicate with MN after learning its CoA,
CN encapsulates message within a datagram addressed to CoA;
then FA decapsulates the datagram and forwards the original message to MN;
no traffic pass HA.

Figure 4: Direct Routing

2.2.3. Limitation of Mobile IP

In MAGNET, a cluster is a mobile network instead of a simple mobile node. We have to deal with the mobility issues of a mobile network.

The traditional works conducted to mobility support, particularly in the Mobile IP working group, are only to provide continuous Internet connectivity and optimal routing to a mobile host, and are not suitable to support a mobile network. This is due to two reasons. Firstly, not all devices in a mobile network is sophisticated enough to run these complicated protocols; secondly, once a device has joined a mobile network, it may not see any link-level handoffs even as the network moves.

2.3. Network Mobility (NEMO)

In the IETF (Internet Engineering Task Force), the **NEMO** working group was created to investigate possible solutions for **Network Mobility,** which covers some of the issues of the cluster mobility in MAGNET.

2.3.1. Mobile Networks

A network with the possibility for mobility is called a **mobile network**. A mobile network could migrate in the Internet topology. The attachment point of the mobile network to the Internet could be changed within one administrative domain or between different administrative domains. Comparing with **Host Mobility** (the mobility of one mobile node), **Network Mobility** describes the situation like a router connecting an entire network to the Internet dynamically changes its point of attachment. The connections of the nodes inside the network to the Internet are also influenced by this movement [NMSGR] [SNMS].

A mobile network can be connected to the Internet through one or more **Mobile Router** (**MR**, the gateway of the mobile network), there are a number of nodes (**Mobile Network Nodes**, **MNN**) attached behind the MR(s). These nodes could be Local Fixed Nodes, i.e., nodes which belong to the mobile network and cannot move with respect to the MR(s), they are not able to achieve global connectivity without the support of the MR(s); or **Local** (its home link belongs to mobile network) and **Visiting** (its home link does not belong to mobile network) **Mobile Node**s, i.e., nodes which belong to the mobile network and can move with respect to the MR(s). It is not realistic to ask each node to individually handle its mobility. The logical way is that only the MR(s) changes the physical point of attachment and handles the mobility of the entire mobile network during the movement of the mobile network. **Figure 5** shows an example of a cluster; and **Figure 6** shows the different mobility situation of a mobile node and a mobile network.

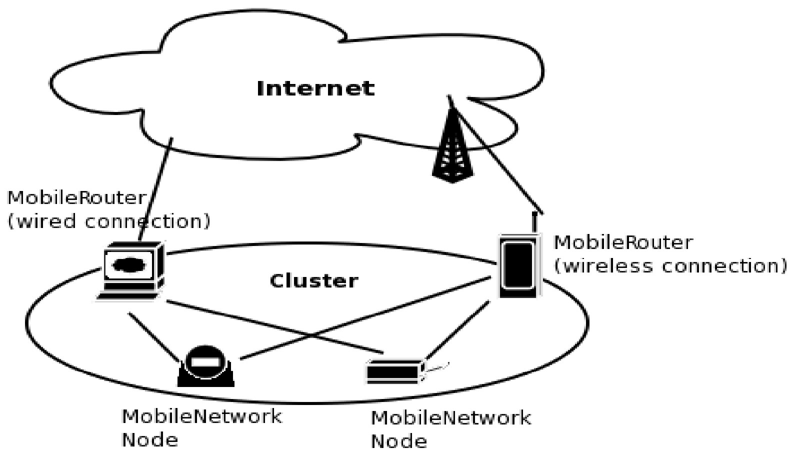


Figure 5: A Cluster Architecture Example

A mobile network can also be nested when a MR allows another MR to attach to its mobile network. The operation of each MR remains the same whether the MR attaches to another MR or to a fixed **Access Router** on the Internet. As the level of nested mobility is unlimited, management might become very complicated. [MNMN]

By using a Network Mobility architecture that relies on the MR, several advantages can be achieved, for instance, reduced transmission power, handoff, software and hardware complexity on network nodes, bandwidth consumption and location updates delays.

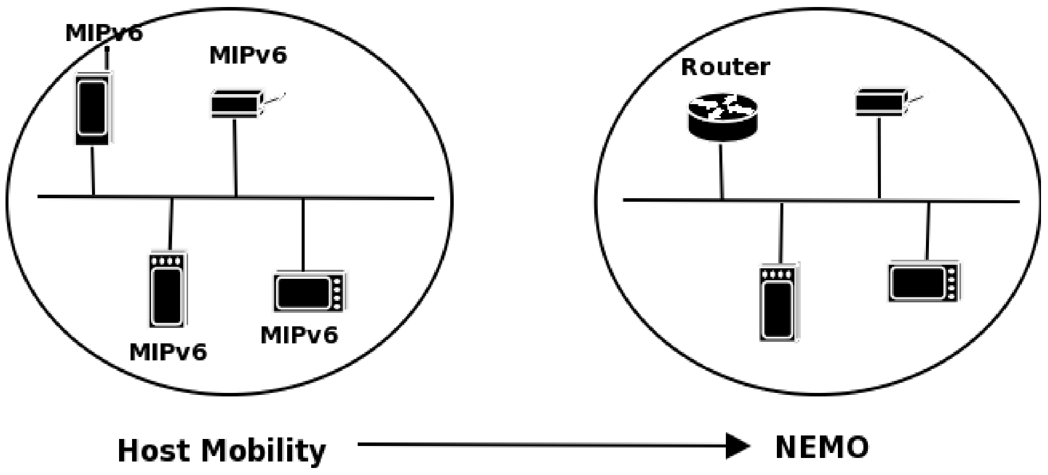


Figure 6: Comparison between Host Mobility and NEMO

More and more devices are produced with the capability of accessing the Internet directly. IPv6 provides enough identification addresses so that the devices could be managed wherever they are. Mobile IP is one solution of maintaining an uninterrupted connection during movement, or to say, providing movement transparency, but not enough to support Network Mobility.

So in NEMO Basic Support, some mechanisms are proposed to allow mobile network nodes to remain connected to the Internet and continuously reachable at all times while the mobile network they are attached to changes its point of attachment. Meanwhile, it would also be meaningful to investigate the effects of Network Mobility on various aspects of internet communication such as routing protocol changes, implications of real-time traffic, fast handover and optimization.

In this thesis, when we talk about **Cluster Mobility** we are considering the cluster moving in its entirety and not separate mobility issue of nodes within a cluster.

2.3.2. NEMO Basic Support Protocol

As mentioned before, NEMO [NEMO Basic Support Protocol, RFC 3963] extends Mobile IPv6 to enable support for Network Mobility. The definition of MR extends that of a Mobile IPv6 mobile node, by adding the capability of routing between itself and the mobile network behind it.

Mobile IPv6, which supports host mobility, has some limitations of supporting Network Mobility. The main problem is that a **Home Agent (HA)** could only forward packets addressed to a MR, but not those nodes behind the MR in the mobile network. So we need a method to register the whole mobile network to the HA [MNAMIP].

To solve the above problem, a new term **Mobile Network Prefix (MNP)** is introduced. It is an IPv6 prefix delegated to a MR and advertised in the mobile network. More than one Mobile Network Prefix could be advertised in a mobile network. Mobile nodes in the network then get their IP addresses within these prefixes. Therefore, if a HA knows these prefixes, it could forward traffic to a mobile network.

We notice the difference between a CoA and a MNP, a CoA is only able to identify one mobile node, while a MNP can be used to identify a group of mobile nodes, namely a mobile network.

As an extension, **Prefix Scope Binding Update** is an enhanced Mobile IPv6 Binding Update which associates a care-of-address with a **prefix** instead of a single address. A node receiving such a Prefix Scope Binding Update could route any packet that shows the prefix address in the destination field via the MR's CoA, thus it ensures that any packets to the nodes in the mobile network would be routed correctly.

Since the MNP is sent to HA, the content of the binding cache at the HA is changed, compared to Mobile IP. Besides the basic (CoA, IP) pair, another (CoA, MNP) pair is added for Network Mobility management.

2.3.2.1. Some Extended Message Formats

1. Binding Update

Its format is described in **Figure 7**. A reserved bit R is used as a flag to indicate whether this is a node binding update or a network binding update. The other fields are the same as defined in Mobility Support in IPv6, RFC 3775[MIPv6].

						Sequence#
A	H	L	K	***R***	Reserved	Lifetime
Mobility Options						

Figure 7: Binding Update message Format

2. Binding Acknowledgement

A reserved bit R is also used here for the similar purpose as in a Binding Update message to indicate whether this is a node binding acknowledge or a network binding acknowledge.

3. Mobile Network Prefix Option

Type	Length	Reserved	Prefix Length
Mobile Network Prefix			

Figure 8: Mobile Network Prefix Option Format

This is a part of the Binding Update to indicate to the Home Agent the prefix information for the mobile network. Its format is shown in **Figure 8**. The important information here is the mobile network prefix and its length.

In NEMO Basic Support, no route optimization is considered. In case of a moving network, the route optimization problem is very hard; while for mobile hosts, the un-optimal path usually has a triangular shape involving a HA, a MH (Mobile Host) and a CN (Correspondent Node), the mobile network shape could be much more complicated due to the effects of nesting and the presence of multiple HAs. Consequently, the route optimization problem for a mobile network leads to multiple triangular paths without an upper bound limitation. Route optimization for a mobile node is performed by sending a CoA to a CN. If we simply use the same method for a mobile network, this kind of additional traffic would be not upper bounded, which might not provide better performance. As a solution, all traffic has to pass through a bi-directional tunnel. Some discussion about Route Optimization could be found in [NROPSA] and [NMROPS].

2.3.2.2. Network Mobility using NEMO

1. MR Operation

A MR can act as a simple mobile host, in which case a HA does not maintain any prefix information related to the mobile host's home address but does maintain the CoA related to the home address; or it can act as a MR, in which case the HA also maintains forwarding information related to the prefixes assigned to the mobile network.

A MR maintains a **Binding Update List**, which records all the information sent in Binding Update messages. This information is used to establish tunnels between the MR and the HA.

A MR has the basic functions of sending Binding Updates, receiving binding acknowledges, error processing, establishing bi-directional tunnels and neighbour discovering for the MR, etc.

A MR obtains one or more Care-of-Address on each foreign link using either stateless or stateful DHCPv6 address configuration [DHCPv6][1]. It sends its Prefix Scope Binding Update message to its HA when moving to a foreign domain, and as a result, a tunnel connecting the MR and HA is set up. When the MR receives packets encapsulated by the HA, it would forward them to the nodes in the network. On receiving packets created by nodes in the mobile network, the MR would encapsulate them and forward them onto one tunnel according to some routing rules.

2. HA Operation

As a HA cooperates with a MR, it is able to deal with the two types of MR behaviours (a mobile host or a MR).

A HA maintains a **Binding Cache** for each MR which is currently registered. The format of the Binding Cache is the same as the Binding Update List in the MR. A Prefix table is used to prevent a MR from claiming MNPs belonging to other MRs. The Prefix table is checked while the HA receives a Binding Update.

Once a HA receives a binding update, it performs the validity check. If the binding

[1] Stateless Autoconfiguration is a method defined to allow a host to configure itself without help from any other devices; Stateful Autoconfiguration is a technique where configuration information is provided to a host by a server.

update passed, the HA would create a new entry in its Binding Cache according to the information in the binding update message and send back the binding acknowledgement.

When receiving a packet sent by a CN, the HA would compare the destination address with these entries in its binding cache (the full IP address or the MNP). If the destination address is matched with an entry, the corresponding CoA would be returned. As a result, the packet is forwarded to the CoA of the MR. When receiving a message from the tunnel, the HA would remove the outer IP header and forward the packet on, if necessary.

So with the proposed extension, when a MR and its mobile network move to a foreign domain, the MR would register its care-of-address (CoA) with its HA for both MNNs and itself. An IP-in-IP tunnel is then set up between the MR and its HA. All the nodes behind the MR would not see the movement, thus they would not have any CoA, removing the need for them to register anything at the HA. All the traffic would pass the tunnel connecting the MR and the HA. **Figure 9** describes how NEMO works.

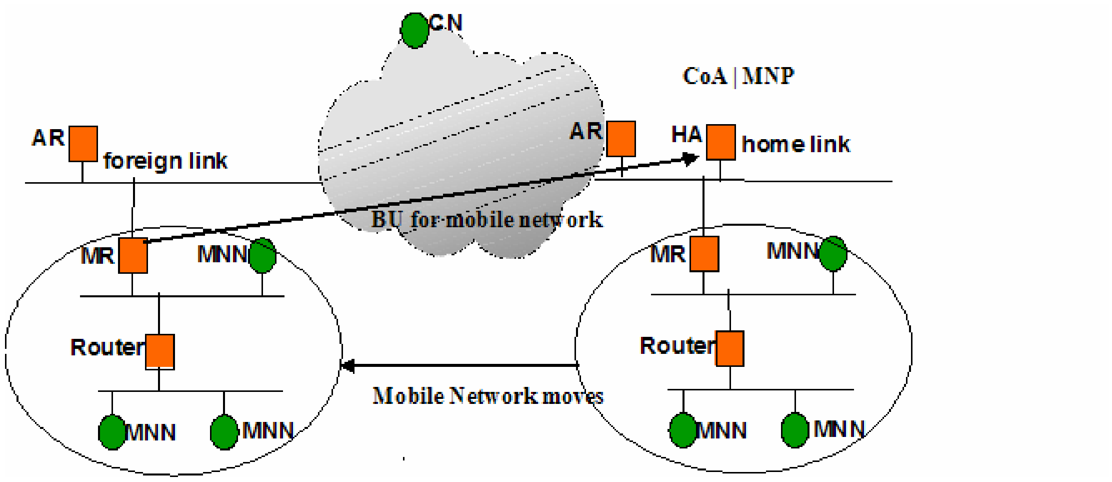


Figure 9: Network Mobility

3. Multihoming support in NEMO

NEMO Basic Support is simple since it is a logical extension of the MIPv6 operations at the **Mobile Router**s (**MR**) and their **Home Agent**s (**HA**). However, the practical management of a mobile network depends on the ability of providing other support mechanisms. One of them is multihoming support.

In this section, we are going to describe the definition of multihoming, its classification, and its issues.

3.1. What is Multihoming?

A mobile network is said to be **multihomed** when it could be connected via several interfaces, several access technologies and distinct access networks. In NEMO terms, a mobile network is considered multihomed when either the mobile network is simultaneously connected to the supporting infrastructure via more than one MR, or via only one MR which has multiple egress interfaces.

Different interfaces may indeed be active simultaneously. A mobile network must be able to deal with both horizontal handover (between access points using the same communication medium, which happens when the mobile network moves to a foreign link) and vertical handover (between distinct communications medium, which might happen without the movement of the mobile network, for instance, one connection is broken for some reason and its traffic should be handed to other available connections).

3.2. Why do we need Multihoming?

A road with multiple lanes is a real life example of multihoming. Redundancy happens when there is a car accident in one lane and the following cars could take the other lanes rather than waiting. Load Sharing is carried out when a car arrives at the road entry, and selects a lane with little traffic; as a result, it would cost shorter time to get to the end of the road. Preference Setting means we specify the fast lane and slow lane (like bus lane), different kinds of vehicles use different lanes; this avoids the occurrence of there is a slow-speed vehicle in each lane and the high-speed vehicles can not move fast but as slow as the slow-speed vehicle which is in front of it. [GBM] lists the goals and benefits of Multihoming.

For a multihomed network, the lanes are there but not visible; no cars on the lanes but data packets.

The benefits for a mobile network of using the “multihomed” property can be briefly summarized as:

- Fault-Tolerance/Redundancy, where the connectivity is maintained even in the event of a failure to a primary connection. It could be split into two sub-classes:

i. Without Transparency: The ongoing transport sessions are broken in the case of the fail of one connection, and a new connection would be established to support the data transportation;

ii. With Transparency: The ongoing transport sessions are not broken even in the

case of the fail of one connection. For instance, in the cases where only one **Mobile Network Prefix (MNP)** is advertised, as all the packets are sent from/to the same ingress interface of the MR, redundancy could be performed transparently.
- Load-Sharing, where traffic load of the mobile network is simultaneously spread over several routes.
- Policy-based Routing, where the route to the mobile network is decided upon by a user-defined metric, such as cost, efficiency, guaranteed quality, etc. This policy could be defined statically or dynamically, and initiated either by a MR, the mobile network's HA, or by a node within the mobile network itself.

In order to understand it sufficiently, we focus on the detailed discussion of multihoming support benefits. We mainly describe those aspects: Redundancy, Load Sharing and Preference Setting. These benefits would also be illustrated in the simulation part in Section 7.

3.2.1. Redundancy

Since no multihoming support is specified in basic NEMO, when the used connection goes down, the MR has to start a whole procedure of establishing a new connection, which includes the following sub procedures:

Old Access Router Unavailable Detection, this can be performed by receiving a new Router Advertisement which is different from the current one, or not receiving any old Router Advertisements within a given time period.
Router Discovery, which is achieved through the receipt of a router advertisement sent from a new Access Router. A MR might receive a router advertisement without sending any request, or it can start to send a router solicitation asking the routers to send a router advertisement. In the first situation, Old Access Router Unavailable Detection can be regarded as a part of the Router Discovery.
Care of Address Configuration, which is performed by the MR configuring itself with an IPv6 address to be used on the new network. To obtain a care-of address, a MR can use either stateful or stateless Address Autoconfiguration. In a stateful configuration, the MR gets a care-of address directly from an address server like a DHCP server; in a stateless configuration, the MR extracts the network prefix from the Router Advertisement and forms its care-of address using its MAC layer address, in this case, Duplicate Address Detection has to be performed.
Duplicate Address Detection, to perform this, the MR has to send out a Neighbour Solicitation (NS) message with its new CoA as the target address of the solicitation message.
Other configurations might include the AAA establishment and QoS requirements.
After these procedures, a MR also has to inform its HA of its new location. This is performed by sending BU and receiving BA.

As a conclusion, the total handover delay may be represented by **T** defined as the sum of the aforementioned latency components as follows:

T = **Movement Detection Tim**e (Old Router Unavailable Detection, Router Discovery) + **IP CoA configuration Time** + **Context Establishment Time** + **Binding Registration Time**

The whole period would take around 1 or 2 seconds, which differs for different cards and Access Point models. The time sequence chart is depicted in **Figure 10**.

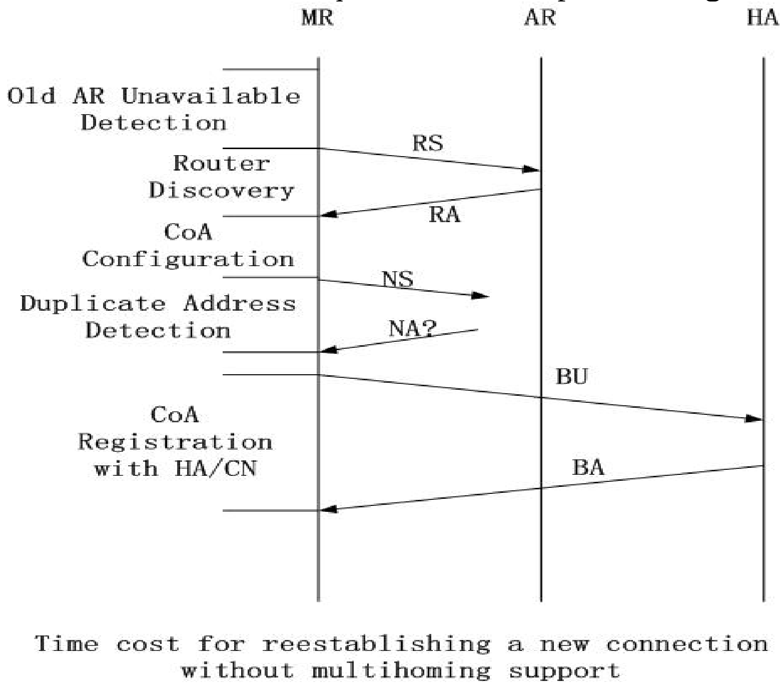

MR
AR
HA
Old AR Unavailable Detection
Router Discovery
RS
RA
CoA Configuration
NS
Duplicate Address Detection
NA?
BU
CoA Registration with HA/CN
BA
Time cost for reestablishing a new connection without multihoming support

Figure 10: New Connection Establishment without Multihoming Support

In the case of multihoming NEMO, since it provides the method of maintaining multiple interfaces by dynamically recording the situation on each interface, it makes handover simpler and faster. When one connection goes down, we already know which interfaces are still available, and we could pick up one interface according to some policy. Thus, the handover procedure discussed in the case without multihoming support is avoided. That is a great benefit, since handover time greatly influences the mobility performance.

By sending some information to the HA, the suitable connection for inbound traffic can be chosen. This part is similar to the configuration part of CoA Registration in basic NEMO. Because no other configuration processes are operated, the time used for changing to a new connection is much less than before. It is extremely significant for providing good service to users, especially for some traffic types like TCP. The corresponding time sequence chart is shown in **Figure 11**.

3.2.2. Load Sharing

In basic NEMO, the network traffic load is totally put on one route. If the load is light, that it all right, but if the load is too heavy to handle, that would result in a high congestion and huge data loss.

With multihoming support, several interfaces can be used to spread the traffic of the mobile network. On one hand, this implies the choice of the IPv6 address to use for each flow and the ability to choose a different address for each flow; on the other hand, it enables choosing the less loaded connection or according to preference on the mapping between flows and interfaces.

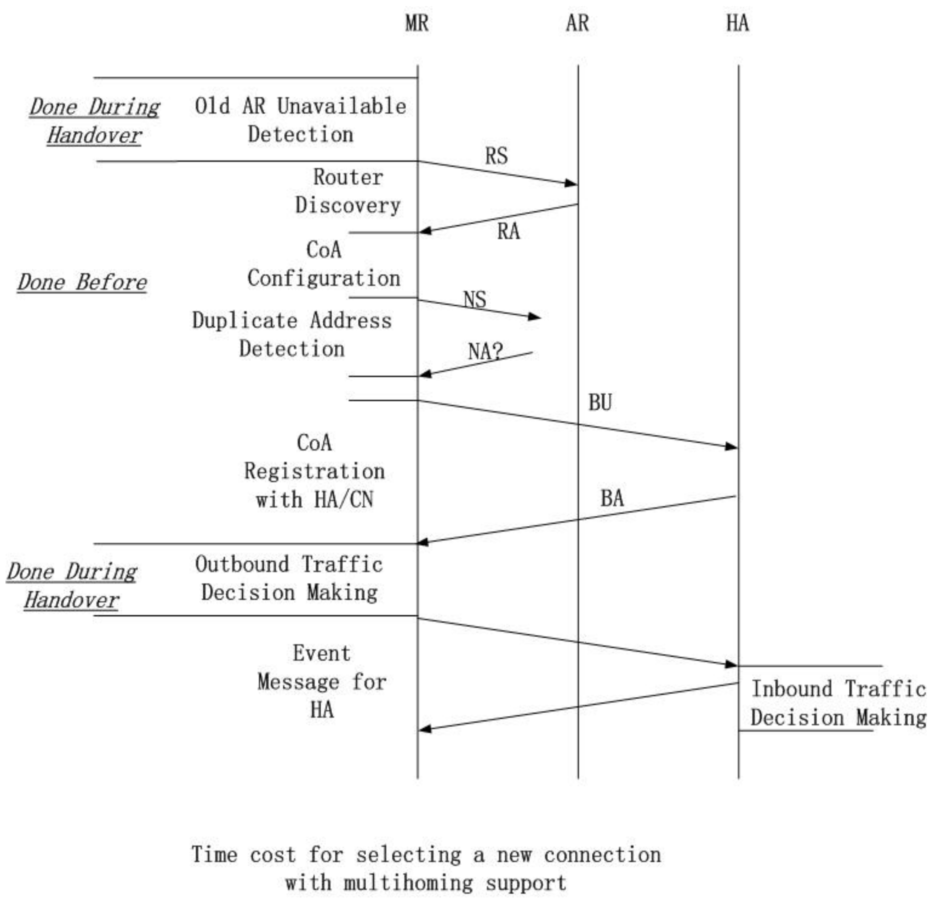

MR
AR
HA
Done During Handover
Old AR Unavailable Detection
RS
Router Discovery
RA
CoA Configuration
Done Before
NS
Duplicate Address Detection
NA?
BU
CoA Registration with HA/CN
BA
Done During Handover
Outbound Traffic Decision Making
Event Message for HA
Inbound Traffic Decision Making
Time cost for selecting a new connection with multihoming support

Figure 11: New Connection Establishment with Multihoming Support

3.2.3. Preference Setting

Preference Setting could be achieved while using multiple interfaces. It would provide the user, the application or the ISP the ability of choosing the preferred transmission technology or access network, for the matters of cost, efficiency, politics, bandwidth requirement, delay etc. It is clear that providing preference setting would make traffic control more flexible.

3.2.4. Bi-Casting

To ensure the delivery of packets on the node, bi-casting can be performed. It duplicates a particular flow and sends them out simultaneously through different routes. Just like we always back up the important files in case of loss, bi-casting minimizes packet loss typically for real-time communication and burst traffic. It also decreases delay of packet delivery caused by congestion and achieves more reliable real-time communication than single-casting.

Some work on benefits of multihoming is referred to [LSSPMMR].

3.3. Some Multihoming Protocols

Multihoming protocols could exist in any layer from application layer to network layer, so far there are several protocols implemented on network and transport layer, which are transparent to application [MHomeProtocol].

They include:

1) LIN6 (Location Independent Networking for IPv6)[LIN6]: The basic idea is that each host has a 64bit globally unique identifier, called LIN6ID, which is presented in IPv6 interface identifier portion of an IP address used by a host. A host can be uniquely named by prefixing its LIN6 ID with the LIN6 prefix, which is a non-routable IPv6 reserved by LIN6, resulting a so-called LIN6 Generalized ID (GI). Since GI is globally unique, Multihoming is supported through allowing a single GI to be associated with several real addresses.
2) MHTP (Multi Homing Transport Protocol) [MHTP]: it is proposed for IPv6 network layer multihoming and targeted for site multihoming. The main idea is that multihomed traffic is transformed into single-homed traffic at a router close to the source and transformed back into multihomed traffic at the last router being the MHTP prefix.
3) Multihomed TCP [MTCP]: it uses a separate 32 bit connection identifier rather than IP address to identify connections. Addresses can be updated dynamically during communication, enabling multihomed mobility.
4) Stream Control Transmission Protocol [SCTP]: it is a reliable transport protocol operating on top of the network layer where both of the end points may be presented by multiple IP addresses. A host has one primary address and several alternative addresses. The primary address is used by peer hosts as the destination address for all packets in normal data transmission. Alternative addresses are used for retransmitting packets.
5) Flow-based mobility: Per-flow movement in MIPv6 [FlowMo] proposes an extension to Mobile IPv6 by containing flow IDs in the binding updates. It enables that different flows traverse through different interfaces.
6) Homeless Mobile IPv6 [HMIP]: connections are presented by hosts themselves with a set of IP addresses instead of host interfaces. Technically, it removes the difference between the home address and the care-of-address. It does not require home addresses or HA anymore, but still allow them to be used as in MIPv6.
7) Host Identity Payload and Protocol [HIP]: it describes a new space called Host Identity (HI), which completes the IP and DNS name spaces. Host Layer Protocol located between the IP and transport layers is required. HI is used to identify the connection, while IP address is only used for routing information.

These protocols provide some methods to identify multiple connections for a variety of scenarios, but not for a mobile network.

3.4. Classification of Multihoming with Network Mobility

The NEMO Basic Support does not specify any particular mechanism to manage multihoming, but it does not prevent the use of Multihoming either. Multihoming is still an open topic.

Three multihoming classification approaches are proposed by NEMO [MultiClass]. They are, namely, (1) the Configuration-Oriented Approach; (2) the Ownership-Oriented Approach and (3) the Problem-Oriented Approach. Here we only describe the Configuration-Oriented Approach.

According to the Configuration-Oriented Approach[COClass], multihoming could be classified depending on how many MRs are present, how many egress interfaces, **Care-of-Address** (**CoA**) and Home Address the MRs have, how many Prefixes(MNP) are advertised to the mobile network nodes, etc. Three key parameters are used to differentiate the multihomed configurations. Each configuration would be identified by a three tuple (x,y,z), where the meanings of 'x', 'y' and 'z' are described as followed:

I. 'x' indicates the number of MRs of this mobile network:
x=1 means there is only one MR in this mobile network, and the MR is multihomed by having more than one egress interface.
x=n means there are more than one MR in this mobile network, each one could be multihomed or non-multihomed[1].

II. 'y' indicates the number of HAs associated with this mobile network by the tunnels"
y=1 means there is only one HA which sets up the tunnels with the MR(s) of the mobile network.
y=n means there are more than one HA assigned to this mobile network.

III. 'z' indicates the number of MNPs announced to **Mobile Network Node**s (**MNN**):
z=1 means there is only one MNP advertised to the MNNs. Each MNN has one IP address.
z=n means there are more than one MNP advertised to the MNNs. Each MNN has multiple IP addresses.

So by identifying the configuration with the above three orthogonal parameters, eight kinds of possible configurations are formed. Each type of configuration has its constraints on implementation and might need other extra control mechanisms, thus we could only say which configuration is suitable for which scenario rather than which one is the best.

Requirements of each configuration are discussed from the perspective of achieving multihoming benefits:

I. (1,1,1) Single MR, Single HA, Single MNP

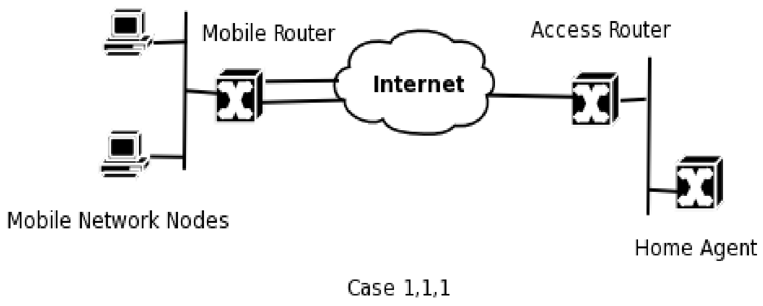


Figure 12: (1,1,1) Configuration Architecture

Figure 12 describes this configuration. This is the simplest class, where only one MR is associated with one HA and only one prefix is advertised in the mobile network.

[1] A Mobile Router is called multihomed when it has multiple interfaces.

As there is more than one tunnel (several interfaces could be used simultaneously) available between the MR and the HA, load could be shared simultaneously by these tunnels. Each interface might register itself into the HA on behalf of same mobile network. Thus several entries with same MNP and diverse CoA would appear in the Binding Update Table.

If all interfaces of the MR share one home address, a mechanism for identifying which CoA a Prefix-Binding Update is going to update is required. We could allocate a permanent address so called HoA for each interface as its identifier. Hence, modified binding update table is comprised of three items: (CoA, HoA and MNP); HoA and MNP is the index of the table. In [MCARMIP], this problem is discussed in detail.

Traffic redirection provides transparent handover. A MR could use a new CoA to handle the traffic, which was handled by the old interface, namely, I1. The MR sends a new Binding Update from the new interface, namely, I2, with the home address field of this Binding Update set to the IP address bound to the previous I1, the CoA field set to the new IP address bound to the new I2. Then the HA adds a new entry to its Binding Cache so that traffic destined to I1 is redirected to I2 successfully.

II. (1,1,n) Single MR, Single HA, Multiple MNP

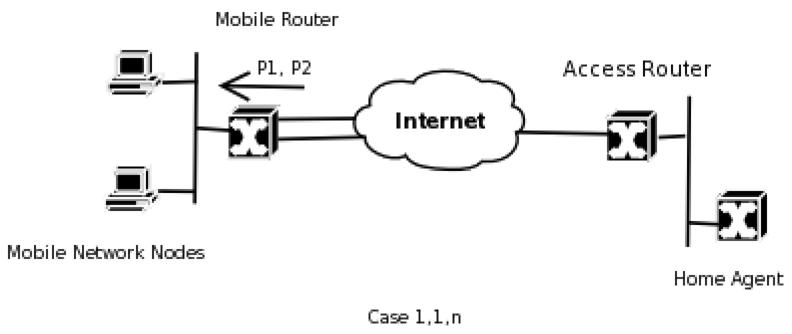


Figure 13: (1,1,n) Configuration Architecture

From **Figure 13**, we can see that more than one MNP are advertised to the mobile network, so that each MNN gets more than one global address. Source Address Selection Function is necessary for MNN.

Due to the multiple IP addresses on one node, when one of them is unavailable, another address built with another prefix should allow it to recover this sort of failure; however, transparency might not be achieved since the on-going session using the old IP address would have to be determined. In order to avoid this, a **recovery mechanism** is needed to redirect all current communication to its possible new address.

In this case, because a variety of IP addresses could be chosen, the benefits of load sharing are mainly for the network side: if different routers or different routes (because of using more than one destination IP address which identify the same MNN) could be used to forward the node traffic, it will share the traffic load on the network.

III. (1,n,1) Single MR, Multiple HA, Single MNP

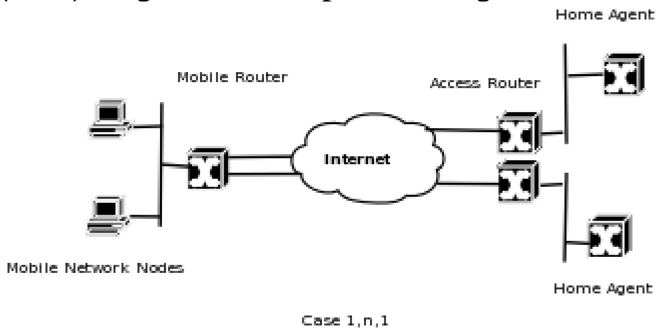


Figure 14: (1,n,1) Configuration Architecture

In this case shown in **Figure 14**, as there is more than one HA, the MR has to send the Prefix-BU to each HA to establish a tunnel on each of its multiple egress interfaces with different HAs.

One benefit of using more than one HA is to avoid the traffic bottleneck in the **HA** caused by using only one HA. At last, the load of mobile node registration information and the tunnelled data traffic information are balanced. One potential problem is whether a MR can register the same home-address to different HAs.

The traffic burden would be shared on not only multiple paths between the CN and multiple HAs, but also the multiple paths between the MR and multiple HAs. But it is possible that only one active HA will advertise a route to the MNP, and then load-sharing in multiple HAs could not be fulfilled.

IV. (1,n,n) Single MR, Multiple HA, Multiple MNP

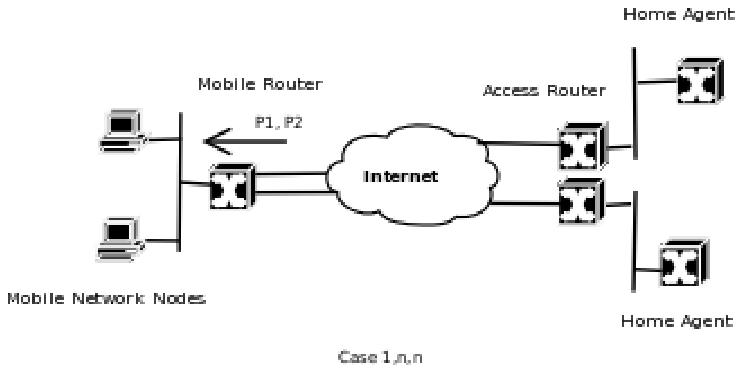


Figure 15: (1,n,n) Configuration Architecture

One potential problem for the configuration type presented in **Figure 15** is that it is hard to keep the sessions alive when one tunnel is broken if one prefix is registered only on one HA, cause the influenced MNP may not be advertised in other HAs, MNN must choose another IP address, which causes the old session broken.

If the multiple tunnels could be used simultaneously(only if one MNP is registered in multiple HAs), the outbound traffic could be shard; if the CN sends packet to one MNN via its multiple IP addresses, the inbound traffic could be shared on multiple tunnels too. However the **ingress filtering problem [IFMN]** has to be solved.

V. *(n,1,1) Multiple MR, Single HA, Single MNP*

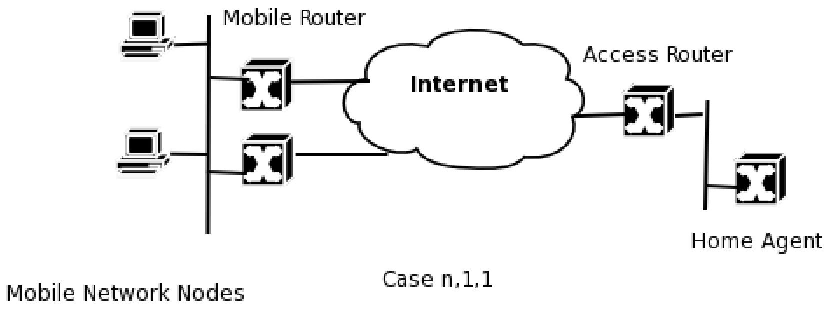


Figure 16: (n,1,1) Configuration Architecture

In **Figure 16**, one bi-directional tunnel is established between each MR and the HA. So there are at least n tunnels between the mobile network and the HA (in the case when each MR only have one egress interface), if one tunnel or one MR is broken, other tunnels could be used.

The traffic could be shared on these tunnels if only those MRs could work simultaneously.

Potential Problems (common problems for the next three cases with multiple MRs) :
A mechanism for all MNNs to select its default router is needed.
A mechanism of quickly changing the mobile network's default router in the case of egress interface failure is needed.
And a mechanism for the failed MR to deregister quickly towards its HA when the MR notices the failure is also needed.

It should be noted that for all the next four cases, the distribution of outbound traffic is thus no longer operated by the MR between its multiple CoAs (in the case of only one MR) but by the MNN between multiple MRs. Each MNN should choose through which MR it wants to send its packets by using the router selection mechanism.

VI. *(n,1,n) Multiple MR, Single HA, Multiple MNP*

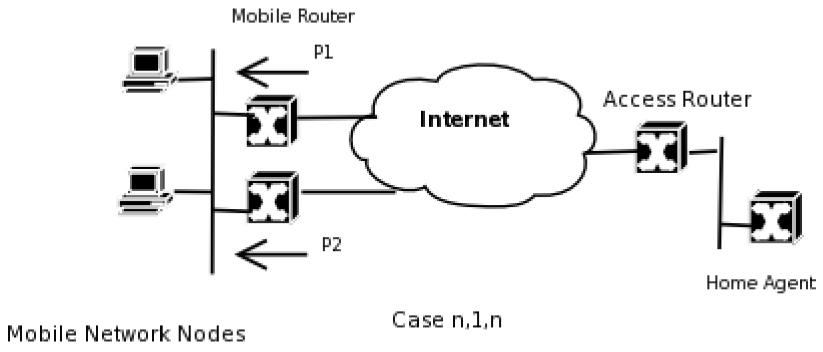


Figure 17: (n,1,n) Configuration Architecture

In the architecture configured as **Figure 17**, more than one tunnel are established between multiple MRs and the HA, if one MR or one tunnel is broken, another MR could be used.

One MNN could get more than one source address, so address selection function is required. And of course, the failure detection function and redirection of communication is needed for redundancy.

If the multiple MRs could be used simultaneously, the outbound traffic could be shared via the management from MNNs and the inbound traffic could be shared by the management from the HA. **Ingress filtering problem** should be considered as well.

VII. (n,n,1) Multiple MR, Multiple HA, Single MNP

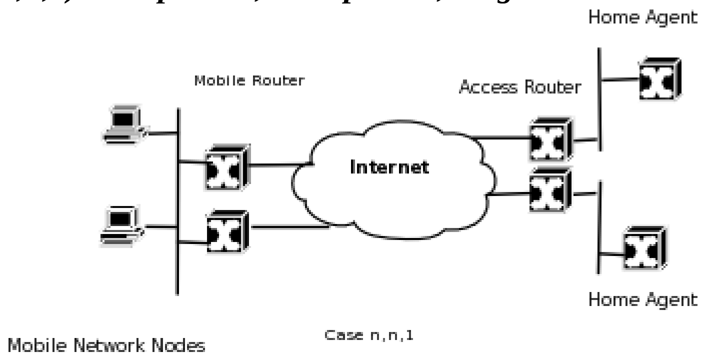


Figure 18: (n,n,1) Configuration Architecture

The combination of case (1,n,1) and case (n,1,1) is shown in **Figure 18**. Because the same MNP must be advertised to MNNs by different paths, one problem is *How could these HAs delegate the same prefix to different MRs?*

VIII. (n,n,n) Multiple MR, Multiple HA, Multiple MNP

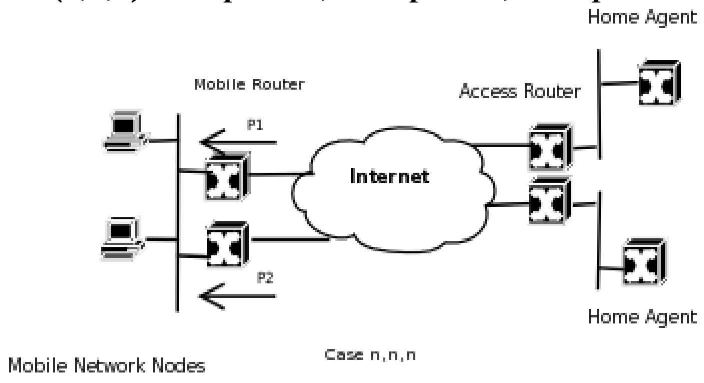


Figure 19: (n,n,n) Configuration Architecture

Figure 19 describes the most complicated case, as one MNN could communicate with a CN through different MRs, different HAs with different IP addresses. However, in some cases, each MR takes care of one specific MNP and is registered to an individual HA, when one MR or HA or the tunnel between them goes down, the MNNs using the MNP which is mapped to the tunnel have to choose a new IP address corresponding to the new MNP advertised by a new MR from a new HA. The management about load sharing is tough too.

Conclusion of the eight scenarios:

I. For all the cases in which the number of MNP is larger than 1, because the MNN could choose its own source address, if the tunnel to one MNP is broken, related MNNs have to use another source address which is created from another MNP. In order to keep sessions alive, both **failure detection** and **redirection of communication mechanisms** are needed. If those mechanisms could not perform very well, the transparent redundancy can not be provided as well as in the cases where only one MNP is advertised.

II. Ingress filtering problem: As both the MR and the HA could perform the Ingress Filtering function by using some routing algorithm like RPF (Reverse Path Forwarding)[1], the MR or the HA might refuse to forward the packet with a different source prefix other than the MR advertised MNP for security reasons. In all the cases in which the number of MNP are larger than 1 and at the same time the number of MR or the number of HA or both is larger than one, a mechanism for solving the ingress filtering problem should be used. A proposed solution is to create a second binding on the ingress interface by sending a Prefix-BU through the other MRs and then the HA(s) get(s) all other CoAs.

III. Two levels of redundancy can be seen in these cases: one is that the mobile node's IP address is not valid any more, and the solution is to use another available IP address; the other is that the connection through one interface is broken, and the solution is to use another Internet connection through another interface.

IV. For all the cases, in which a single MNP is registered to one HA with multiple CoAs, a mechanism of distinguishing those CoAs is needed. One solution is to use an extra identifier for different CoAs and include the identifier information in the update message. This kind of situation exists a lot, except for the cases in which one MNP is only allowed to be controlled by one CoA.

V. The difference between using only one MR with multiple egress interfaces and using multiple MRs each of which only has one egress interface is that, multiple MRs could share the processing task comparing with only one MR, and of course it provides the redundancy of the disrupting of the MR. However, on the other side, other mechanisms for managing on and cooperating between these MRs are needed.

VI. Similarly, if there are multiple HAs, the redundancy of the HA is provided, if one HA is broken, another one could be used.

VII. For the last four cases described above, in which there is more than one MR, the analysis of redundancy and load sharing is based on the assumption that each MR only has one egress interface. Of course, the MRs in these cases could have multiple egress interfaces, and if that is the case, the mobile network would have all the redundancy and load sharing abilities mentioned in the case (1,1,1).

Further information could be found in [EMSNBS], [AMNMS], [MHIBT] and [EMMRMNPN].

3.5. Policy Issues of Multihoming

A mobile network being multihomed due to a multihomed MR is the simplest

[1] The reverse path forwarding algorithm is a protocol for distributing messages throughout networks. The intention is to preserve correctness -messages sent will eventually be received by all nodes in the originator's connected component- whilst minimising the number of propagations of each message. The RPF algorithm allows a router to accept a multicast datagram only on the interface from which the router would send a unicast datagram to the source of the multicast datagram.

multihoming configuration. This case could be handled by the solutions proposed for host mobility. Multihoming support of NEMO provides the benefit of policy-based routing. Policies could be set for multiple interfaces of one MR or multiple MRs. As an award, when implementing diverse applications, users have a more flexible choice on access technologies, which offer different QoS, security and so on. Apart from these, it is necessary to consider accounting issues that arise inadvertently. In order to properly match user requirements with dynamic interface situation and take good use of different interfaces, routing policies have to be designed carefully for mobile networks.

In Section 5, we are going to investigate the reasonable mobile network policy-based routing framework for the configuration case (1,1,1). Some policies could also be performed in static scenarios besides mobile environments.

There were a lot of discussions about policy-based vertical handover, which is implemented on mobile devices. It is different from the policy-based routing discussed in this paper. Vertical handover policy is solely used to decide when to hand access over to another network by using a new CoA; our Routing Policy is composite, considering handover, as well as traffic (both inbound and outbound) sharing on multiple tunnels.

4. Policy-based Routing

To obtain a good understanding of policy-based routing, a Policy-based Management System is introduced in this section, including its components and structure. Benefits of policy-based routing and potential information used by interface selections are also briefly introduced.

4.1. Policy

Policy-based management provides the ability of controlling a system according to some rules specified by the users or the applications.

A policy is basically an event, condition and action construct. One policy example could be “if the data loss on one link is larger than 30%, then stop using this link until its data loss is lower than 10%”. Policies could be implemented in several layers, such as, IP layer, session layer or application layer. It could also be classified in terms of different goals. One policy type is obligation policies, which could be used to define adaptable management actions, such as changing quality of service; another type is authorisation policies, which could be used to define what services or resources a subject could access; in addition, security management policies are needed to define the actions to be taken when security violation is detected.

4.1.1. Policy-based Framework Components

A policy-based management system is one in which its operation is determined by a set of rules and instructions. Policies define choices in behaviour in terms of conditions under which predefined operations or actions can be invoked rather than changing the functionality of the actual operations themselves. Policy rules are evaluated when triggered by an event.

The Internet Engineering Task Force has defined a policy framework (RFC 2753, RFC 3198)[POLICY] within which sets of policy rules described in the form of policy models are transformed into network device configurations in an administrative domain. In that framework, two main architectural elements for policy-based control are PEP **(Policy Enforcement Point)** and **PDP (Policy Decision Point)**.

The **PEP** is the component that actually encounters the packets and is responsible for enforcement and execution of policy actions, such as filtering, packet marking, rate enforcement, shaping, resource management, etc. It would typically be co-located with the packet forwarding component of an Access Router or a network server [APFIDSI].

The **PDP** is the component that is responsible for determining what actions are applicable to which packets. The PDP interprets policy rules for one or more PEPs based on information contained in data or signalling packets, current network conditions, as well as dynamic information such as account balances, dynamically allocated addresses, etc. As an example, the PDP could decide whether a specific reservation request ought to be honoured or rejected depending on the identity of the originator of the request. The PEP may query the PDP to make decisions on its behalf on the occurrence of specific events, such as the arrival of a new reservation request or a data packet.

The **Policy Repository** is the location where the policies defined for the domain are stored. The repository may be located in a single physical site within the policy domain, or replicated at several devices. The repository could be a database, a flat file, an administrative server or a directory server. Policies can be stored in the repository by means of a policy management tool [NPL-ASNA] [ASPSA]. A policy management tool can also provide functions that validate whether policies stored in the repository are well-formed, mutually consistent and can be satisfied by the network.

The policy rules stored in the Policy Repository can be retrieved by a PDP in response to events triggered by an **Environment Detector,** which is defined as the component of detecting all kinds of changes that might trigger an event. The PDP translates the acquired policy rules into a set of actions based on the capability of the PEP and the current network conditions. The PEP then executes these PDP-supplied actions to handle the triggering policy events in accordance with required service.

4.1.2. Policy-based Framework Structure and Conflicts

Since the start-up of a system, environment detectors work all the time to detect all kinds of relevant events. Two types of actions might happen, once an event is detected. The environment detector might send a notification or message which requires a policy decision to a PEP. The PEP would formulate a request for a policy decision and sends it to the PDP. All useful information for making decision could be added to the request. Then the PDP would access to its Policy Repository and probably other databases, and make the decision. The policy decision would be sent back to and enforced on the PEP. This is called “PULL”. The alternative type is “PUSH”, in which the message created by the environment detector is sent to a PDP and the PDP would make the decision and send it to the PEP.

Environment detectors can be located on in the same device with a PEP or other devices. PDPs and PEPs in the framework could be located on one device, or distributed located on multiple devices. And it is also possible for one type of component to be implemented on to different devices. For instance, a PDP could be implemented on a centralized Policy Server, which is responsible for implementing policies of a wide scope on behalf of multiple network nodes in an administrative domain, and at the same time, some policies which only depend on information and conditions local to one node might be implemented locally. The two types of architectures could be described by two charts, **Figure 20** shows the centralized architecture; **Figure 21** is the distributed case.

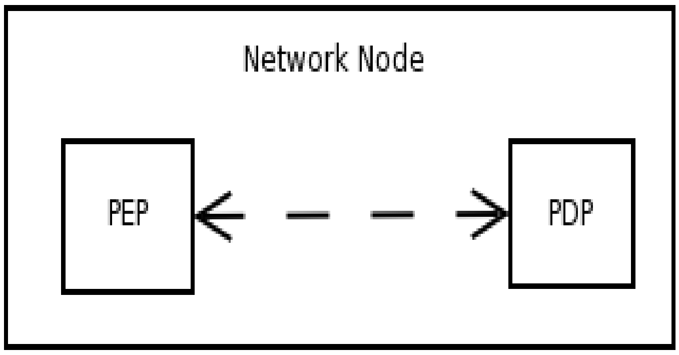


Figure 20: Architecture with co-located PDP and PEP

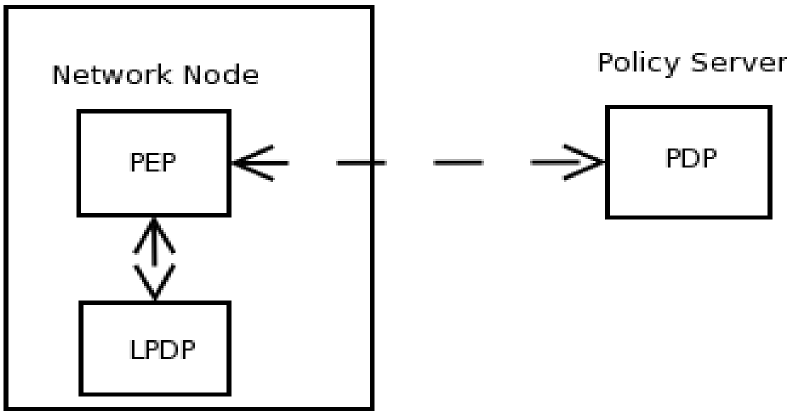


Figure 21: Architecture with distributed PDP and PEP

With regard to the distributed architecture, besides the major components mentioned before, there are two communication protocols used: COPS (Common Open Policy Service) [COPS] and LDAP (Lightweight Directory Access Protocol) [LDAP]. PDPs and PEPs exchange information by using COPS, which is a query and response protocol between a policy server (PDP) and a client (PEP). COPS obtains the reliability by using TCP as its transport protocol. Port number 3288 is used on the server side. The PEP is responsible for initiating a TCP connection to a PDP. Then the PEP uses this TCP connection to send policy requests to and receive policy decisions from the remote PDP. In other words, it employs the client/server model. COPS is also extensible, such as being used for general administration, configuration, and policy enforcement [POLICY].

LDAP is the other protocol, which is "a standardized TCP/IP protocol for access to a central X.500-based directory that is shared by many different services.” The PDP uses LDAP to retrieve policy information from the Policy Repository.

Conflicts can arise in the set of policies. Different policies might make opposite decisions, especially in large distributed systems. There is limited value in developing policy-based management models that do not provide ensuing support for both policy-based conflict detection and resolution. There are static and dynamic policy-based conflicts. We should make distinction between these two classes of conflicts as the process of conflict detection and resolution is frequently computationally intensive, time-consuming and hence, expensive and is most preferably performed statically, at compile time. Some conflicts could only be detected at run time, a series of algorithms for conflict detection need to be designed and implemented. Following conflict detection, conflict resolution has to communicate with conflict detection to obtain specifications of conflicts, decide when it is appropriate to resolve the conflict and how to resolve the conflicts.

4.2. Policy-based Routing

Policy-based routing is the process of forcing packets to take a certain route, often different from the default route, based on some policy rules such as certain packet attributes (source, type of packets, interface, etc.).

Policy-based routing provides a routing scheme which is beyond the ability of traditional "best-effort" routing protocols. For instance, it allows the routers to route traffic from different users through different Internet connections. In other words, it allows routing based on the source information of the packets, instead of the destination information used by traditional routing protocols. It can also support QoS, for example, it can support DiffServ by using packet marking, which differentiates traffic by setting the DS (Differential Service) or TOS (Type of Service) fields in the IP header at the edge routers of the network. Then the routers in the core of the network can treat the packets differently based on the marking.

By implementing policy-based routing in a network, some benefits can be achieved. They are [Policy-Based Routing, white paper by Cisco Systems]:

1) **Source-Based Transit Provider Selection**: Internet Service Providers and other organizations can use policy-based routing to route traffic originating from different sets of users through different Internet connection across the policy routers.
2) **Quality of Service (QoS)**: Organizations can provide QoS to differentiated traffic by setting the precedence or type of service values in the IP packet headers at the periphery of the network and leveraging queuing mechanisms to prioritize traffic in the core or backbone of the network.
3) **Cost Savings**: Organizations can achieve cost savings by distributing interactive and batch traffic among low-bandwidth, low-cost permanent paths and high-bandwidth, high-cost, switched paths.
4) **Load Sharing**: In addition to the dynamic load-sharing capabilities offered by destination-based routing, network managers can implement policies to distribute traffic among multiple paths based on the traffic characteristics.

4.3. Interface Selection

Having a central Internet connection for multiple private networks can require policy-based routing. The need for policy-based routing arises for multi-homed cases, which can not use a single default route.

Interface selection is a term that could be used for local routing of packets through local interfaces in a multihomed host. Routing can be based on a connection association and other information like QoS.

Traditionally routing decisions are made on a hop-by-hop basis solely upon the destination of the packet. The mostly used mechanism is to search for a matching entry for the destination in the routing table using a technique known as longest prefix match. However, for a multihomed host, this is inadequate, since other information might be taken into account when selecting interfaces for outgoing traffic too. Mainly, that information contains:

1. **Link Layer Information**
   The information from Link Layer, like signal quality and other metrics, might play an important role while deciding which interface to use. To be able to make smooth and intelligent handoffs, link quality must be monitored constantly and information must be made available for IP layer and user application. Most of currently available wireless network devices support link layer information gathering.
2. **IP Layer Information**
   From IP header, some information could be retrieved, for instance, source and destination addresses, flow type etc. Also some other metrics could be retrieved from some IP extension headers.
3. **Network Service Information**
   Different service providers could provide different services in different areas. The service provider might disseminate information about cost, bandwidth and availability of the Internet access. Disseminating information from network may be implemented by a new protocol or as the extension of router advertisement. However, the security problems should be considered.
4. **User and Application Information**
   Users or applications might specify the service requirement. This information has to be delivered to the selection mechanism. The preferences could be presented as policies.

The interface selection decisions can be based on the above-mentioned information. The operation of interface selection must be continuous as the information might change at any point of time. Therefore, there have to be a policy database and a mechanism to hold and maintain policies for interface selection. Policies provide entities a possibility of controlling the placement of the mobile host's traffic flows onto different network interfaces.

5. Policy-based Routing for Multihomed Mobile IPv6 Networks

5.1. Introduction

The policy-based routing situation in a wireless network is distinct from that in a wired network. A wired node might be able to select the most appropriate interface because it can configure the best policy statically fit to constant wired network. But a wireless node can not use the static policies, as the wireless link is mutable and less reliable than wired networks. Additionally, user’s preference should be assigned as higher priority because wireless network always faces all kinds of unsecured environments and the resource is limited and costly.

NEMO Basic Support does not allow direct routing between **Correspondent Node**s (**CN**) and **Mobile Router**s (**MR**). Although solutions do exist for route optimization, they are too immature for standardization. Routing Optimization problems are left for later discussion. So, in NEMO Basic Support, a CN only knows one address of the **Mobile Network Node** (**MNN**) or MR, namely permanent home address. In some related research about policy-based routing on a mobile node, a CN could do policy-based routing as route optimization is carried out [MNISPBR]. But in our case, a CN could not realize the multiple routes to its destination, which makes it impossible to do policy-based routing.

Considering the situation when a multihomed mobile network is in a foreign link, several bidirectional tunnels might be set up between the MR and the **Home Agent** (**HA**) based on the NEMO Basic Support. The MR, the gate of the multihomed mobile network, takes the responsibility of forwarding each outgoing packet through a proper tunnel (interface); similarly, the HA takes the role of forwarding each incoming packet through a selected tunnel (interface). As these tunnels use different access technologies, they do not have the common characteristics, like transferring capability, stability; meanwhile, the network states (e.g. available bandwidth, congestion situation) of each of them might also change with time, location or traffic. In order to handle the traffic efficiently (like improve the whole network bandwidth, decrease the packet loss rate), policy-based routing, which focuses on managing both inbound and outbound packets, should be designed.

Policies could be created from diverse points of views. On one hand, users might worry about the cost and application programs might have special QoS requirements; on the other hand, tunnels have varying capabilities and real-time situations. The routing decision should be made on the basis of some or all of these constraints. There is a tradeoff between the decision correction and decision efficiency. If all the constraints are considered, the decision would be quite correct but not efficient for some reasons like the energy spent on making decisions is too high or the time is so long that the decisions are not useful any more.

There are several ways of handling received packets, which means we could have several types of policies. Packets could be dropped for some reasons like the special requirements of packets could not be satisfied by the MR at that moment; the user of the mobile network does not want to receive any packets originated from the hosts in his blacklist or somehow these packets are regarded to be illegal from the security perspective. The alternative is that packets are forwarded through some tunnel successfully, in which selecting a tunnel is involved.

In this section we are going to analyze how policy-based routing could be used to automatically control the traffic on a multihomed mobile network with the consideration of users' preferences.

Our discussion is based on the following assumptions:

1) Network mobility based on mobile IPv6, multihoming issues are solved in network layer.
2) No route optimization support: bi-directional tunnels are established between the MR and the HA, both of whom would take the load of routing packets.
3) We focus on multihoming configuration type 1 discussed in Section3.4. Mobile nodes inside the mobile network could communicate with the MR with IP-based technology so that they could specify the flow type with understandable (for all mobile IP nodes) values, how this is performed is out of the content of this thesis.
4) The connections between the MR and MNNs are stable; no error management mechanism is used.
5) This framework is only employed when the mobile network is in a foreign link.
6) We just ignore the discussion of network operator selection for each interface (if at the same time, there are more than one service providers providing the same type of connections), and suppose that the MR is able to choose one operator in some way.
7) The static information stored in the HA could be achieved from service provider periodically. The specific way of doing that would not be discussed here.
8) It is also assumed that each new flow type has its corresponding requirement specification sub-option in its first packet.
9) The security between the MR and the HA is assumed to be proved.

5.2. General Policy Functional Components in Mobile Networks

According to the policy management framework proposed by IETF, the characteristics of a mobile network and the assumptions described above, in general, the following functional components in a multihomed mobile network policy-based routing framework should be defined:

Component 1: Environment Detector

An **Environment Detector** is responsible for taking care of all kinds of useful information, like monitoring, updating and storing them. During the management period, the Environment Detector would detect all changes, from which some triggering events are created. Which information we should detect, why we use this information and how we could get it should be clearly reasoned. In the mobile network scenario, that information could be classified as:

Class One: Dynamic Information of interface situation

This type of information describes the current interface situations, which change with some factors like time, location and current traffic, etc.

Some possible information examples are:

Availability: It describes whether interfaces are usable or not at some given time point. This information is necessary since only the enabled interfaces could be considered during the interface selection period.

The fast and best way to know the availability is asking help from MAC layer by detecting the signal strength; otherwise, we have to use the heartbeat signals, which might be a huge cost. In our case, the MR could use layer 2 information; but because a HA is normally connected with a wired link, which can be considered available all the time. [FHMIP6] introduces the basic fast handover mechanisms for Mobile IPv6.

RTT[1]: Round Trip Time, this is used to measure the quality of tunnels, such as traffic congestion. Each given tunnel has its own default RTT, which is determined by its connection way and the distance between the two endpoints of a tunnel. In real world, RTT is influenced by traffic congestion significantly, which enables it to present the current link congestion situation. It is not wise to use a congested tunnel to forward packets.

The RTT between the HA and the MR could be calculated from the time difference between BU ACK and BU; or we could only calculate the latency between the MR and its Access Router from the time difference between Router Solicitation and Router Advertisement messages.

Packet Loss Rate: it is also used to measure the quality of a tunnel. Packet loss might be resulted from several reasons: traffic congestion, bad stability in wireless networks, etc. A tunnel with a high packet loss rate is unreliable.

Buffer Usage: each interface has a certain number of buffers, if the buffer of one interface is almost full, that interface should not be used since the traffic on that tunnel is already too heavy to be handled properly.

This information is maintained directly by the device, it could assist congestion detection.

[1] Round Trip Time is defined as the time it takes for a small packet to travel from client to server and then back to the client. It includes packet-propagation delays, packet-queuing delays in intermediate routers and switches, and packet-processing delays.

Class Two: Static Information of general interface capability

Different interfaces use different technologies to access the core network. Each technology has its advantages and disadvantages comparing with others. As a result, these characteristics would affect the routing decision for the sake of efficient transmission.

Some possible information includes:

Maximum Transmission Rate: each access technology has its specific transmission capability; therefore different types of packets with specific QoS requirements should be matched to different access technologies. This information is stored in the device initially.

Error Rates: as using diverse communication media, general error rate due to interference is varied. For instance, WLAN (Wireless Local Area Network) has a higher error data rate than fiber optics.

Safety and Security: safety situation might differ largely among varying network technologies, for example, using radio waves for data transmission might interfere with other high-tech equipments, and open radio interface makes eavesdropping much easier in WLAN than in the case of fiber optics.

Class Three: User Decision Information

As the user is the service target who enjoys and pays for the service, he/she, for some reason, may specify some policies which override the default policies. One good example is the user would like to reject all the traffic from some CN for some time.

This kind of user specified overriding information could be very little or extremely huge depending on complexity and functionality of the system.

Some possible information includes:

Refused host list: the IP address list of all refused hosts. It is meaningful when a user does not want to be disturbed by some correspondents.

Refused flow type list: this is meaningful when a user does not want to receive some kinds of flows, for example, he/she would like to refuse video call, perhaps, considering the energy or payment cost.

Class Four: External Information

Routing decisions are not only influenced by the interface situations but also some external information since we have to consider the user requirements.

Some possible information includes:

Service Cost: As a normal service user, he/she cares about cost and QoS. At the same time, different types of technologies are charged diversely. Even for the same technology, different service providers might charge differently. Furthermore, for the same service, one service provider might have special billing strategies for some cases, like the price in weekends is lower than the price in weekdays, the price in some districts is lower than in other locations, etc. And the uplink and downlink costs are different. Some related information might change with the movement of users. So we have to record this information to help making decisions.

This information is provided by all service/network providers. It might be changed sometimes. In our system, we suppose that all this type of information is stored at a HA, since the amount of information might be large, involving all the charging policies from all service providers, and the HA is much more stable to handle a lot of information. Updated information is sent to the HA by service provider automatically or alternatively the HA requests the related information and gets the answer from service providers. Each time when the MR sends a Binding Update message, the HA could get the location information from its CoA, so that it knows from whom it would ask for the cost information.

Component 2: Policy Decision Point (PDP)

As it was mentioned before, different kinds of policies could be implemented for the traffic management of a multihomed mobile network. For instance, we could define two types of policies, namely, **User Rejection Policy** and **Routing Policy**.

User Rejection Policy is used to decide whether discarding some packets or not. As nowadays the user has to pay for both incoming and outgoing traffic, he/she might refuse accepting some incoming traffic because of the cost or some other reasons like security. A user rejection decision event would trigger Interface Rejection Decision; the rejected packets could be identified by its source address and port number, or flow type decided by the events. The decision of a User Rejection Policy is the updated Rejection List where all to-be rejected packets information is maintained.

Routing Policy is used to decide through which interface the received packets should be sent out. It could be triggered by all kinds of related events from PEPs, for example, an interface is enabled or disabled. The output of Routing Policy evaluation is the updated Interface Lists for all kinds of cases.

Component 3: Policy Enforcement Point (PEP)

Decisions made by a PDP are enforced on a PEP. PEPs and PDPs could be located at the same device or distributed.

Component 4: Policy Repository

All related policies for managing the traffic of a mobile network can be stored in a Policy Repository. A PDP might have to download policies from Policy Repository. Once a policy is downloaded to a PDP, the PDP would use this policy until it is asked to update the Policy. The repository may also reside separately from the PDP as for one thing, a centralized Policy Repository could make multiple devices be loaded with consistent policies, for another, the simplicity of device can be achieved in the case of a centralized Policy Repository.

5.3. Placement of Functional Components

After analyzing which policy-based functional components are involved in a mobile network environment, we are in front of another question: how to place these functional components?

Since we are dealing with traffic management policies on multiple interfaces when a mobile network is in a foreign link, the MR and the HA are two 'must' devices.

Before we go deep into the discussion, first let us recall the architecture of a multihomed mobile network in a foreign link. As mentioned in **Figure 9**, a bi-directional tunnel is set up between the MR and the HA. Furthermore, the relation among different policy functional components is described in **Figure 22**. Our following discussion is based on the multihomed mobile network architecture and the policy components relation.

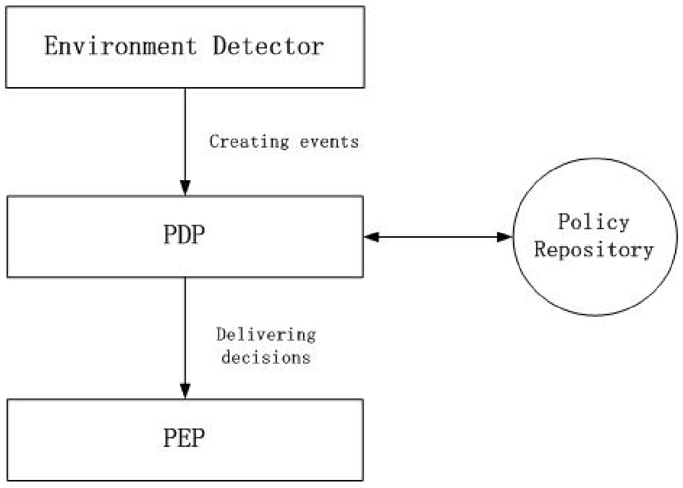


Figure 22: Functional Components Relation Chart

5.3.1. Environment Detector

All the information that might initiate an event should be managed by Environment Detector. Environment Detector should locate at the place where it could sense the information which is described in Section 5.2. So both the MR and the HA should have this functional component.

5.3.2. PDP

Now we are going to talk about the PDP, some general problems about the location of the PDP are discussed here.

When we think about the location of a PDP, we should pay some attention to several factors that determine the correctness of our design. These factors are:

First, ***the location of the triggering event***: a PDP can be triggered by all kinds of events; it is intuitive to have the idea that the closer the event creator (Environment Detector) located to the PDP, the better the performance is. The good performance might be, for instance, faster, safer or with less traffic. So the best way is to put them in the same device.

Second, ***the location of the PEP*****:** as the decision results of a PDP are going to be sent to PEPs, it is also desirable to locate the PEPs near to the PDP for the same reason of locating triggering event.

Third, ***the location of the assistant information:*** as the external information is used by the PDP, if this information is located too far from the PDP, the decision making time might be too long to make sense. Besides that, you also have to take the risk of safety problems, since the communication might get tampered with by an eavesdropper.

Fourth, ***Decision making time***: as described before, it is affected by the location of the external information. At the same time, the device computing ability, energy consuming rate and storage ability, and decision complexity would also influence the decision making time.

Due to the requirements of the system, those factors listed above might be extremely critical or somehow non-significant. However, one thing is clear: we have to consider the whole procedure performance from the event creation to the policy enforcement.

It should be noticed that one policy might overwhelm other policies; and policies made by different PDPs might meet conflictions.

Besides these factors, some other factors specific to the multihomed mobile networks should also be considered carefully.

Although we always say that the tunnel between the MR and the HA is bi-directional, when we analyze the Routing Policy for a mobile network, we had better treat this tunnel as two uni-directional tunnels. Cause in reality the situations of downlink and uplink on one bi-directional tunnel might be quite different. They work independently in most techniques. At the same time, decisions enforced by the MR are about which interface the MR would use to forward a received packet (only uplink traffic); and decisions enforced by the HA are about to which interface the HA would forward the received packet (only downlink traffic).

Policies used for traffic on two directions might be different, for instance, they may all take cost into account, but the MR may also consider its buffer situation on each interface for the outbound traffic, and the HA may not consider that because all tunnels might end at the same interface on the HA. Furthermore, even if they use the same policy, for example, if one interface is turned on/off, the cheapest tunnel is chosen, their decisions might be different since uplink and downlink costs are calculated individually. It is possible that you get the cheapest uplink service on tunnel 1 and the cheapest downlink service on tunnel 2. So in this framework, two PDP subtypes, namely **Inbound Traffic PDP** and **Outbound Traffic PDP**, should be distinguished. Inbound Traffic PDP takes care of all policy decisions on the incoming traffic to the mobile network, and Outbound Traffic PDP takes care of all policy decisions on the outgoing traffic from the mobile network.

In our case, the traffic between the MR and the HA is mainly determined by three factors: event transfer, information request and reply, and decision transfer. Time cost is determined by the above traffic in addition to computing time. The time spent for communication among components on one device could be ignored comparing the communication time between different devices. This procedure has been showed in **Figure 19**, and would be elaborated more in the Section5.4.

5.3.3. PEP

Because all the traffic from and to a mobile network have to pass the tunnels between the MR and the HA, routing policies have to be executed on both inbound and outbound traffic. Once realizing the MR is the manager of the outbound traffic and the HA is the manager of the inbound traffic, it would not be hard to come to the conclusion that both endpoints of the tunnels should take the role of a PEP.

5.3.4. Policy Repository

It is reasonable to place a Policy Repository in a fixed, stable location since the PDP might retrieve policies any time. So the Policy Repository should stay in the home link.

5.4. A Policy Example Study

To make the PDP location analysis as clear as possible, we pick a **policy example**, which covers all the possible message exchange cases.

5.4.1. Policy events and actions

Table 1: Event Type and Decision Classification[1]

Event	***Events created by***		***Decisions made by***	
	MR	HA	OutboundTrafficPDP	InboundTrafficPDP
New interface	√		√	√
Transmission Quality change	√		√	
UserRejection	√			√
UplinkCost Change		√	√	
DownlinkCostChange		√		√

This ***policy*** consists of 4 kinds of events:

One is the appearance/ disappearance of an available interface (Dynamic Information of interface situation) which is going to be handled by both the Inbound Traffic PDP and the Outbound Traffic PDP. In this case, decision making might need external information, for instance, a new service provider provides some services on one interface, but at that moment, the MR knows nothing about the new service.

Another one is the transmission quality change (Dynamic Information of interface situation), which is only going to be deployed by the Outbound Traffic PDP.

1 The Availability and Transmission Quality Changes on one interface decide whether this interface is able to join in the interface selection or not. The Uplink Cost Change and Downlink Cost Change alter the interface selection directly.

Transmission quality might be checked through different parameters for different access technologies.

The third one is the change of cost (External Information), which would trigger both the Inbound Traffic PDP and the Outbound Traffic PDP.

The last one is user rejection event (User Decision Information), which would only trigger the Inbound Traffic PDP.

The relationship among events, events creator, and decision acceptors could be described in **Table 1**.

Related Policy Actions are:

1, Routing Policy for outbound traffic:

If one interface is turned on, reselect the cheapest one.

If one interface is turned off and it is the selected one, reselect the cheapest one.

If the Transmission Quality situation of one of the marked interfaces goes down to its low threshold, mark this interface, if it is the selected one, reselect another interface from those unmarked interfaces.

If the Transmission Quality situation of one of the marked interface goes up to its high threshold, unmark this interface, and reselect the cheapest interface.

If the outbound traffic cost on one interface is changed, select the cheapest one.

2, Routing Policy for inbound traffic:

If one interface is turned on, reselect the cheapest one.

If one interface is turned off and it is the selected one, reselect the cheapest one.

If the inbound traffic cost on one interface is changed, select the cheapest one.

3, User Rejection Policy for inbound traffic:

If user sets rejection on some CN, add IP address of the CN into the rejected host list to filter all traffic from this CN.

So simple and independent are the above designed policies, that the occurrence of policy conflicts is disabled.

5.4.2. Analysis of Four PDP Location Possibilities

Now let us have a look at four PDP location plans from the perspectives of time cost, traffic overhead for a network, and battery etc. Each plan is described through two cases (scenarios), which are identified by the type of events. The first scenario covers the events created by Environment Detector in the MR, such as appearance of a new available interface; buffer usage overload and user rejection. The second considers events created by Environment Detector in the HA, such as cost change.

Here we make an assumption that the MR would not download new policies from Policy Repository, so in all described framework architectures (**Figure 23**, **Figure 26**, **Figure 29**, **Figure 30**), the communications between the MR and the Policy Repository are depicted in dashed lines. Another assumption is that all external information from the service providers could be downloaded to the HA in some way.

5.4.2.1. Plan1: Both the Inbound Traffic and the Outbound Traffic PDP are located on the MR

The architecture could be depicted in **Figure 23**.

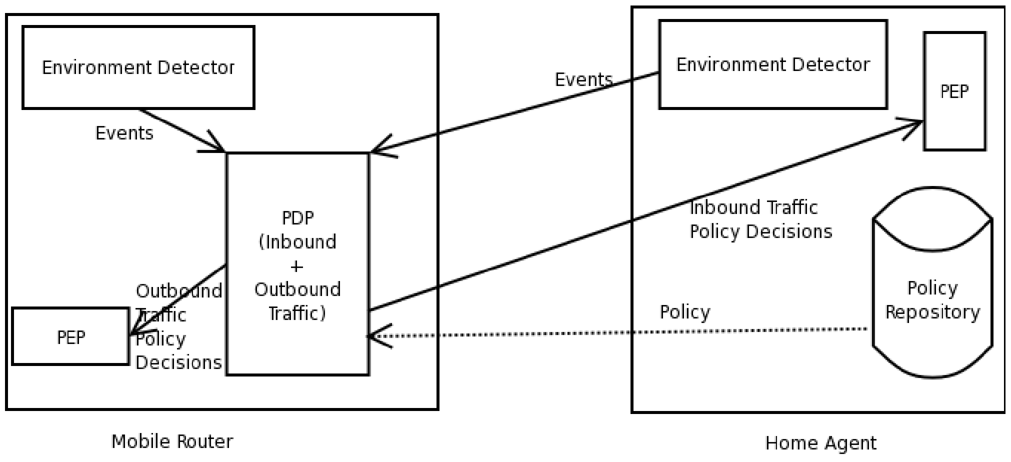


Figure 23: Policy-based Framework Architecture Plan1

The analysis of required interactions between* the *MR and* the *HA

***case1**: PDP is triggered by the Environment Detector in* the *MR*

The whole procedure (from event creation to decision enforcement) time sequence chart of this case is shown in **Figure 24**. In this case, for a new interface appearance event, three communication messages are created within one decision making procedure.

Information request is the first message. When the HA receives the message, it retrieves information from its Environment Detector. This retrieved information forms the second message. The MR then gets the requested information, and makes its Inbound and Outbound Traffic Decisions. At last, the Inbound Traffic Decision is sent to the HA through another message. So two messages are about information exchanges, one is for the decision transmission.

In some cases, the MR might cache some old information for future use; the first two messages could be saved.

For a transmission quality event, no communication is made between the MR and the

HA. The reason is this event would only trigger the Outbound Traffic PDP which can make its decision and inform decisions to its own PEP without disturbing the HA.
For user rejection event, one communication message is used.

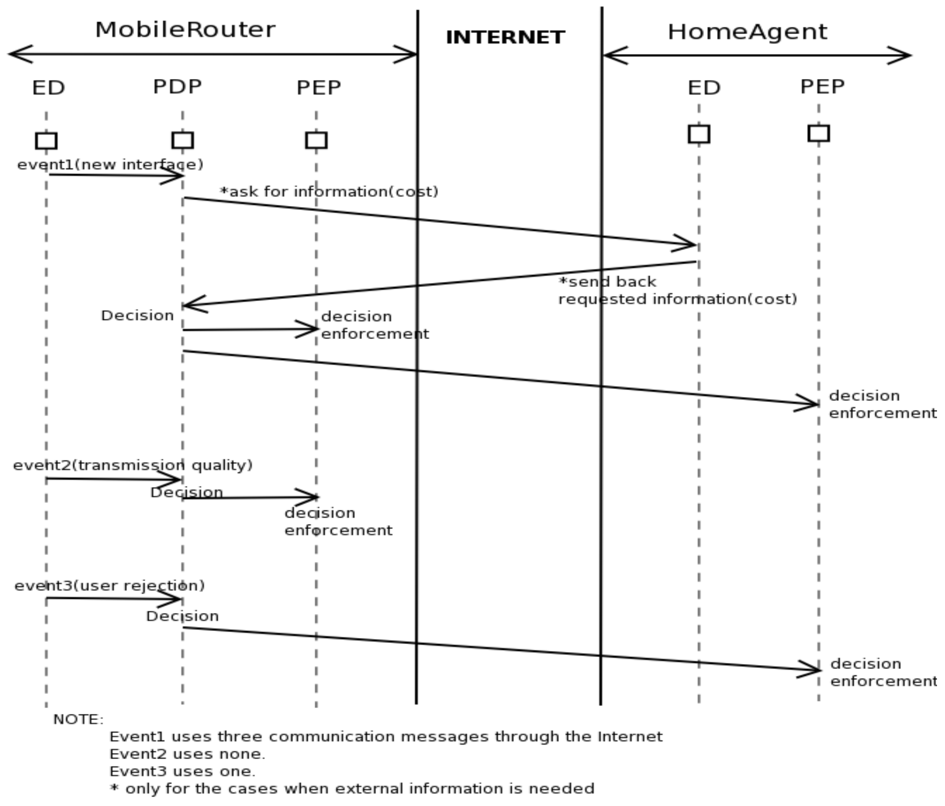


Figure 24: Time Sequence Chart, Plan1 Case 1

***case2**: PDP is triggered by the Environment Detector in the HA*

The whole procedure (from event creation to decision enforcement) time sequence chart of this case is shown in **Figure 25**.

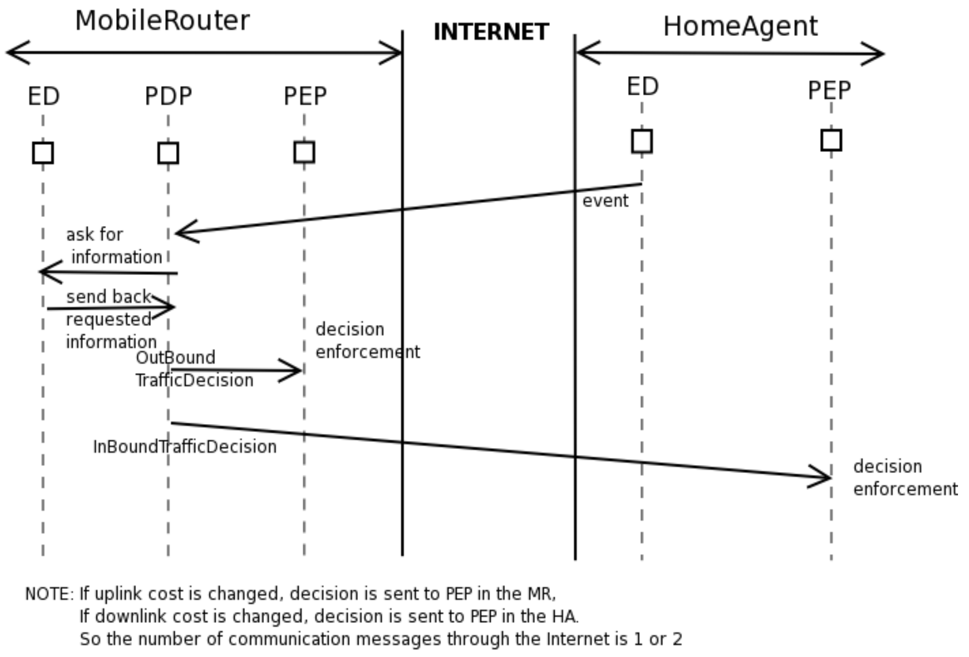

MobileRouter
INTERNET
HomeAgent
ED
PDP
PEP
ED
PEP
event
ask for information
send back requested information
OutBound TrafficDecision
decision enforcement
InBoundTrafficDecision
decision enforcement
NOTE: If uplink cost is changed, decision is sent to PEP in the MR,
If downlink cost is changed, decision is sent to PEP in the HA.
So the number of communication messages through the Internet is 1 or 2

Figure 25: Time Sequence Chart, Plan1 Case 2

In this case, **one** or **two** messages (cost on one or two directions is changed) have to be used during one decision making procedure. Events are created by the HA, in order to use as less messages as possible; the event creator should be smart enough to add all necessary information to the message so that the PDP in the MR do not need to ask for information from the HA any more. This event triggers one or both of the Inbound and the Outbound Traffic PDP. If the Inbound Traffic PDP is triggered, the MR makes the Inbound Traffic Decision and sends it back through another message.

5.4.2.2. Plan2: Both the Inbound Traffic and the Outbound Traffic PDP are located on the HA

The architecture could be depicted as **Figure 26**.

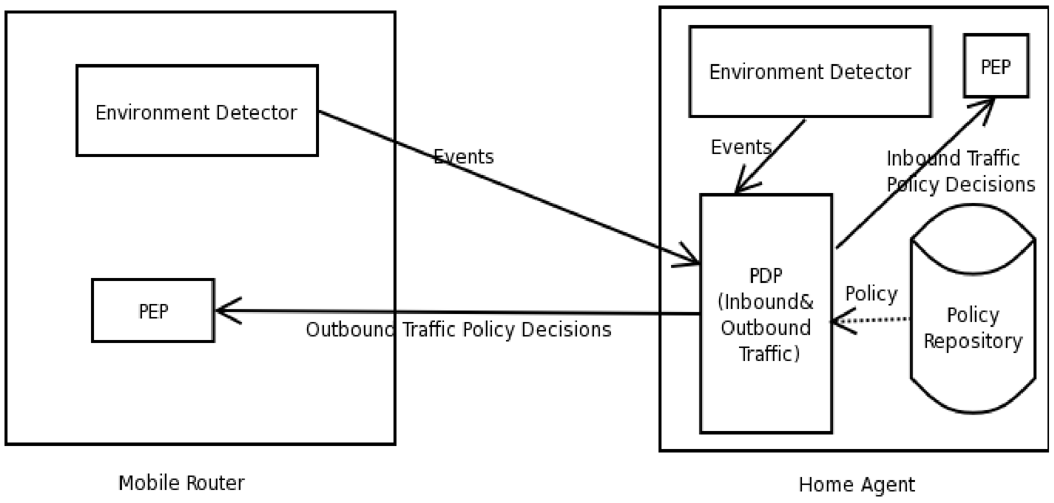


Figure 26: Policy-based Framework Architecture Plan2

The analysis of required interactions between the MR and the HA

case1*: PDP is triggered by the Environment Detector in the MR*

The whole procedure (from event creation to decision enforcement) time sequence chart of this case is shown in **Figure 27**. In this case, for the new interface appearance event, **two** messages have to be used in one decision making procedure. Event is created and sent out through one message. This event triggers both the Inbound and the Outbound Traffic PDP. Then the HA makes its decisions and sends back the Outbound Traffic decision through the second message.

For a transmission quality event, **two** messages are needed. We notice that because the Outbound Traffic PDP is located on the HA, the MR has to communicate with the HA, though the decision is still enforced by the MR. For user rejection event, one communication message is used.

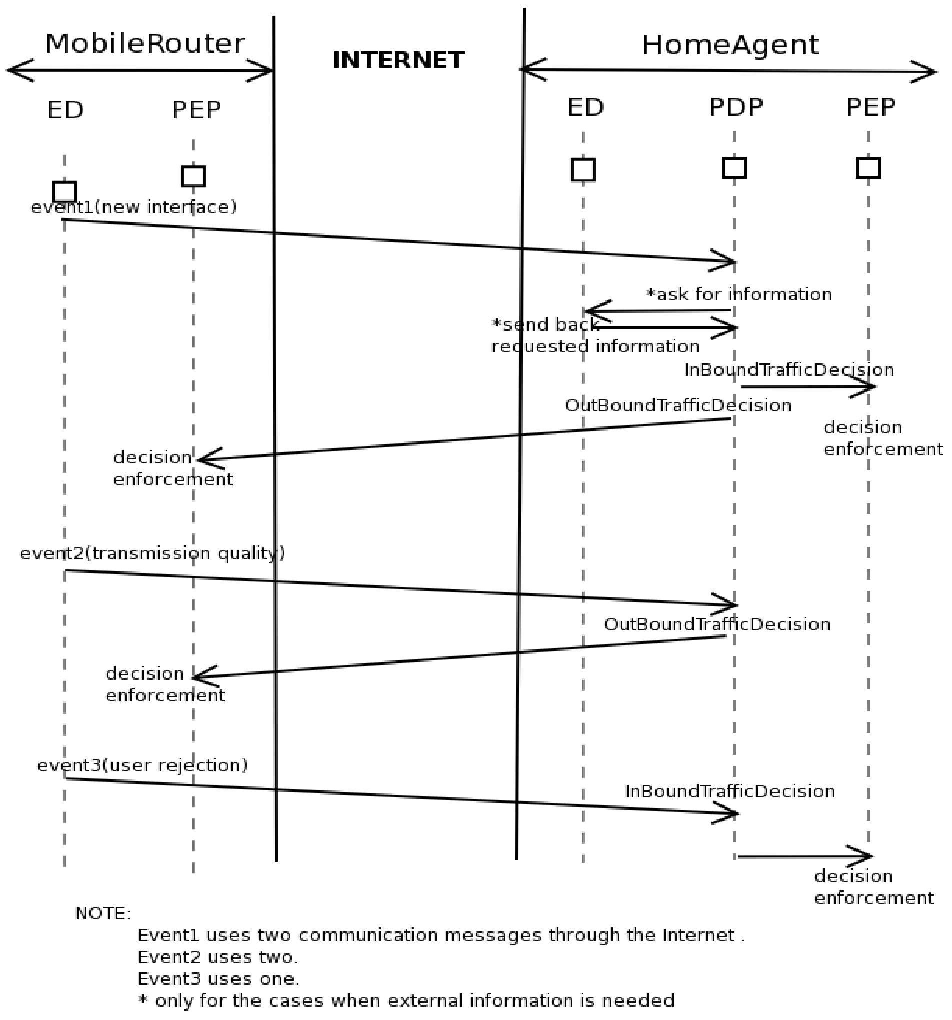


Figure 27: Time Sequence Chart, Plan2 Case 1

case2*: PDP is triggered by the Environment Detector in the HA*

The whole procedure (from event creation to decision enforcement) time sequence chart of this case is shown in **Figure 28**.

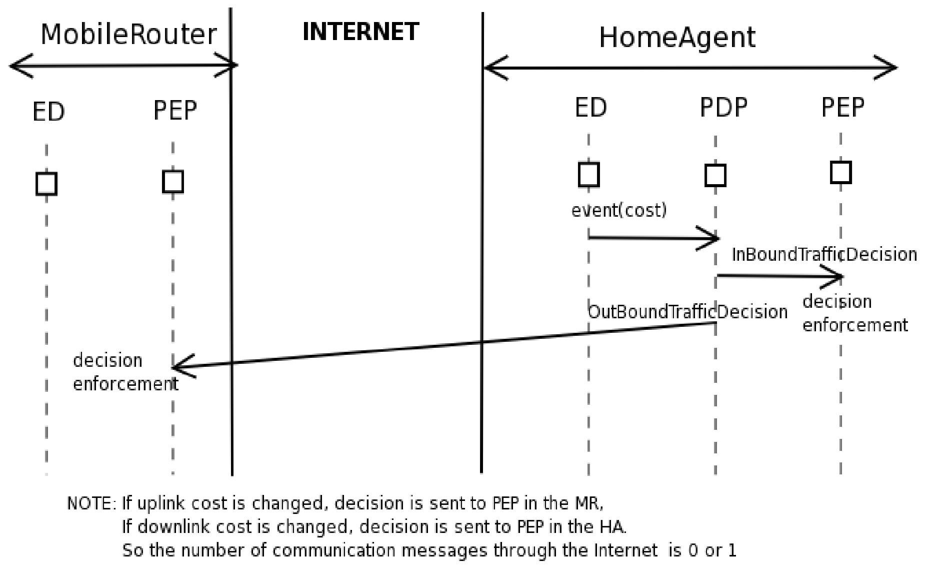

MobileRouter
INTERNET
HomeAgent
ED
PEP
ED
PDP
PEP
event(cost)
InBoundTrafficDecision
decision enforcement
OutBoundTrafficDecision
decision enforcement
NOTE: If uplink cost is changed, decision is sent to PEP in the MR,
If downlink cost is changed, decision is sent to PEP in the HA.
So the number of communication messages through the Internet is 0 or 1

Figure 28: Time Sequence Chart, Plan2 Case 2

In this case, **none** or **one** message is required in one decision making procedure. If merely the downlink cost is changed, no Outbound Traffic Decision is made or sent to the PEP in the MR, so no communication traffic is made between the MR and the HA. If the uplink cost is also changed, the decision has to be sent to the MR through one message.

5.4.2.3. Plan3: PDP Located close to Environment Detector

Now we would like to decrease the traffic between the event creator and the PDP to improve the performance of this framework. Both the MR and the HA run a PDP, and events created by the MR would be handled merely by the PDP in the MR; events created by the HA would be handled only by the PDP in the HA. The decisions made by each of them still have to be enforced on both of them. However, one big weak point of this plan is the decision conflicts.

The MR and the HA might be triggered respectively at the same time, which means both of them would make their own decisions based on the received events. As the events are different, decisions made on one type of traffic might be different. Thus conflicts occur. So if we would like to deploy this plan, we have to appoint a master PDP who is going to handle the conflicts. Meanwhile, the mechanisms for finding the conflicts have to be designed as well.

It is clear that dealing with conflicts is time and energy consuming; here we just give out some potential solutions of conflict control. One way to find conflicts is to number all the decisions made by two PDPs. When a new decision is made or received, the expected decision number is increased by one, so that the next made decision would be numbered as the expected decision number, or the next received decision would be checked whether its id is equal to the expected one. One conflict is found when the received decision number value is different from the expected one.

Once a conflict is found, the conflict control mechanism is invoked. It could be similar to TCP retransmission mechanism, which would remake all the conflicted decisions. We could imagine that the decision redo procedure would take quite much time. But if conflicts take place seldom, it might decrease the traffic amount in a long run. Also, some similar conflict resolution mechanisms might be required in the complicated cases like multiple MRs and multiple HAs.

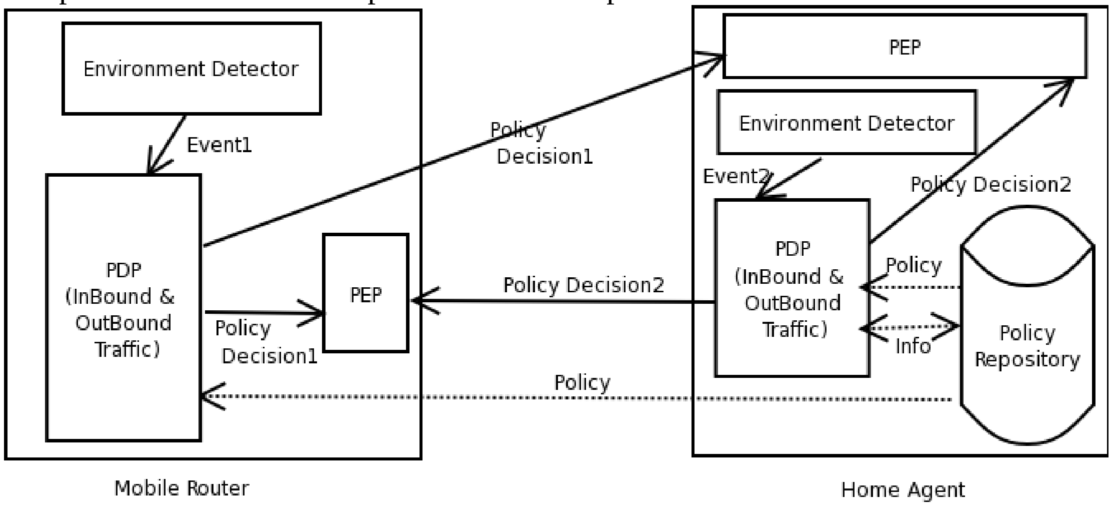


Figure 29: Policy-based Framework Architecture Plan3

The architecture is shown in **Figure 29**. We are not going to analyse the traffic in this case, since there is no doubt that it makes heavier burden than other choices.

5.4.2.4. Plan4: PDP Located close to PEP

Through the analysis of Plan1 and Plan 2, we find that in some cases (e.g. buffer usage event), event creation, decision making and decision enforcement occur on the same device, thus in those cases traffic between different devices is a waste. In order to avoid this type of traffic burden, we come to plan4, in which the OutBound Traffic Decision is made and deployed on the MR and the InBound Traffic Decision is made and deployed on the HA. There would not be any conflicts between two PDPs since they make decisions on different traffics. However, we still notice that the events from both the MR and the HA could trigger the two PDPs. The architecture is presented in **Figure 30**.

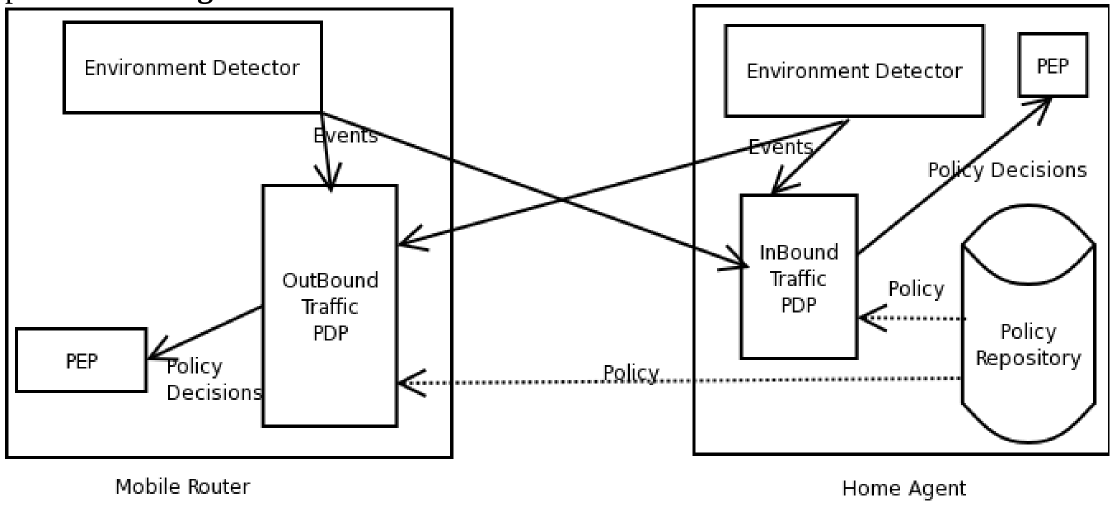
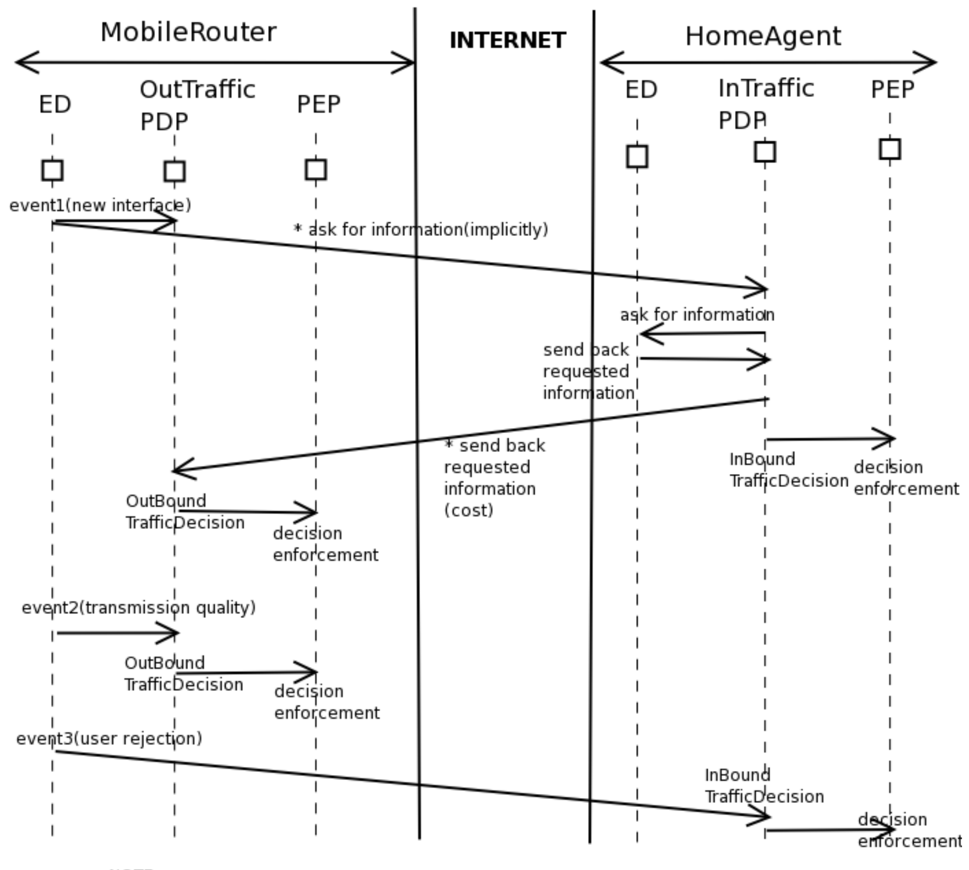


Figure 30: Policy-based Framework Architecture Plan4

The analysis of required interaction between the MR and the HA

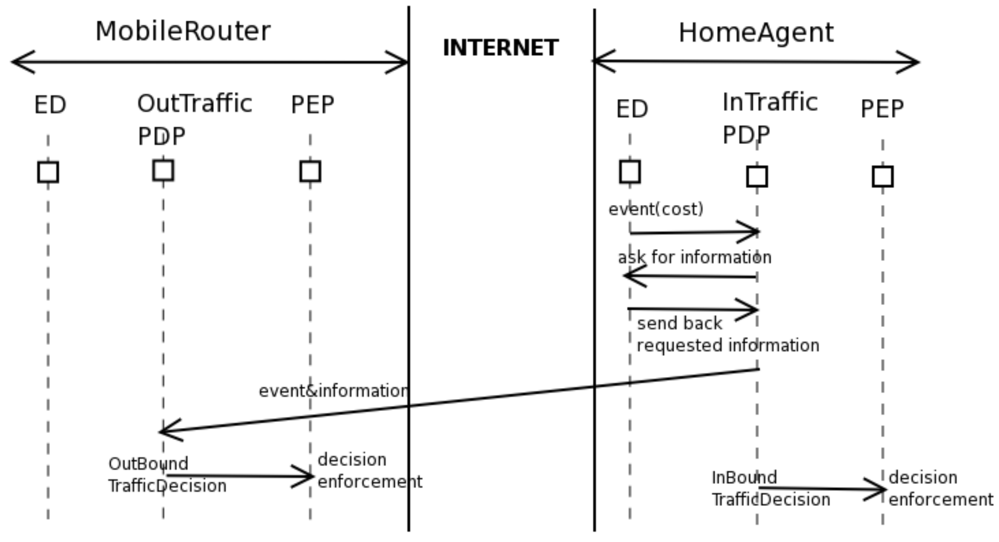
case1*: PDPs are triggered by the Environment Detector in the MR*
The whole procedure (from event creation to decision enforcement) time sequence chart of this case is shown in **Figure 31**. In this case, for new interface appearance event, two PDPs are triggered. If the HA is intelligent enough to send back some information through the second message to the MR according to the event received from the MR, the OutBoundTraffic PDP does not need to create an extra message to ask for information any more. Information request from the OutBoundTraffic PDP is implicitly included in the event which triggers the InBoundTraffic PDP in the HA.
For a transmission quality event, no traffic is caused between the MR and the HA. For a user rejection event, one communication message is used.

case2*: PDPs are triggered by the Environment Detector in the HA*
The whole procedure (from event creation to decision enforcement) time sequence chart of this case is shown in **Figure 32**.



NOTE:
Event1 uses two communication messages through the Internet .
Event2 uses none.
Event3 uses one.
* only for the cases when external information is needed

Figure 31: Time Sequence Chart, Plan4 Case 1



NOTE: If uplink cost is changed, decision is sent to PEP in the MR,
If downlink cost is changed, decision is sent to PEP in the HA.
So the number of communication messages through the Internet is 0 or 1

Figure 32: Time Sequence Chart, Plan4 Case 2

In this case, none or one message is used, which is also similar to the case2 in Plan2. Another benefit of plan4 is that it avoids the required confirmation mechanism between the PDP and the PEP in the case of plan1 and plan2.

The comparison among plan1, 2 and 4 on the aspects of Number of required messages through the Internet, Processing requirement of the MR etc., is summarized in **Table 2**.

Table 2: Conclusion of 4 plans

		No. of messages	***Processing Requirement of the MR***	***Computing Entity***
Plan1	case1	4	Full	MR
	case2	1 or 2		
Plan2	case1	5	None	HA
	case2	0 or 1		
Plan4	case1	3	Shared	MR & HA
	case2	0 or 1		

So, we could notice that **Plan4** is the best choice, and we are going to implement Plan4 in the next section.

5.5. Comparison between a Mobile Node and a Mobile Network

In our example, only one MR is used in the mobile network, which looks similar to the mobile node case. We would like to emphasize the difference between them.

A mobile node has multiple interfaces; it always has to select a suitable interface for the outbound traffic. As a mobile node can be identified by multiple CoAs(each interface has its own IP address), if route optimization is not used, all the traffic would go to the HA and the HA forwards the datagram according to the binding cache(in the binding cache, one HoA is mapped with one CoA, so the HA does not need to select one interface. There is only one matched entry); if route optimization is used, the CN has to choose one CoA as the destination address of datagrams. **Figure 33** reveals this procedure.

In mobile network configuration case (1,1,1) (see Section 3.4), instead of registering HoAs of all mobile nodes, only the network prefix is registered to the HA with multiple interfaces CoA of the MR. The outbound traffic control is the same as mobile mode case, but the inbound traffic control is different. Since no route optimization takes place, all inbound traffic has to pass the HA. The datagrams could be matched with more than one entry with the same MNP, and the HA has to take the responsibility of choosing one tunnel for each datagram. The policy-based routing is for selecting tunnels which are set up by the MR and the HA. These two actors are enough to take care of policy-based routing on all the traffic. **Figure 34** describes this case.

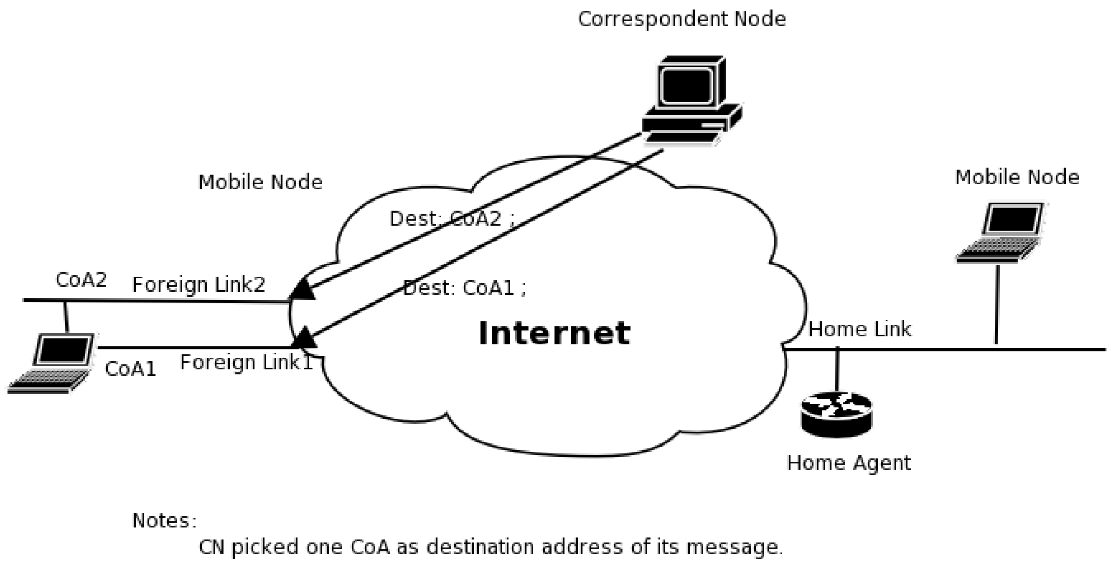


Figure 33: A Multihomed Mobile Node

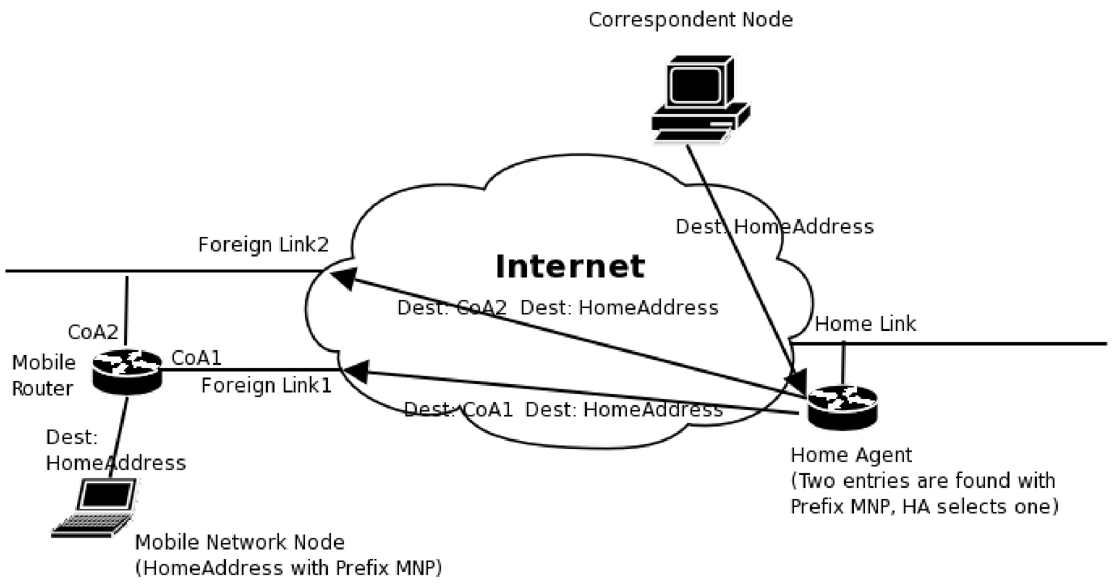

Correspondent Node
Dest: HomeAddress
Internet
Foreign Link2
Dest: CoA2 Dest: HomeAddress
Home Link
CoA2
Mobile Router
CoA1
Foreign Link1
Dest: CoA1 Dest: HomeAddress
Dest: HomeAddress
Home Agent (Two entries are found with Prefix MNP, HA selects one)
Mobile Network Node (HomeAddress with Prefix MNP)

Figure 34: A Multihomed Mobile Network

6. A Concrete Policy Implementation

In this section, we work on a concrete policy, which takes into account the channel situation, user preference, and cost etc. With regard to the data relevant to the policy-based routing framework in **Figure 30**, not only the data maintained by all entities, but also the data transferred between them in manner of a protocol, are described.

6.1. The Concrete Policy

Table 3: A Policy Set

Information Types	***Actions***
A new interface becomes active; its signal strength goes up to some threshold.	Reselect the cheapest interface
The selected interface becomes inactive; its signal strength drops below some threshold.	Reselect the cheapest interface
Fatal failure on some interface	Reselect the cheapest interface if necessary
Congestion occurs on one interface, its MAC Delay turns longer than some threshold. ***1**	Mark this interface (and maybe reselect the cheapest one from unmarked interfaces if necessary)
Congestion is solved on one interface; its MAC Delay turns shorter than some threshold. ***1**	Unmark this interface and reselect the cheapest one
Cost on some tunnel changes	Reselect the cheapest one
User actively decides to add/remove some CNs from blacklist stored on the Home Agent. ***2**	Refuse packets arriving at the HA from its blacklisted nodes
A MNN becomes inaccessible, since the MNN is turned off or out of power or leaves its current network. ***3**	Inform the HA to refuse packets to this MNN generated from any CNs
Energy of a Mobile Router is Low	Stop the Binding Update on other interfaces; Refuse new applications(new CNs) on both directions
QoS requirement unsuitable for selected interface ***4**	Reselect interface for this special flow and reserve the requested resources

***1:** Transmission Quality could be signal strength, congestion and DataLossRate;

***2:** User Rejection information could be the IP address of peer node; flow type;
***3:** A Node Discovery Program could help to find which nodes are connected or disconnected;
***4:** Flow with QoS requirements should be handled differently (have high priority on using any interface) from other flows.

In our Policy framework, messages are used for transferring the information which would trigger or help PDPs (Policy Decision Point). All types of information are listed in Table 3, and some of them will be implemented in our policy protocol. Our future discussion is how to present this information in a proper way.

6.2. Information Maintenance for a Policy Framework

As two entities of the policy-based routing framework, the MR and the HA, each maintains some data.

6.2.1. Mobile Router

A Mobile Router has to maintain an **Interface State Table** described in **Figure 35**, which extends the Binding Update List[1] used by basic NEMO. All the information that influences the policy decision is recorded there.

Interface ID	***CoA***	***MNP***	***Marked***	***Cost***
: :	: :	: :	: :	: :

Figure 35: Interface State Table

Interface ID and *CoA* are used to identify an entry; *Marked* describes the connection situation, when it is set to one, it means the situation is not good, if there is better choice (other connected and unmarked interface), the marked interface is not recommended any more.

Interface State Table is updated when necessary. The *Selected interface (the result of actions taken by a PDP)* is determined by changes of items in the table, for instance, a new entry is added (denotes the availability of a new interface, only available interfaces are stored in this table), an old entry is deleted (denotes the unavailability of an old interface), *Cost* of one interface is updated (denotes receiving new cost information), *Marked* of one entry is updated (denotes quality of this interface alters), etc.

6.2.2. Home Agent

We use the similar **Tunnel State Table,** shown in **Figure 36**, for the HA.

In this Tunnel State Table, Cost presents the downlink cost of a tunnel. There is no *Marked* item. For implementation, PDP would be triggered by the changes of this table as well.

[1] For more information about the Binding Update List, please see Section 2.3.2.

Interface ID	*CoA*	*MNP*	*Cost*
:	:	:	:
:	:	:	:

Figure 36: Tunnel State Table

Meanwhile, a Home Agent manages two rejected node lists: the rejected MNNs and the rejected CNs. Packets can be delivered by the selected tunnel, only after they pass the rejection test, which compare the relative IP address with the IP addresses contained in these lists. Their formats are introduced in **Figure 37** and **Figure 38**.

Inaccessible MNNs (IP address)
:
:

Figure 37: Rejected MNN List

Rejected CNs(IP address)
:
:

Figure 38: Rejected CN List

6.3. Policy Information Transfer Protocol

As Section 5 explained, policy information would be exchanged between the MR and the HA. Thereby, a protocol for carrying policy information is required. In this section, we list and study two implementation possibilities: an application layer protocol or an IP extension header.

6.3.1. An Application Layer protocol

The idea is derived from the consideration of the sources of all transmitted information. Since on one hand, we need some policy information which is not provided directly by IP layer, and on the other hand we would like to make sure the delivery of this information is performed correctly, it is intuitive to achieve the idea of making a new application layer protocol with the guarantee from the transport layer protocol TCP. The benefit is leaving the message control task to TCP.
Since policy messages are created in the application layer, a TCP connection is identified by (HoA of MR, SourcePort, DestAddress, DestPort), and the HoA of a MR is a permanent address. The IP packet created for this TCP message, like all other IP packets received from MNNs, would be encapsulated into another IP Header, which takes CoA of the selected interface as its Source Address. The whole procedure

is actually an application program running on a MR using TCP. Because MR itself also runs Mobile IP, TCP connection would not break with the movement.
One information package can include multiple pieces of information. One piece of information can be uniquely identified by *#Sequence* of the information package and *InformationID* of the information. Information about other interfaces could be inserted into the same IP package if only we can distinguish them.

Policy Protocol Header Format
The application level header has its format described in **Figure 39**.

Version	Reserved	PolicyType	MessageLength
PolicyMessageData			

Figure 39: Policy Protocol Header Format

Version (8bits), describes the Version No. of this Policy Information Transfer Protocol;
Reserved (8bits), for future extension;
PolicyType (8bits), can be 0 or 1. Type 0 implies a **Request** message; Type 1 means a **Response** message;
MessageLength (8bits), describes the length of *a PolicyMessageData* field;
PolicyMessageData (a variable length): stores all kinds of information.

PolicyMessageData field Format

PolicyMessageData field has one or multiple sub options shown in **Figure 40**.

InformationID	InformationType	InformationLength	Reserved
InterfaceID (CoA)		Information	

Figure 40: PolicyMessageData Format

InformationID (8bits), is the identity of the policy information. Multiple policy information can be sent together through one policy package;
InformationType (8bits), defines which kind of information it is; For each kind of PolicyType, there are more than one kinds of information types;
InformationLength (8bits), defines the length of *Information* field;
Reserved (8bits): some information type might be subdivided by some reserved bits.
InterfaceID (8bits), as the identifier of an interface, it shows which interface this information is about. Here CoA could also be used, each interface has one CoA;
Information (a variable length): the value/content of this information, the length is specified in the field *InformationLength*.

Request Type
Several InformationType in Request Type is defined, and their related PolicyMessageData field is described here:

InformationType=1: This message describes that the policy requestor needs information about the given interface. The reason might be an interface is on or starts using service from another ISP. The detailed format of this information type is shown in **Figure 41**.
InterfaceID is used to identify the requested interface. Information field could be

empty, as we simplify the problem by only replying the cost information. In the future, this field can be extended to specify which kinds of other information it asks for.
When "Response" part received this information, it would reply a message with subtype 1 or 3. If the response type is 1, it would also trigger the PDP.

InformationID	InformationType=1	InformationLength	Reserved
InterfaceID (CoA)		Information(0)	

Figure 41: PolicyMessageData Format of Information Type1

InformationType=2: This type of message presents the case of one interface going down. Its format is the same as type 1.

InformationType=3: The variation of uplink cost would be presented by this message type. Here, new cost is stored in the Information field.
The response type might be 2 or 5.

InformationType=4: This message introduces the rejected/released CN list. **Figure 42** presents the information format. The type is specified by a one bit flag T. When it is set to 0, it is the rejected CN list, which should be added into the rejected list; when it is 1, it is the released CN list, which should be deleted from the rejected list. RequestData field stores the IP list.

InformationID	InformationType=2	InformationLength	T	Reserved
InterfaceID (CoA) *		CNlist		

Figure 42: PolicyMessageData Format of Information Type4

When a policy replier receives this information, it would send back a response message with subtype 2 or 5. If the subtype is 2, the policy replier would also change its rejected/released CN list.
*** :** if the traffic from the CN list is rejected by all interfaces of MR, InterfaceID is set to 0; otherwise, it is set to the ID of rejecting Interface.

InformationType=5:
Similar to type 4, type 5 message presents rejected/released MNN list. **Figure 43** describes this information format. When flag T is set to 0, it is the rejected MNN list, which should be added into the rejected list; when it is 1, it is the released MNN list, which should be deleted from the rejected list.

InformationID	InformationType=3	InformationLength	T	Reserved
InterfaceID (CoA) *		MNNlist		

Figure 43: PolicyMessageData Format of Information Type5

The response is same as type 2.
*: InterfaceID field could be set to 0 to present that all traffic to the MNN list can not pass MR, or it is set to the ID of the interface through which traffic to the MNN list could not pass.

Response Type
Recall that our goal is to make sure the transfer of information which might create a policy evaluation event, the reply to these information messages could be

one success flag or some requested information, depending on the information type. Format of ***PolicyData*** field in ***Response*** **message** is shown in **Figure 44.**

ResponseID	ResponseType	ResponseLength	Reserved
ResponseData			

Figure 44: PolicyData Format in Response Message

ResponseID: the value is the same as the one received.
ResponseType: since there are two kinds of request: one is for asking for information and the response is the information; the other one is for making decision and the response should be whether the decision is made or not.

According to real situation, several possible ResponseType might be:
1 = requested information is found and sent back
2 = PDP accepted the information and made policy decision correctly
3 = both 1 and 2 are performed
4= requested information is unavailable
5= received information is not enough to make decision
For type 1 and 3, ResponseData field has the format described in **Figure 45**.

InfoType 1	InfoLength	Info 1
InfoType n	InfoLength	Info n

Figure 45: ResponseData Format

ResponseData field is with variable length.
For the other types, ResponseData field is empty.

6.3.2. An IP Extension Header

Another choice is to put policy information into an IP extension header, like the Mobility Header extended by NEMO. Alike means could be adopted for controlling and retransmission management.

The difference between BU (Binding Update) class messages and Policy messages is, in the first case, a MR sends BUs and receives BAs, and a HA receives BUs and replies BAs; in the second case, each of them might be a Policy message requester or replier[1]. But for each requester and replier pair, no matter who is taking on which part, same mechanism for retransmission can be used.

Let us first have a look at how a MR and a HA handle BUs and BAs. A MR records an *initial retransmission timer*, if the MR fails to receive a valid matching response within the selected initial retransmission interval, it should retransmit the message until a response is received. The retransmissions by the MR must use an exponential back off process in which the timeout period is doubled upon each retransmission, until either it receives a response or the timeout period reaches the value of a parameter MAX_BINDACK_TIMEOUT. The MR may continue to send these messages at this slower rate indefinitely. It is also required that the MR should start a separate back off process for different message type, different home addresses and different CoAs. More information can be found in MIPv6 [MIPv6].

[1] In this protocol, one policy request message is related with one policy response message. The entity which initiates a policy request is called a policy requester, and the one which replies is named policy replier.

To control policy information transmission, we could also make use of the field "#Sequence". Every egress link identified by a (SourceAddress, DestAddress) pair maintains its sequence and retransmission. In our studied case, all links are controlled by the same entity. For a policy replier, if it receives a request message with the expected sequence number, it replies the suitable information with the same sequence number; otherwise, it resends the last response since it means the policy requester did not receive the response before. The policy requester could detect the loss of information through "TimeOut"[1] and resend the last Policy message. If the requester receives one response which has been received before, it simply discards that response. Our management mechanism requires that the information package exchange between the MR and the HA is performed one after another for getting rid of multiple simultaneous accesses to PDPs. Next policy information package can be sent out after the response of last policy information package is received.

Therefore a new IP Extension Header, namely Policy Header, is designed with the similar format as Mobility Header. There is no IP-in-IP in this case, only one IP header whose source address is the CoA of the selected interface.

Similar to choice one, multiple information messages can be handled by one information package. **Figure 46** shows the **Policy Header Format.**

PayloadProtocol	HeaderLen	PHtype	Reserved
Checksum		#Sequence	
PolicyData			

Figure 46: Policy Header Format as an IP Extension Header

We can specify the value of the *PayloadProtocol* field for *Policy Header* to be 10, which has not been used in Mobile IPv6 yet.

PHType means Policy Header Type, two possible values are: 0 means *Policy Request* message; 1 means *Policy Response* message.

The other fields have the same meanings as in a Mobility Header.

PolicyData Format:

PolicyData field share the same format as *PolicyMessageData* field described in choice 1.

Frankly speaking, these two choices described in the Section6.3.1 and 6.3.2 have no big differences, no one is superior to the other. In our simulation (the Section7), we implement the choice showed in the Section6.3.2.

6.4. Cooperation between a Mobility Header and a Policy Header

As explained in Section 3.2.1, our policy-based routing framework is based on the

[1] TimeOut is a timer which might trigger some actions when time is out. It is mainly used in some cases like retransmission.

maintenance of all potential connections. NEMO has specified the approach of using BU and BACK messages in the Mobility Header to finish the CoA registration for the whole network onto the HA. After registration, CoA of each interface becomes legal, and the Policy Header bound to an registered interface can be created because of the availability of CoA (The format of policy information in Policy Header is showed in **Figure 40**). **Figure 11** has showed the case of an old interface going off. Our policy messages, in the cases of a new available connection and the interruption of an old connection, could be regarded as an extension of BU/BACK, as they enable the registration of multiple connections by adding interface ID into a message field. Other information, such as rejected host list and cost, can only be sent after a successful registration of the related interface.

7. Simulation Results

After the design of a policy-based routing system, in this section we present our simulation results. From different points of view, four simulation cases are carried out to illustrate the benefits of multihoming and the performance of this routing system.

7.1. Simulation Tool – Network Simulator

Network Simulator [NS2] is a simulation tool developed by UC Berkeley, and is widely used in the field of network technology research. As an open source simulation tool, many extensions targeted at various network technologies, for instance, support for mobile networks, have been added since its creation.

The simulator is written in C++, and uses OTcl as a command and configuration interface. NS2 uses two languages to provide two different simulation requirements. The scenario generation and parameter configuration are carried out by OTcl code, and the detailed simulation of protocols is carried out by C++ code. Additionally, NS2 provides a method to combine these two languages.

7.2. Simulation Architecture

We used NS2 (Network Simulator) to simulate our policy-based routing module. Since, at present, there is still no multiple interface support in NS2, and what we wish to investigate is the routing part which could not be influenced by the mobility of nodes, we decided to use the basic ‘node’ instead of mobile nodes, although our module is designed for a mobile network. The simulation topology is described in **Figure 47**.

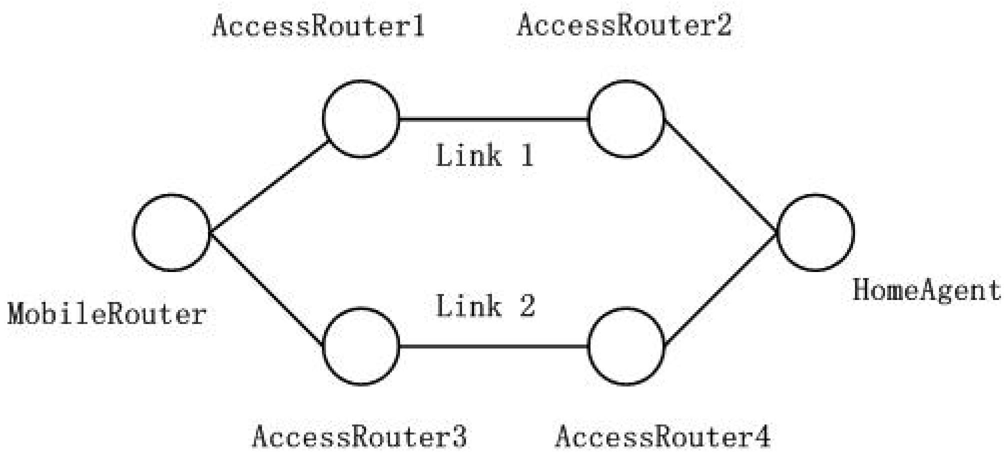


Figure 47: Simulation Topology

All the nodes in this topology are basic nodes defined in NS2. Two tunnels, passing through Link 1 and Link 2, are set up between the MobileRouter and the HomeAgent. AccessRouter1 and AccessRouter3 are used to simulate the two interfaces of the MobileRouter. The tunnel selection is thus simulated as an Access Router Selection. Link 1 covers AccessRouter1 and AccessRouter2; Link 2 covers Access Router3 and

Access Router4. Both links are bidirectional. Initially, both of them are set up in the simulation, and Link 1 is the default path for all devices in the mobile network supported by the MR, because the cost of Link 1 is lower than that of Link 2.

In addition, we designed two new types of Agent: MRPolicyAgent and HAPolicyAgent. In our simulation, a MR is a Node attached with a MRPolicyAgent and a HA is a Node attached with a HAPolicyAgent. MRPolicyAgent and HAPolicyAgent take the responsibility of creating, receiving and processing Policy Messages, and making the routing decisions. They use the policy protocol to communicate with each other.

As explained earlier, in NS2, an object is described in C++ and OTcl. In our simulation, the policy agent object (both MRPolicyAgent and HAPolicyAgent) is designed with the architecture shown in **Figure 48**.

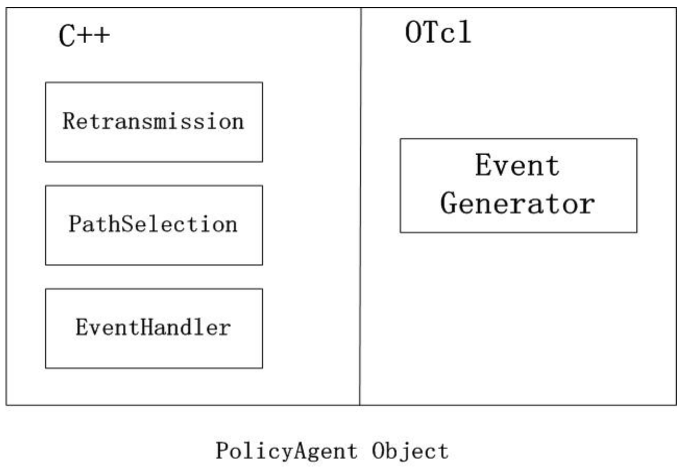


Figure 48: Policy Object in NS2

After attaching policy agents to the basic nodes, we get the architecture of a MR/HA, as shown in **Figure 49**.

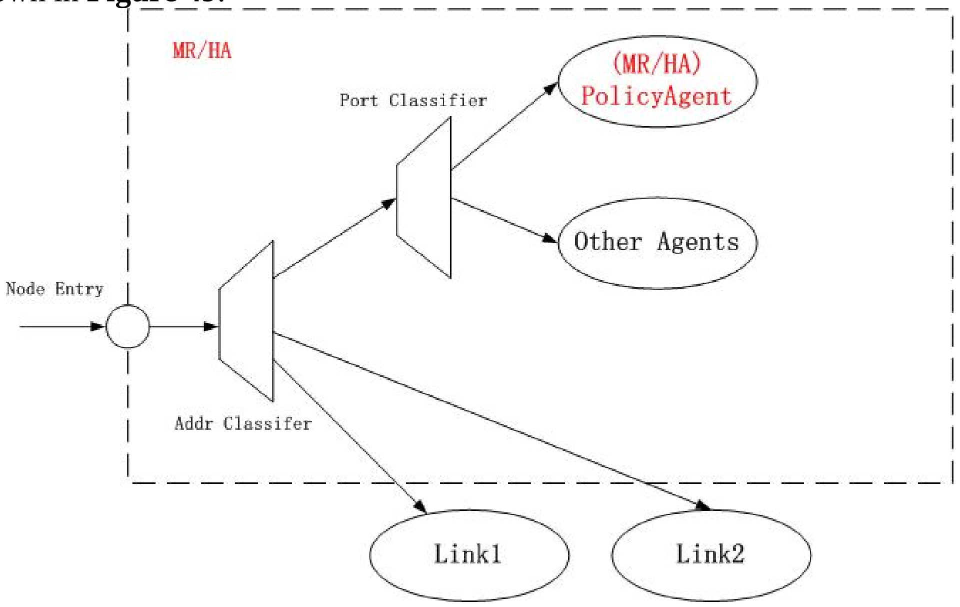


Figure 49: Architecture of a Node attached with Policy Agent

A basic unicast Node in NS2 consists of two objects: an Address Classifier and a Port Classifier. Arriving packets first pass to the Address Classifier, which routes the packets destined to other nodes to the correct link by following its specified routing protocol. If the destination address matches with its IP address, the packets are sent to the Port Classifier where they are passed to an Agent according to their packet type.

Six routing modules are provided in NS2. They are Unicast, Multicast, Hierarchical, Manual, Virtual Circuit and MPLS. As we are going to simulate policy-based routing, Manual routing module is implemented in the simulation.

7.3. Simulation Cases

The simulation topology has been described in **Figure 47**. The basic parameter setting of this topology is shown in **Table 4**. The size of a policy request message is 30bytes; and the size of a policy response message is 10 bytes.

Table 4: Basic Parameter Setting on Simulation Topology

Parameter Name	Parameter Value
Transmission Delay between every connected two nodes	10ms
BandWidth between every connected two nodes	2.0Mb
Queue Length on each AccessRouter	5000

Before we investigate the performance of our routing algorithm, we first test the routing framework and observe the traffic flow in NS2's Network Animator[1] [NAM]. A TCP connection is set up between the MR and the HA, and the following events are generated:

At time=1 sec, traffic starts.
At time=2 sec, Link 1 turns down, we see both uplink and downlink traffic switch to Link 2.
At time=4 sec, Link 1 turns on, we notice both uplink and downlink traffic switch back to Link 1.
At time=6 sec, the uplink cost of Link 1 varies to be higher than that of Link 2, we find uplink traffic switches to Link 2, and the downlink traffic does not change.
At time=8 sec, the downlink cost of Link 1 also turns higher than that of Link 2, we observe downlink traffic also switches to Link 2.

After checking the correctness of the system, we do some other simulations to show how this system is influenced by other factors, and the benefits of multihomed NEMO.

[1] NS2's Network Animator is a Tcl/TK based animation tool for viewing network simulation traces and real world packet trace data. It is able to read large animation data sets and be used in different visualization situations.

7.3.1. Simulation 1: Policy Message Round Trip Time and Traffic Overhead

To determine how our system is affected by other factors, we investigate how Policy Message Round Trip Time (RTT) and Traffic Overhead are affected by different link situations. This simulation is achieved by sending both policy messages and other traffic on the same link.

RTT is defined as the time interval for a policy packet to travel from its sender to its receiver and then back to the sender; Traffic Overhead is defined as the policy data percentage over all traffic on a link.

We tested the average RTT of sending 500 consecutive policy messages in different link traffic situations and *TimeOut* value settings. *TimeOut* is the specified waiting time for retransmission a previous sent message. The link bandwidth is set as 2Mb, and its delay is 10ms. Traffic with Exponential On/Off distribution is on the link. *On* time (during which the traffic is sent in the given rate) is set to be 500ms and *Off* time (during which the traffic is stopped) is set to be 100ms. The traffic size is set to 4000 bytes. The traffic situation is shown in **Figure 50**.

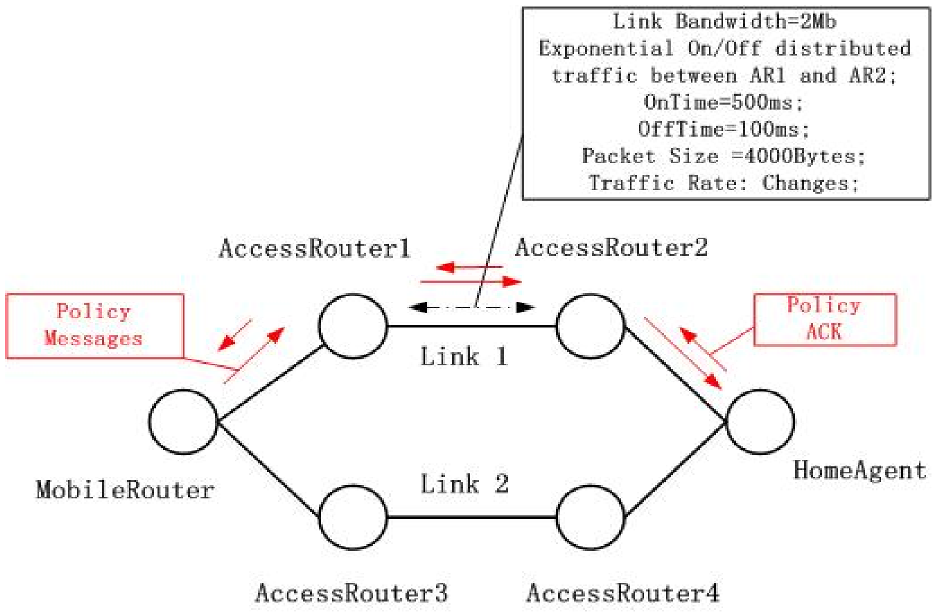


Figure 50: Traffic Chart in Simulation Case 1

In **Figure 51** and **Figure 52**, 4 lines are plotted for different *TimeOut* values: 0.1second, 0.2second, 0.5second and exponentially increasing from 0.5second.

From **Figure 51**, we can see that for a fixed *TimeOut* value, with the increase of link traffic, the RTT of sending a policy message is increased, no matter how *TimeOut* is specified. Meanwhile, we notice, if *TimeOut* is assigned to be too large, it takes a longer time to detect the loss of a policy message, which results in a high RTT. However, since less retransmission messages are sent, the overhead in **Figure 52** is lower. If *TimeOut* is assigned to be too small, which means a large number of policy messages would be sent out for the reason of retransmission and then the link situation turns bad after a short time because the retransmitted messages themselves make the link congested, thus the transmission of almost all of these 500 policy messages are finished on a highly congested link, this explains why the value of RTT in this situation is relatively high. Nevertheless, the overhead in **Figure 52** turns higher with the decrease of *TimeOut*.

In reality, we prefer lower RTT, which means events can be sent to a PDP within a short time, we also prefer to have lower overhead, which would not create too much management traffic. By observing **Figure 51** and **Figure 52**, we notice that these two preference could not be achieved at the same time, if one parameter gets a really good result, the other can not get a very good result, we must choose a trade-off.

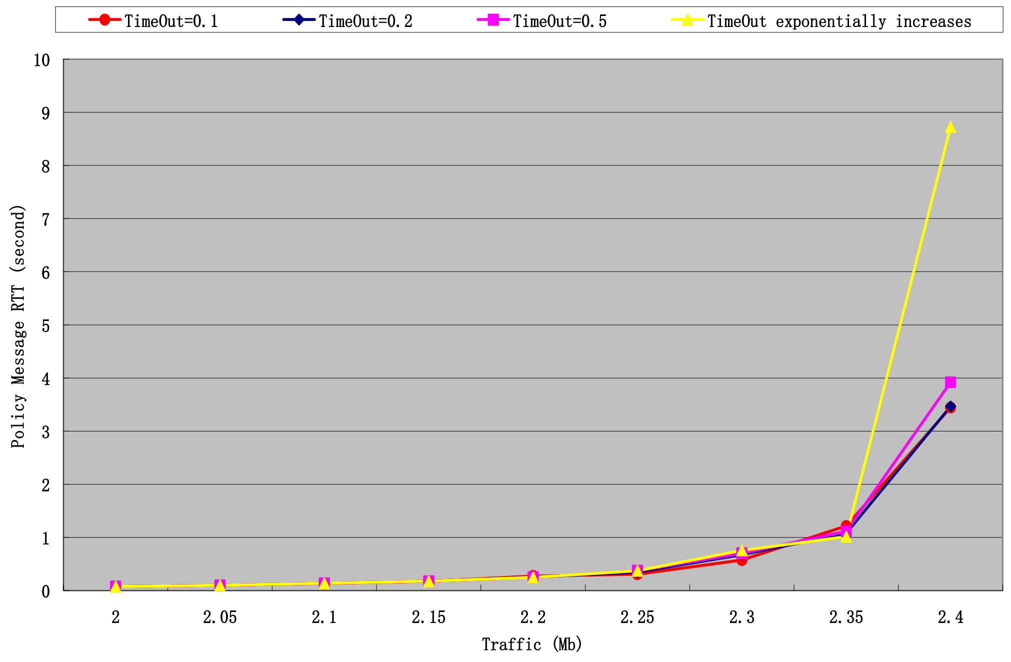


Figure 51: Policy Message RTT

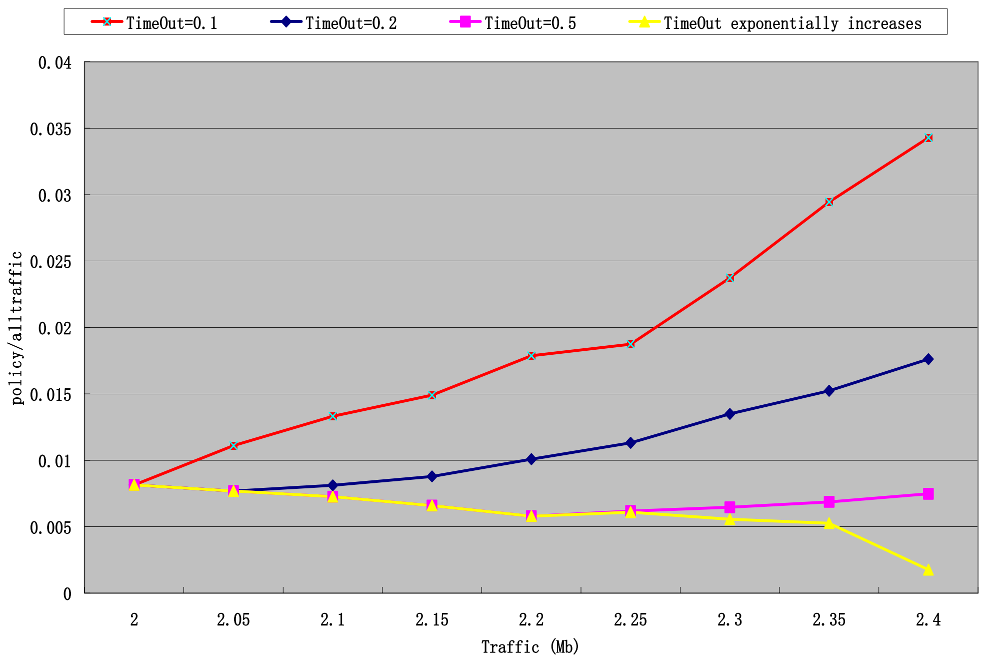


Figure 52: Overhead Ratio over Traffic

7.3.2. Simulation 2: Path Selection Benefit of Multihoming

To show the path selection benefit of multihoming support, in this simulation, we use congestion situation controlled path selection policy: once the throughput we specified two sets of traffics: one is on the link from the AccessRouter1 to the AccessRouter2, the other one is UDP traffic from the MR to the HA which also takes Link 1 as its default link. The simulation runs for 100 seconds, the UDP traffic starts at 0 second and the other traffic starts at 30 second. At the 35 second (as at that moment, the throughput decreases to some value), the MR reselects Link 2 as its path, after that, the UDP traffic flows along Link 2. The traffic situation is shown in **Figure 53** and **Figure 54**.

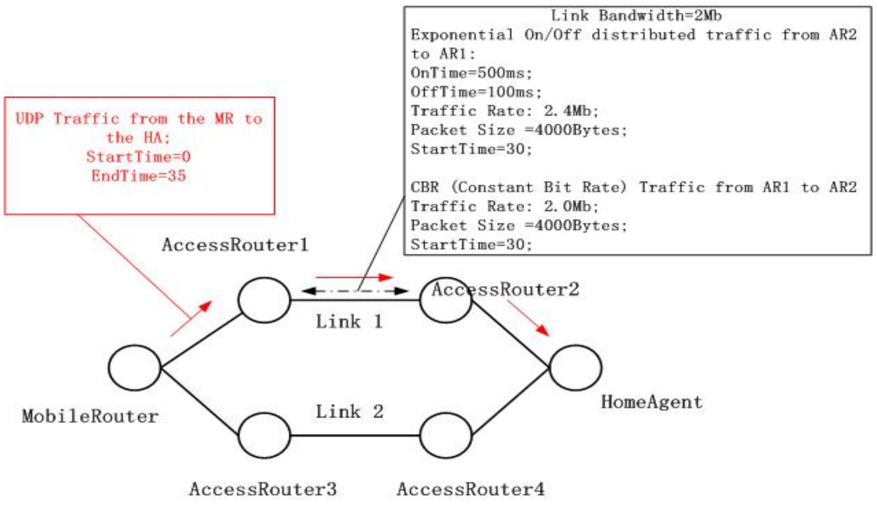


Figure 53: Traffic Chart 1 in Simulation Case 2

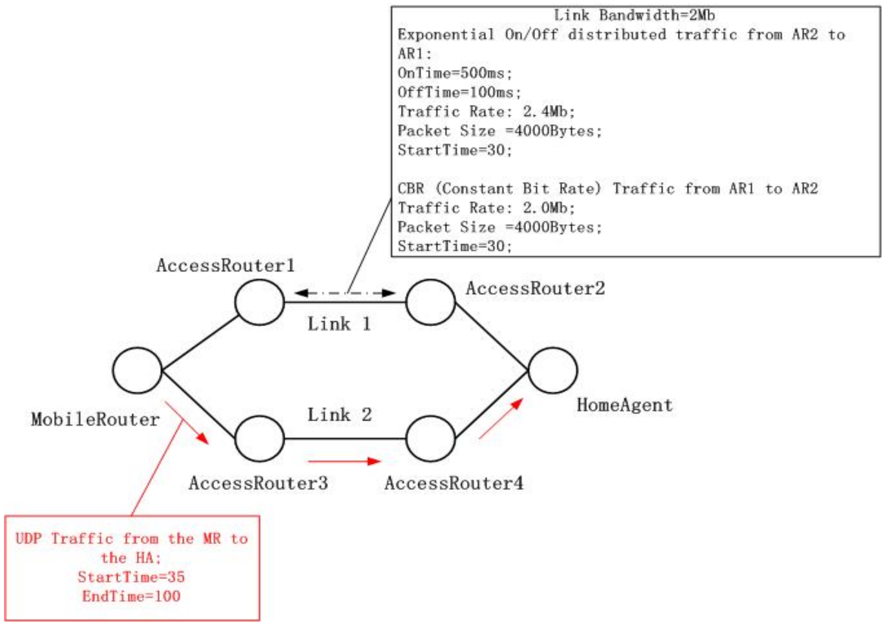


Figure 54: Traffic Chart 2 in Simulation Case 2

In **Figure 55** and **Figure 56**, the throughputs of two traffic are plotted for the cases with and without multihoming support. Here, the throughput is defined in terms of the number of received UDP packets per second.

As shown in **Figure 55** and **Figure 56**, the throughput of the UDP traffic drops at 30 second as the other traffic begins to flow. From 35 second, due to multihoming support, both the UDP, and the other traffic, achieve higher throughput than in the case without multihoming support.

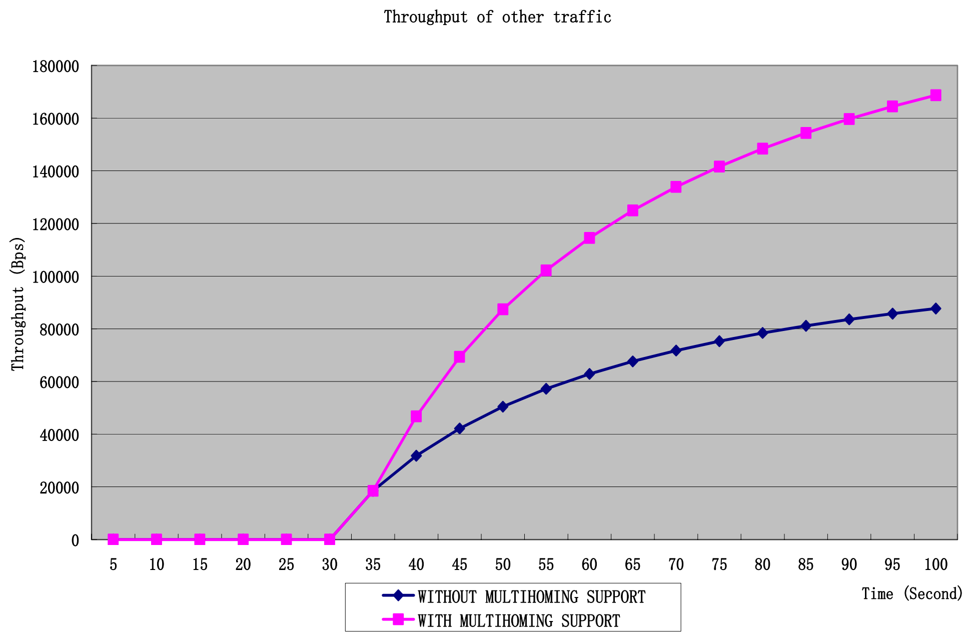


Figure 55: Other Traffic Throughput

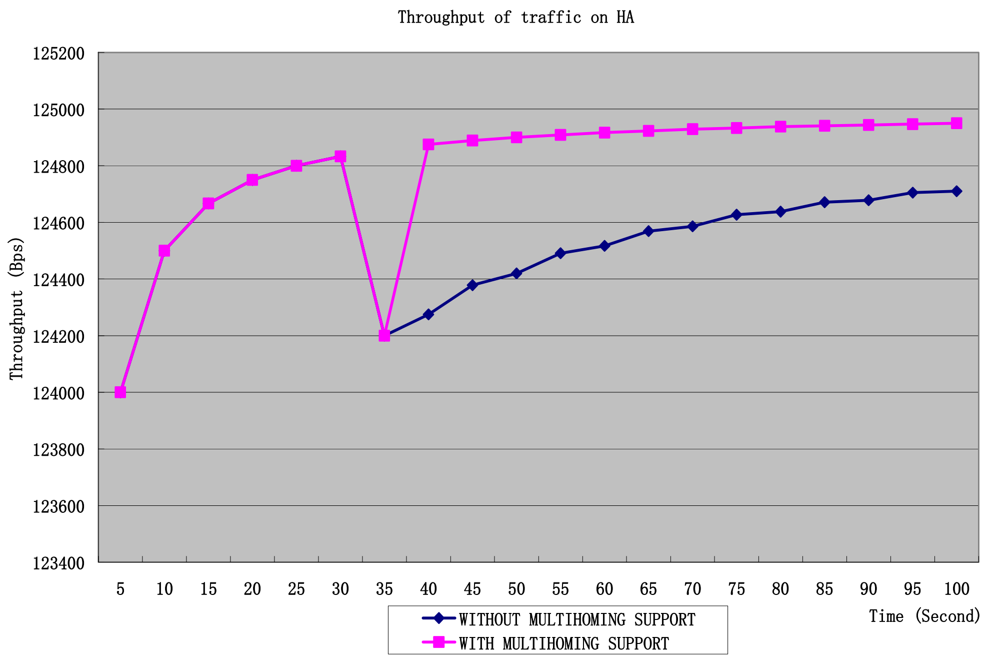


Figure 56: UDP Traffic Throughput

7.3.3. Simulation 3: Handover Delay Effect on TCP Traffic

As mentioned previously in Section 3.2.1, the decrease in handover delay, when multihoming is implemented, is a large advantage. Through the following simulation, we would like to determine the influence the handover delay has on TCP traffic.
We created TCP traffic and an interfering traffic, and both of them make use of Link 1.There is no traffic on Link 2. The interfering traffic is exponentially distributed with a rate of 2.5Mb, and active for a simulation time of 10 seconds. The traffic situation is described in **Figure 57**.

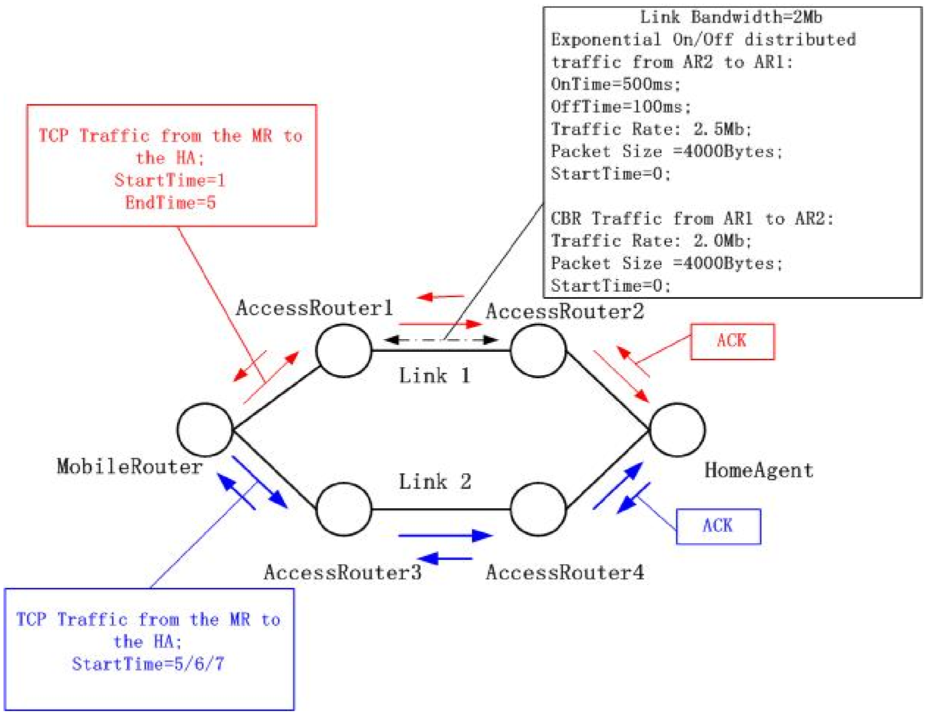


Figure 57: Traffic Chart in Simulation Case 3

The interfering traffic starts at the beginning of the simulation, and the TCP traffic begins at the first second and the handover request is created at 5 second. In **Figure 58**, we plot the TCP throughput when the delay is 0, 1 and 2 seconds respectively. The Delay is defined as the time slot between receiving the handover request and changing the path.

Although the interfering traffic generation rate is relatively high compared with the link bandwidth, the link situation is not extremely bad since our simulation time is not that long. By observing the simulation result, we notice during the first 5 seconds, due to the deterioration of link situation, the throughput of TCP traffic decreases as time goes on. From 5 second on, corresponding to different delays, the TCP traffic is transmitted through the Link 2 and these throughput lines come to their turning points at different time points.

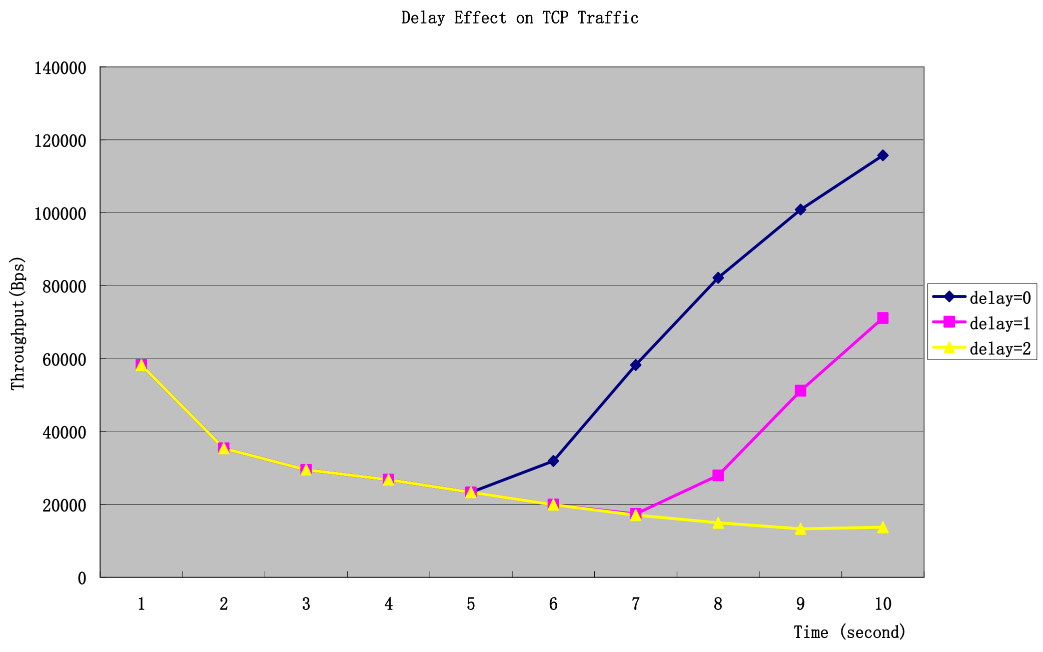


Figure 58: Delay Effect on TCP Traffic

7.3.4. Simulation 4: Packet Type Based Path Selection and Load Sharing

In this simulation, the two links between the MR and the HA are still under different traffic situations shown in **Figure 59**, similar to simulation 3. Link 1 has more non-user traffic[1] but cheap, while the Link 2 has a small delay but expensive. Our policy is that important traffic is carried by Link 2 and less important traffic by the link1. There is a TCP traffic sourced from the HA and destined for the MR. The sending of TCP traffic is decided by the correct receiving of previous ACK messages. Thus, packets with type ACK could be regarded as more important, and should be delivered along the Link 2.

In **Figure 60**, we compare TCP throughput in the case with multihoming support to the case without multihoming support. The results show that multihoming really improves the performance of TCP traffic.

However, even in the case of using type-based routing, the throughput does not continue to increase. As described in **Figure 60** the throughput turns down at 8 second; the reason is that the condition of the link through which TCP packets are delivered deteriorates, thus the average time for a TCP packet to arrive at the destination is longer than before and the number of received TCP packets drops.

The trend of the square line is determined by multiple factors. For instance, the uplink and downlink traffic situations, or the transmission mechanism of TCP. That explains why its throughput varies irregularly.

[1] Non-user traffic is the traffic that we are not interested in.

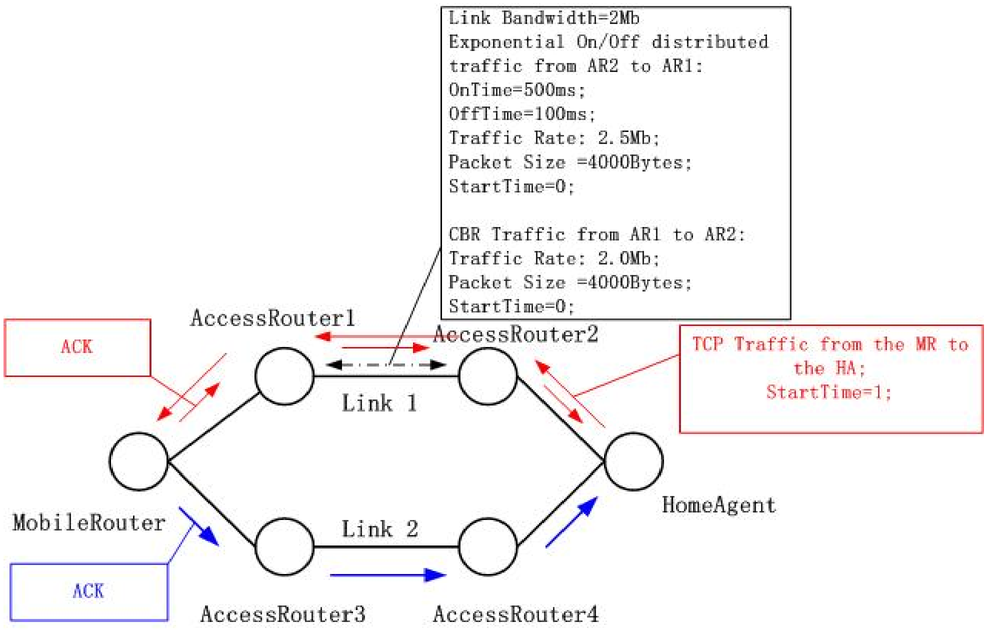


Figure 59: Traffic Chart in Simulation Case 4

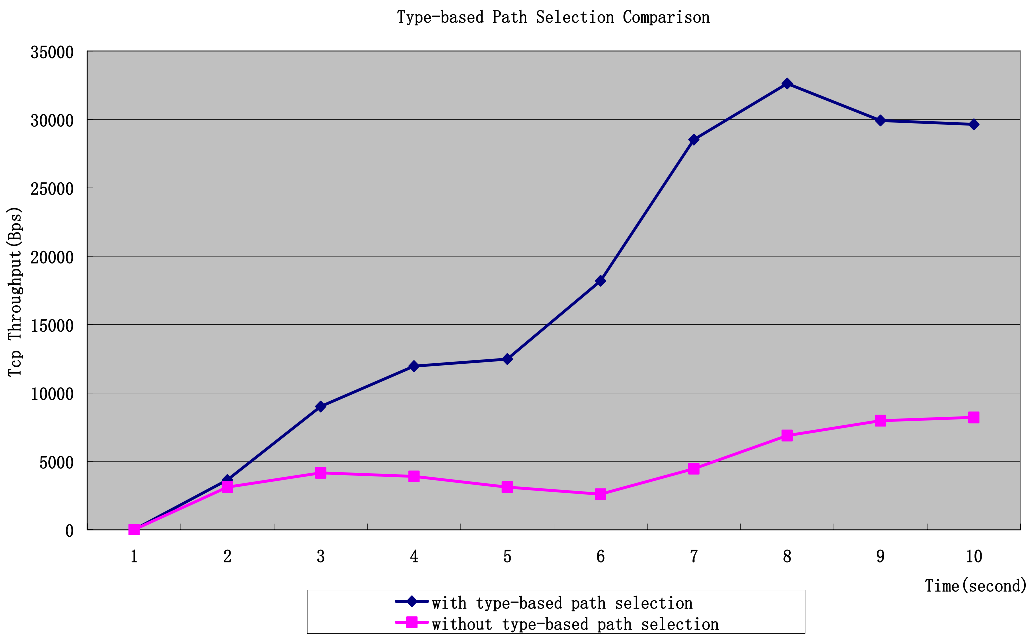


Figure 60: Type-based Path Selection Comparison

7.4. Simulation Conclusions

In this section, after the validation of our policy-based routing system, several simulations are carried out. Simulation 1 shows that the congestion situation and data loss would really affect the delivery of the policy messages, and when deciding the value of TimeOut, there is a trade-off between RTT and overhead of the system. Simulation 2 shows that with the multihoming support, traffic throughput can be improved by path selection. In Simulation 3, we recall the benefit of decreasing the connection reestablishment time because of the multihoming support, and show the connection reestablishment greatly influence the performance. Simulation 4 is a combination of path selection and load sharing, because of the right setting of path for different types of packets, the performance is largely increased.

8. Conclusions and Future Works

8.1. Conclusions

Through simulation, in this report, we can come to the conclusion that multihoming support in MIPv6 Networks does provide a lot of benefits, like redundancy, load-sharing and policy-based routing, which largely improves the whole performance of a mobile network.

However, the discussion of multihoming classification also indicated the difficulty of providing multihoming support since many other mechanisms have to be added and no one solution is suitable for all multihoming scenarios. As shown by simulation results, our policy-based routing framework, designed for one scenario (one MR, one HA, one MNP), supports policy-based routing for both inbound and outbound traffic, and achieves the multihoming benefits, by allocating various policy components on to the MR and the HA. Our detailed policies enable the policy-based routing on the basis of interface availability, connection quality, cost, and user preference. The delivery of policy information between different entities is also guaranteed by adding them into an IP extension header. Through the simulation, it was also found that the network situation had a large influence on the performance of the proposed policy-based routing framework, because the proper delivery of policy information is a crucial part of this framework.

Our policy-based routing system is suitable for a fixed cluster in a PN. The split and mix of a cluster is not considered.

8.2. Future Works

In this thesis, our policy-based routing framework is built up on the multihoming configuration class 1 of a mobile network proposed in the NEMO Basic Support. Research of other classes is also interesting, especially the cases with multiple Mobile Routers in which each MNN has to select its Mobile Router. One question is how to integrate information from different MRs and MNNs to complete a policy-based routing system. Intra-domain Network Mobility solutions might also be involved for these topologies. Works carried out by MANET (Mobile Ad-hoc Networks) group could be referred to. Some other discussions about multiple MRs and multiple HAs can be found in [HAHA], [DHAHA].

Meanwhile, our framework is limited by some assumptions, i.e. no route optimization. In the future, when the mature route optimization methods are entirely designed for a mobile network, we could also investigate the possibility of implementing a policy-based routing system with route optimization support.

Another possible future work is to study the multihoming case which combines uni-directional (e.g. Digital Video Broadcast) and bi-directional access technologies (which is discussed in this thesis).

References

[AMNMS] C.Ng, E.Paik and T.Ernst, “Analysis of Multihoming in Network Mobility Support”, Internet Draft, October 2004.

[ANAPN] Martin Jacobsson, Jeroen Hoebeke, Sonia Heemstra de Groot, Anthony Lo, Ingrid Moerman, Ignas niemegeers, Luis Munoz, Mikko Alutoin, Wajdi Louati, Djamal Zeghlache, “A Network Architecture for Personal Networks”, 2005.

[APFIDSI] Raju Rajan, Dinesh Verma, Sanjay Kamat, Eyal Felstaine, Shai Herzog, “A Policy Framework for Integrated and Differentiated Service in the Internet”.

[ASPSA] Nicodemos Damianou, Arosha K Bandara, Morris Sloman and Emil C Lupu, “A Survey of Policy Specification Approaches”, April 2002.

[COClass] Thierry Ernst and Julien Charbon, “Multihoming with NEMO Basic Support”

[COPS] D.Durham, Ed., J.Boyle, R.Cohen, S.Herzog, R.Rajan and A.Sastry, “RFC 2748- The COPS (Common Open Policy Service) Protocol”, January 2000.

[CN-ATDAFI] James F.Kurose, Keith W.Ross, “Computer Networking – A Top-Down Approach Featuring the Internet” (Second Edition), Addison Wesley, 2003.

[DHAHA] B.Koh, C.Ng, J.Hirano, “Dynamic Inter Home Agent Protocol”, Nov 2004.

[DHCP] R.Droms, “Dynamic Host Configuration Protocol”, RFC 2131, March 1997.

[DHCPv6] R.Droms, J.Bound, B.Volz, T.Lemon, C.Perkins and M.Carney, “RFC3315 – Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, July 2003.

[EMMRMNPN] R.Kuntz, E.Paik, M.Tsukada, T.Ernst, K.Mitsuya, “Evaluating Multiple Mobile Routers and Multiple NEMO-Prefixes in NEMO draft-kuntz-nemo-multihoming-test-00.txt”, July 2004.

[EMSNBS] J.Charbon, C-W.Ng, K.Mitsuya and T.Ernst, “Evaluating Multi-homing Support in NEMO Basic Solution”, Internet Draft, July 2003.

[FHMIP6] Rajeev Koodli, “Fast Handovers for Mobile IPv6”, work in progress, October 2004.

[FlowMo] Hesham Soliman, Karim ElMalki and Claude Castelluccia, “Flow movement in Mobile IPv6”, June 2003.

[GBM] T.Ernst, N.Montavont, R.Wakikawa, E.Paik, C.Ng, K.Kuladinithi, T.Noel, “Goals and Benefits of Multihoming”, January 2005.

[HAHA] Ryuji Wakikawa, Vijay Devarapalli, and Pascal Thubert, “Inter Home Agents Protocol (HAHA)”, Feb 2004.

[HIP] R.Moskowitz and P.Nikander, “Host Identity Protocol Architecture, draft-ietf-hIP-arch-02”, January 2004.

[HMIP] Pekka Nikander, Janne Lundberg, Catharina Candolin and Tuomas Aura, “Homeless Mobile IPv6”, Internet-Draft, February 2001.

[IFMN] F.Bakere, P.Savola, “RFC3704 – Ingress Filtering for Multihomed Networks”, March 2004.

[LDAP] W. Yeong, T.howes and S.Kille, “RFC1777- Lightweight Directory Access Protocol”, March 1995.

[LIN6] Fumio Teraoka, Masahiro Ishiyama, Mitsunobu Kunishi, and Atsushi Shionozaki, “LIN6: A Solution to Multi-Homing and Mobility in IPv6”, INTERNET DRAFT, December 2003.

[LSSPMMR] Eun Kyoung Paik, Hosik cho, Thierry Ernst and Yanghee Choi, “Load Sharing and Session Preservation with Multiple Mobile Routers for Large Scale Network Mobility”.

[MAGNET] http://www.telecom.cec.ntua.gr/magnet

[MCARMIP] Ryuji Wakikawa, Keisuke Uehara and Thierry Ernst, “Multiple Care-of Address Registration on Mobile IPv6”, Internet Draft draft-wakikawa-mobileip-multiplecoa-02, September 2003.

[MHIBT] C.Ng and J.Charbon, “MultiHoming Issuse in Bi-directional Tunneling”, Internet Draft, May 2003.

[MHTP] M.Py, “Multi Homing Translation Protocol (MHTP)”, INTERNET DRAFT.

[MIP] C.Perkins, “IP Mobility Support”, RFC 2002, October 1996.

[MIPv4] C.Perkins, “IP Mobility Support for IPv4”, RFC 3220, January 2002.

[MIPv6] D.Johnson, C.Perkins and J.Arkko, “Mobility Support in IPv6”, RFC 3775, June 2004.

[MNAMIP] Thierry Ernst, Alexis Olivereau, Ludovic Bellier, Claude Castelluccia, Hong-Yon Lach, “Mobile Networks Support in Mobile IPv6 (Prefix Scope Binding Updates)”, March 2002.

[MNISPBR] R.Wakikawa, K.Uehara and Jun Murai, “Multiple Network Interfaces Support By Policy-Based Routing on Mobile IPv6”.

[MNMN] Nicolas Montavont, Thierry Ernst, and Thomas Noel, “Multihoming in Nested Mobile Networking”.

[MTCP] Christian Huitema, “Multi-homed TCP”, Internet Draft, May 1995, expired.

[MultiClass] C.Ng, E.Paik and T.Ernst, "Analysis of Multihoming in Network Mobility Support draft-ietf-nemo-multihoming-issues-02", Work in Progress, February 2005.

[NAM] http://www.isi.edu/nsnam/ns/

[NEMO] V.Devarapalli, R.Wakikawa, A.Petrescu and P.Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, January 2005.

[NMROPS] C.Ng, P.Thubert, M.Watari, F.Zhao, and UC Davis," Network Mobility Route Optimization Problem Statement draft-ietf0nemo-ro-problem-statement-00", July 2005.

[NMSGR] T.Ernst, "Network Mobility Support Goals and Requirements draft-ietf-nemo-requirements-03", October 2004.

[NPL-ASNA] Gary N.Stone, Bert Lundy, and Geoffrey G.Xie, "Network Policy Languages: A Survey and a New Approach", 2001.

[NROPSA] F.Zhao, S F.Wu, UC Davis and S.Jung, "NEMO Route Optimization Problem Statement and Analysis draft-zhao-nemo-ro-ps-01", Februray 2005.

[NS2] http://www.isi.edu/nsnam/ns/

[POLICY] R.Yavatkar, D.Pendarakis and R.Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000; Westerinen, j.Schnizlein, J.Strassner, M.Scherling, B.Quinn, S.Herzog, A.Huynh, M.Carlson, J.Perry and S.Waldbusser, "Terminology for Policy-Based Management", RFC 3198, November 2001.

[SCTP] R.R.Stewart, Q.Xie, K.Moreault, C.Sharp, H.J.Schwarzbauer, T.Taylor, I.Rytina, M.Kalla, L.Zhang, and V.Paxson, "Stream Control Transmission Protocol, RFC 2960", October 2000.

[SNMS] Eranga perera, Vijay Sivaraman and Aruna Seneviratne, "Survey on Network Mobility Support", 2004.

Appendix - Codes and Scripts

Simulation1: RTT of a policy message TCL Script

```
#Create a simulator object
#debug 1
remove-all-packet-headers
add-packet-header Mac TCP IP Common Policy
set ns [new Simulator]

#Open the nam and output trace file , by weiwei
set f  [open out.tr w]
#set nf [open outJune6.nam w]
#$ns namtrace-all $nf
$ns trace-all $f

#Define a 'finish' procedure
proc finish {} {
        global ns nf f
        $ns flush-trace
 #Close the trace file
 #       close $nf
         close $f
 #Execute nam on the trace file
 #exec nam outJune6.nam &
        exit 0
}

#Create seven nodes
$ns rtproto Manual
set cn_ [$ns node]
#debug 1
set ha_ [$ns node]  ;# address is 1
set mr_ [$ns node]   ;#address is 2
set ar1_ [$ns node]  ;# connect to MR,interface 1, address is 3
set ar2_ [$ns node] ;# connect to MR, interface 2, address is 4
set ar3_ [$ns node] ;# connect to HA, interface 1, address is 5
set ar4_ [$ns node];# connect to HA, interface 2, address is 6


 $ns duplex-link $ar3_ $ha_ 2mb 10ms DropTail
[$ar3_ get-module "Manual"] add-route-to-adj-node $ha_
[$ha_ get-module "Manual"] add-route-to-adj-node -default $ar3_

$ns duplex-link $ar4_ $ha_ 2mb 10ms DropTail
[$ar4_ get-module "Manual"] add-route-to-adj-node $ha_
[$ha_ get-module "Manual"] add-route-to-adj-node  $ar4_

$ns duplex-link $ar1_ $mr_ 2mb 10ms DropTail
[$ar1_ get-module "Manual"] add-route-to-adj-node $mr_
[$mr_ get-module "Manual"] add-route-to-adj-node  -default $ar1_

$ns duplex-link $ar2_ $mr_ 2mb 10ms DropTail
[$ar2_ get-module "Manual"] add-route-to-adj-node $mr_
[$mr_ get-module "Manual"] add-route-to-adj-node  $ar2_

$ns duplex-link $ar1_ $ar3_ 2mb 10ms DropTail
$ns queue-limit $ar1_ $ar3_ 5000
$ns queue-limit $ar3_ $ar1_ 5000
[$ar1_ get-module "Manual"] add-route-to-adj-node -default $ar3_
[$ar3_ get-module "Manual"] add-route-to-adj-node -default $ar1_

$ns duplex-link $ar2_ $ar4_ 2mb 10ms DropTail
[$ar2_ get-module "Manual"] add-route-to-adj-node -default $ar4_
[$ar4_ get-module "Manual"] add-route-to-adj-node -default $ar2_

set mragent [eval new Agent/MRPolicyAgent]
set haagent [eval new Agent/HAPolicyAgent]
$ns attach-agent $mr_ $mragent
$ns attach-agent $ha_ $haagent

#Create a TCP sender agent and attach it to node cn_
set tcp [new Agent/TCP]
$ns attach-agent $ha_ $tcp
#Create a TCP receiver agent and attach it to node mr_
set sink [new Agent/TCPSink]
$ns attach-agent $mr_ $sink

#Create a FTP application
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# #$ftp attach-agent $mragent
$ns connect $tcp $sink

#additional setting for creating all kinds of link situation
#set up a udp connection between ar2 and ar4
set udp [new Agent/UDP]
$ns attach-agent $ar1_ $udp
set null [new Agent/Null]
$ns attach-agent $ar3_ $null
$ns connect $udp $null
$udp set fid_ 0
set e [new Application/Traffic/Exponential]
$e set packetSize_ 4000
$e set burst_time_ 500ms
$e set idle_time_ 100ms
$e set rate_ 2.25mb
$e attach-agent $udp

set udp2 [new Agent/UDP]
$ns attach-agent $ar3_ $udp2
set null2 [new Agent/Null]
$ns attach-agent $ar1_ $null2
$ns connect $udp2 $null2
$udp set fid_ 0

set e2 [new Application/Traffic/Exponential]
$e2 set packetSize_ 4000
$e2 set burst_time_ 500ms
$e2 set idle_time_ 100ms
$e2 set rate_ 2.25mb
$e2 attach-agent $udp2
#$ns at 1.0 "$ftp start"  ;# FTP start at 1.2 seconds
#
$ns at 1.0 "$e start"
$ns at 1.0 "$e2 start"
$ns at 2.0 "$mragent bad 1"

# THE COMMENTED PART IS USED FOR VAILIDATION
#$ns at 2.5 "$cbr stop"
#$ns at 4.0 "$mragent good 1"
#$ns at 6.0 "$cbr stop"
#$ns at 6.0 "$haagent uplinkcostchange 1 3"
#$ns at 8.0 "$haagent downlinkcostchange 1 3"
# #Call the finish procedure after 5 seconds of simulation time
$ns at 20000.0 "finish"

#Run the simulation
$ns run
```

Simulation1: awk Script(1)

```
cat out.tr | awk '{
        {
          {
         if (($1== "+") && ($3 == "2") && ($5 == "policy1") )
          print $5, $11, $2;
```

```
       }
       {
       if (($1== "r") && ($4 == "2") && ($5 == "policy2") )
          print $5, $11, $2
       }

       }

}' > policytime.seq

cat out.tr | awk '
         BEGIN {
       alltraffic=0;
       allpack=0;
       allpreq=0;
       loss=0;
       }

       {
         {
         if (($1== "+") && ($5 == "exp"))
         alltraffic++;
       }
         if (($1== "+") && ($3 == "1") && ($5 == "policy2") )
           {
                   alltraffic++;
                   allpack++;
           }
            if (($1== "+") && ($3 == "2") && ($5 == "policy1") )
           {
                   alltraffic++;
                   allpreq++;
           }
        if ($1 == "d")
       {loss++;
        }
       }
        END {
         print allpack+allpreq, alltraffic, (allpack+allpreq)/alltraffic, loss, loss/alltraffic
       }
' > alltraffic.seq
```

Simulation1: awk Script(2)

```
cat policytime.seq | awk '

                   BEGIN {
                           sum=0;
                           i=1;
                           aver=0.0;
                           last=0;
                           flag=1;
                           }
                   {
                   if ($1 == "policy1" && $2 == i && flag==1)
                   {sum=sum - $3;
                   last=$3;
                   flag=0;
                    }
                   if ($1 == "policy2" && $2 == i && flag==0 )
                   {sum=sum + $3;
                   flag=1;
                   i++;}
                   }
                   END {
                        i--;
                        if (flag==0) sum=sum+last;
                        aver= sum/i;
                        print sum, i, aver
                   }
' > avertime.seq
```

Simulation2: Path Selection TCL Script

```
#Create a simulator object
#debug 1
remove-all-packet-headers
add-packet-header Mac TCP IP Common Policy
set ns [new Simulator]

#Open the nam and output trace file , by weiwei
set f  [open out.tr w]
set nf [open outJune6.nam w]
$ns namtrace-all $nf
$ns trace-all $f

#Define a 'event' procedure
proc event {e} {
   global ns
   [$e set mrpolicyagent_] badinterface 1

}
#Define a 'finish' procedure
proc finish {} {
        global ns nf f
        $ns flush-trace
 #Close the trace file
      close $nf
         close $f
 #Execute nam on the trace file
exec nam outJune6.nam &
        exit 0
}

#Create seven nodes
$ns rtproto Manual
set cn_ [$ns node]
#debug 1
set ha_ [$ns node]  ;# address is 1
set mr_ [$ns node]   ;#address is 2
set ar1_ [$ns node]  ;# connect to MR,interface 1, address is 3
set ar2_ [$ns node] ;# connect to MR, interface 2, address is 4
set ar3_ [$ns node] ;# connect to HA, interface 1, address is 5
set ar4_ [$ns node];# connect to HA, interface 2, address is 6


 $ns duplex-link $ar3_ $ha_ 2mb 10ms DropTail
[$ar3_ get-module "Manual"] add-route-to-adj-node $ha_
[$ha_ get-module "Manual"] add-route-to-adj-node -default $ar3_

$ns duplex-link $ar4_ $ha_ 2mb 10ms DropTail
[$ar4_ get-module "Manual"] add-route-to-adj-node $ha_
[$ha_ get-module "Manual"] add-route-to-adj-node  $ar4_

$ns duplex-link $ar1_ $mr_ 2mb 10ms DropTail
[$ar1_ get-module "Manual"] add-route-to-adj-node $mr_
[$mr_ get-module "Manual"] add-route-to-adj-node  -default $ar1_

$ns duplex-link $ar2_ $mr_ 2mb 10ms DropTail
[$ar2_ get-module "Manual"] add-route-to-adj-node $mr_
[$mr_ get-module "Manual"] add-route-to-adj-node  $ar2_

$ns duplex-link $ar1_ $ar3_ 2mb 10ms DropTail
#$ns queue-limit $ar1_ $ar3_ 5000
#$ns queue-limit $ar3_ $ar1_ 5000
[$ar1_ get-module "Manual"] add-route-to-adj-node -default $ar3_
[$ar3_ get-module "Manual"] add-route-to-adj-node -default $ar1_
```

```
$ns duplex-link $ar2_ $ar4_ 2mb 10ms DropTail
[$ar2_ get-module "Manual"] add-route-to-adj-node -default $ar4_
[$ar4_ get-module "Manual"] add-route-to-adj-node -default $ar2_


set mragent [eval new Agent/MRPolicyAgent]
set haagent [eval new Agent/HAPolicyAgent]
$ns attach-agent $mr_ $mragent
$ns attach-agent $ha_ $haagent

#Create a TCP sender agent and attach it to node cn_
set tcp [new Agent/TCP]
$ns attach-agent $ha_ $tcp


#additional setting for creating all kinds of link situation
#set up a udp connection between ar2 and ar4

#udp traffic from mr to ha
set udp_mr [new Agent/UDP]
$ns attach-agent $mr_ $udp_mr
set null_ha [new Agent/Null]
$ns attach-agent $ha_ $null_ha
$ns connect $udp_mr $null_ha
$udp_mr set fid_ 0

#set up a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp_mr
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1.0mb
$cbr set random_ true

set udp [new Agent/UDP]
$ns attach-agent $ar1_ $udp
set null [new Agent/Null]
$ns attach-agent $ar3_ $null
$ns connect $udp $null
$udp set fid_ 0

set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp
$cbr2 set type_ CBR
$cbr2 set packet_size_ 4000
$cbr2 set rate_ 2.0mb
$cbr2 set random_ true

set udp2 [new Agent/UDP]
$ns attach-agent $ar3_ $udp2
set null2 [new Agent/Null]
$ns attach-agent $ar1_ $null2
$ns connect $udp2 $null2
$udp set fid_ 0

set e2 [new Application/Traffic/Exponential]
$e2 set packetSize_ 4000
$e2 set burst_time_ 500ms
$e2 set idle_time_ 100ms
$e2 set rate_ 2.4mb
$e2 attach-agent $udp2

$ns at 0.0 "$cbr start"
$ns at 30.0 "$cbr2 start"
$ns at 30.0 "$e2 start"
$ns at 35.0 "$mragent bad 1"
$ns at 100.0 "finish"
#Run the simulation
$ns run
```

Simulation2: Path Selection, awk script for calculating udp traffic throughput and interfering traffic throughput

```
cat out.tr | awk '
         BEGIN{

         hareceived=0;
         time=5;
         }
    {
        if ($2 > time) {print time ,hareceived*1000/time;
                        time=time+5;}

         if (($1 == "r") && ($4 == "1") && ($9 == "2.2") && ($10 == "1.2"))
         hareceived++;

    }
    END{
          print time, hareceived*1000/time;
}' > policyudp.seq

cat out.tr | awk '
         BEGIN{

         noisereceived=0;
         time=5;
         }
    {
        if ($2 > time) {print time ,noisereceived*1000/time;
                        time=time+5;}

         if (($1 == "r") && ($4 == "5") && ($9 == "3.0") && ($10 == "5.0"))
         noisereceived++;

    }
    END{
          print time, noisereceived*1000/time;
}' > policyudpnoise.seq
```

Simulation3: Connection Reestablishment TCL Script

```
#Create a simulator object
#debug 1
remove-all-packet-headers
add-packet-header Mac TCP IP Common Policy
set ns [new Simulator]

#Open the nam and output trace file , by weiwei
set f  [open out.tr w]
set nf [open outJune6.nam w]
$ns namtrace-all $nf
$ns trace-all $f

#Define a 'finish' procedure
proc finish {} {
        global ns nf f
        $ns flush-trace
 #Close the trace file
      close $nf
         close $f
 #Execute nam on the trace file
#exec nam outJune6.nam &
        exit 0
}

#Create seven nodes
$ns rtproto Manual
set cn_ [$ns node]
#debug 1
```

```
set ha_ [$ns node]  ;# address is 1
set mr_ [$ns node]   ;#address is 2
set ar1_ [$ns node]  ;# connect to MR,interface 1, address is 3
set ar2_ [$ns node] ;# connect to MR, interface 2, address is 4
set ar3_ [$ns node] ;# connect to HA, interface 1, address is 5
set ar4_ [$ns node];# connect to HA, interface 2, address is 6

 $ns duplex-link $ar3_ $ha_ 2mb 10ms DropTail
[$ar3_ get-module "Manual"] add-route-to-adj-node $ha_
[$ha_ get-module "Manual"] add-route-to-adj-node -default $ar3_

$ns duplex-link $ar4_ $ha_ 2mb 10ms DropTail
[$ar4_ get-module "Manual"] add-route-to-adj-node $ha_
[$ha_ get-module "Manual"] add-route-to-adj-node  $ar4_

$ns duplex-link $ar1_ $mr_ 2mb 10ms DropTail
[$ar1_ get-module "Manual"] add-route-to-adj-node $mr_
[$mr_ get-module "Manual"] add-route-to-adj-node  -default $ar1_

$ns duplex-link $ar2_ $mr_ 2mb 10ms DropTail
[$ar2_ get-module "Manual"] add-route-to-adj-node $mr_
[$mr_ get-module "Manual"] add-route-to-adj-node  $ar2_

$ns duplex-link $ar1_ $ar3_ 2mb 10ms DropTail
#$ns queue-limit $ar1_ $ar3_ 5000
#$ns queue-limit $ar3_ $ar1_ 5000
[$ar1_ get-module "Manual"] add-route-to-adj-node -default $ar3_
[$ar3_ get-module "Manual"] add-route-to-adj-node -default $ar1_
[$ns link $ar1_ $ar3_] dynamic
[$ns link $ar3_ $ar1_] dynamic

$ns duplex-link $ar2_ $ar4_ 2mb 10ms DropTail
[$ar2_ get-module "Manual"] add-route-to-adj-node -default $ar4_
[$ar4_ get-module "Manual"] add-route-to-adj-node -default $ar2_

set mragent [eval new Agent/MRPolicyAgent]
set haagent [eval new Agent/HAPolicyAgent]
$ns attach-agent $mr_ $mragent
$ns attach-agent $ha_ $haagent

#Create a TCP sender agent and attach it to node cn_
set tcp [new Agent/TCP]
$ns attach-agent $ha_ $tcp

#Create a TCP receiver agent and attach it to node mr_
set sink [new Agent/TCPSink]
$ns attach-agent $mr_ $sink

#Create a FTP application
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns connect $tcp $sink

#additional setting for creating all kinds of link situation
#set up a udp connection between ar2 and ar4

set udp [new Agent/UDP]
$ns attach-agent $ar1_ $udp
set null [new Agent/Null]
$ns attach-agent $ar3_ $null
$ns connect $udp $null
$udp set fid_ 0

set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp
$cbr2 set type_ CBR
$cbr2 set packet_size_ 4000
$cbr2 set rate_ 2.0mb
$cbr2 set random_ true

set udp2 [new Agent/UDP]
```

```
$ns attach-agent $ar3_ $udp2
set null2 [new Agent/Null]
$ns attach-agent $ar1_ $null2
$ns connect $udp2 $null2
$udp set fid_ 0

set e2 [new Application/Traffic/Exponential]
$e2 set packetSize_ 4000
$e2 set burst_time_ 500ms
$e2 set idle_time_ 100ms
$e2 set rate_ 2.5mb
$e2 attach-agent $udp2
$ns at 0.0 "$ftp start"  ;# FTP start at 1.2 seconds

$ns at 0.0 "$cbr2 start"
$ns at 0.0 "$e2 start"

$ns at 5.0 "$mragent bad 1"
#debug 1
$ns at 5.0 "[$ns link $ar1_ $ar3_] down"
$ns at 5.0 "[$ns link $ar3_ $ar1_] down"
#$ns rtmodel-at 5.0 down $ar1_ $ar3_
$ns at 10.0 "finish"

#Run the simulation
$ns run
```

Simulation 3: awk script (Also for Simulation 4)

```
cat out.tr | awk '
    BEGIN{
         i=0;
         time=1;
         }
{
  if (($1 == "d") && (($5 == "tcp")||($5 == "ack")))
   i++;
}
END {print i;}
' > tcpdropnum

cat out.tr | awk '
BEGIN {
   tcpnum=0;
   time=1;
}
{
if ($2 > time ) {
print time, tcpnum*1040/time;
time=time+1;
}
if (($1 == "r") && ($4 == "2") && ($9 == "1.1") && ($10 == "2.1"))
tcpnum++;
}
END {
  print time,tcpnum*1040/time;
}' > tcpthroughput
```

Simulation 4: TCL Script

```
#Create a simulator object
#debug 1
remove-all-packet-headers
add-packet-header Mac TCP IP Common Policy
set ns [new Simulator]

#Open the nam and output trace file , by weiwei
set f  [open out.tr w]
set nf [open outJune6.nam w]
```

```
$ns namtrace-all $nf
$ns trace-all $f

#Define a 'finish' procedure
proc finish {} {
        global ns nf f
        $ns flush-trace
 #Close the trace file
      close $nf
         close $f
 #Execute nam on the trace file
exec nam outJune6.nam &
        exit 0
}

#Create seven nodes
$ns rtproto Manual
set cn_ [$ns node]
#debug 1
set ha_ [$ns node]  ;# address is 1
set mr_ [$ns node]   ;#address is 2
set ar1_ [$ns node]  ;# connect to MR, interface 1, address is 3
set ar2_ [$ns node] ;# connect to MR, interface 2, address is 4
set ar3_ [$ns node] ;# connect to HA, interface 1, address is 5
set ar4_ [$ns node];# connect to HA, interface 2, address is 6

 $ns duplex-link $ar3_ $ha_ 2mb 10ms DropTail
[$ar3_ get-module "Manual"] add-route-to-adj-node $ha_
[$ha_ get-module "Manual"] add-route-to-adj-node -default $ar3_

$ns duplex-link $ar4_ $ha_ 2mb 10ms DropTail
[$ar4_ get-module "Manual"] add-route-to-adj-node $ha_
[$ha_ get-module "Manual"] add-route-to-adj-node  $ar4_

$ns duplex-link $ar1_ $mr_ 2mb 10ms DropTail
[$ar1_ get-module "Manual"] add-route-to-adj-node $mr_
[$mr_ get-module "Manual"] add-route-to-adj-node  -default $ar1_

$ns duplex-link $ar2_ $mr_ 2mb 10ms DropTail
[$ar2_ get-module "Manual"] add-route-to-adj-node $mr_
[$mr_ get-module "Manual"] add-route-to-adj-node  $ar2_

$ns duplex-link $ar1_ $ar3_ 2mb 10ms DropTail
#$ns queue-limit $ar1_ $ar3_ 5000
#$ns queue-limit $ar3_ $ar1_ 5000
[$ar1_ get-module "Manual"] add-route-to-adj-node -default $ar3_
[$ar3_ get-module "Manual"] add-route-to-adj-node -default $ar1_
[$ns link $ar1_ $ar3_] dynamic
[$ns link $ar3_ $ar1_] dynamic

$ns duplex-link $ar2_ $ar4_ 2mb 10ms DropTail
[$ar2_ get-module "Manual"] add-route-to-adj-node -default $ar4_
[$ar4_ get-module "Manual"] add-route-to-adj-node -default $ar2_

set mragent [eval new Agent/MRPolicyAgent]
set haagent [eval new Agent/HAPolicyAgent]
$ns attach-agent $mr_ $mragent
$ns attach-agent $ha_ $haagent

#Create a TCP sender agent and attach it to node cn_
set tcp [new Agent/TCP]
$ns attach-agent $ha_ $tcp

#Create a TCP receiver agent and attach it to node mr_
set sink [new Agent/TCPSink]
$ns attach-agent $mr_ $sink

#Create a FTP application
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# #$ftp attach-agent $mragent
```

```
$ns connect $tcp $sink

set udp [new Agent/UDP]
$ns attach-agent $ar1_ $udp
set null [new Agent/Null]
$ns attach-agent $ar3_ $null
$ns connect $udp $null
$udp set fid_ 0

set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp
$cbr2 set type_ CBR
$cbr2 set packet_size_ 4000
$cbr2 set rate_ 2.0mb
$cbr2 set random_ true

set udp2 [new Agent/UDP]
$ns attach-agent $ar3_ $udp2
set null2 [new Agent/Null]
$ns attach-agent $ar1_ $null2
$ns connect $udp2 $null2
$udp set fid_ 0

set e2 [new Application/Traffic/Exponential]
$e2 set packetSize_ 4000
$e2 set burst_time_ 500ms
$e2 set idle_time_ 100ms
$e2 set rate_ 2.5mb
$e2 attach-agent $udp2
$ns at 1.0 "$ftp start"  ;# FTP start at 1.2 seconds
#
#$ns at 0.0 "$cbr start"
$ns at 0.0 "$cbr2 start"
$ns at 0.0 "$e2 start"
$ns at 5.0 "$mragent bad 1"

#debug 1
#$ns at 25.0 "[$ns link $ar1_ $ar3_] down"
#$ns at 25.0 "[$ns link $ar3_ $ar1_] down"
#$ns rtmodel-at 25.0 down $ar1_ $ar3_
$ns at 15.0 "finish"

#Run the simulation
$ns run
```

policy.h

```
//

#include "config.h"
#include "packet.h"
#include "list.h"
#include "agent.h"
#include "timer-handler.h"
#include <sys/types.h>
// Policy Header default settings
#ifndef ns_policy_h
#define ns_policy_h

#define POLICY_HEADER_DEFAULT_SIZE  6
#define COST_Info_SIZE 2
typedef enum{
        PREQ, PRES
} PolicyType;

////////policy timer part
class Agent;
class POLICYTimer : public TimerHandler{
 public:
       POLICYTimer(Agent *a) :TimerHandler(){a_=a;}
 protected:
       inline void expire(Event *e);
       Agent *a_;
};

class MRPolicyAgent;
class RandomTimer : public TimerHandler{
 public:
       RandomTimer(MRPolicyAgent *a) :TimerHandler(){a_=a;}
 protected:
       inline void expire(Event *e);
       MRPolicyAgent *a_;
};

class NodeEntry;  // for recording both cn and mnn
LIST_HEAD(nodeentry, NodeEntry);
class NodeEntry{
         public:
         ns_addr_t address_; // find out the definitionof ns_addr_t !!!
         inline void insert_entry (struct nodeentry *head){
         LIST_INSERT_HEAD(head, this, link);}
         NodeEntry* next_entry(void )const {return link.le_next;}
         inline void remove_entry(){LIST_REMOVE(this,link);}
         protected:
         LIST_ENTRY(NodeEntry) link;
};

class InfoEntry;  //one infoentry stores one information
LIST_HEAD(infoentry,InfoEntry);
class InfoEntry{
      public:
       int infoid_; // information id
       int infotype_;
       int interfaceid_;
       int reservedT_;
       NodeEntry* nodeentry_;
       float cost_;
       //Info info_;
       InfoEntry(int infoid, int type, int interfaceid, int reservedt, NodeEntry *n){
       infoid_=infoid;
       infotype_=type;
       interfaceid_=interfaceid;
       reservedT_=reservedt;
       nodeentry_=n;
       }
       InfoEntry(int infoid, int type, int interfaceid, int reservedt, float cost){
       infoid_=infoid;
       infotype_=type;
       interfaceid_=interfaceid;
       reservedT_=reservedt;
       cost_=cost;
       }
       inline void insert_entry (struct infoentry *head){
        LIST_INSERT_HEAD(head, this, link);
        }
        InfoEntry* next_entry(void) const {return link.le_next;}
    protected: LIST_ENTRY(InfoEntry) link;
};

// Interface State List
class InterfaceStateEntry;
LIST_HEAD(isentry,InterfaceStateEntry);

class InterfaceStateEntry {
  public:
         int id_;
         //int caddr_;
         // int mnprefix_;
         int marked_;
         float cost_;
         int selected_;
         bool available_;
         InterfaceStateEntry (int interfaceId,  float cost){
         id_=interfaceId;
         //caddr_=coa;
         //   mnprefix_=mnp;
         marked_=0; // initialized as 0: able to use
         cost_=cost;// initialized as an illegal number
         available_=true;
        //  InterfaceEntry *selected; // not selected
        }
        ~InterfaceStateEntry(){;}
        float& cost(){return cost_;}
        inline void insert_entry (struct isentry *head){
        LIST_INSERT_HEAD(head, this, link);
        }

        inline void update_entry(int interfaceId,float c){ //update these terms which are influenced by
messages
                  id_=interfaceId;
                  //caddr_=coa;
                  //mnprefix_=mnp;
                  //marked_=mark;
                  cost_=c;
       }

       //return next element in the chained list
       InterfaceStateEntry* next_entry(void) const {return link.le_next;}

       inline void remove_entry() {LIST_REMOVE(this, link);}
       inline void remove_entry(struct isentry *newhead){
              LIST_REMOVE(this, link);
              LIST_INSERT_HEAD(newhead, this, link);
       }

       protected:
                 LIST_ENTRY(InterfaceStateEntry) link;
};

#define HDR_POLICY(p)  ((struct hdr_policy*)(p)->access(hdr_policy::offset_))
struct hdr_policy{
         public:
         PolicyType ptype_;
         int seq_;
         struct infoentry infoentry_; //June6
         static int offset_;

         inline static int& offset(){return offset_;}
         inline static hdr_policy * access(Packet* p){
                  return (hdr_policy*) p->access (offset_);
         }
         inline PolicyType& ptype(){return ptype_;}
         inline int& seqno(){return seq_;}
         inline struct infoentry& infoentry(){return infoentry_;} //June6
};
#endif
```

MR Agent

```
#include <string.h>
#include <stdio.h>
#include "object.h"
#include "node.h"
#include "policy.h"
#include "ip.h"
#include "random.h"

#define TIMER_POLICY 0.2
//
static class PolicyHeaderClass : public PacketHeaderClass {
         public:
         PolicyHeaderClass() : PacketHeaderClass("PacketHeader/Policy", sizeof (hdr_policy)){
```

```
                                      bind_offset(&hdr_policy::offset_);
         }
} class_policyhdr;

class MRPolicyAgent : public Agent
{
         public:
         MRPolicyAgent();
         int s_seqno_;  //sequence number for policy information initiated by itself
         int r_seqno_;  //sequence number for policy information initiated by its partner
         int& s_seqno(){return s_seqno_;}
         int& r_seqno(){return r_seqno_;}
         struct isentry isl_head_; //interface state entry
         struct nodeentry cnentry_;  //recording all the rejected cn
         struct nodeentry mnnentry_; //recording all the rejected mnn
         struct infoentry response_; //recording the last sent response message for retransmission
         struct infoentry request_;
         Packet *lastsentpacket_;
         Packet *lastreceivedpacket_;
         //Packet *lastrequest_; //weiwei
         //Packet *lastresponse_; //weiwei
         struct infoentry lastresponse_; //recording the last sent response infomessage for retransmission
         struct infoentry lastrequest_;
         InterfaceStateEntry* selected;
         float uplinkcosttable[2];//only two interfaces..
         float downlinkcosttable[2];
         POLICYTimer ptimer_;
         RandomTimer rtimer_;
         int flag;
         int timeflag;
         int pnum;
         void select(int eventtype, int id );
         int selectinterface(int eventtype, int id );
         void timeout(int);
         void randomtimeout(int);
         InfoEntry *head(struct infoentry head){return head.lh_first;}
         void createpolicyrequest(int i, int id);
         void send_packet(Packet* p);
         void set_policy_header(Packet *p);
         void send_ack(Packet *p);
         void send_req(Packet *p);
         void recv(Packet *p, Handler *h);
         InterfaceStateEntry* update_isl(Packet*);

         InterfaceStateEntry* ishead(struct isentry head) {return head.lh_first;}
         void update_istable_enable(int id, float cost);
         void update_istable_cost(int id, float cost);
         void update_istable_disable(int id);
         Packet*  set_policy_packet(PolicyType type,struct infoentry inf, int infosize, int dst, int resend);
         protected:
         virtual int command(int argc, const char*const* argv);
} ;

static class testagentClass : public TclClass {
public:
       testagentClass() : TclClass("Agent/MRPolicyAgent") {printf("MRPolicyagentclass");};
       TclObject* create(int, const char*const*) {
       printf("create MRPolicyagent1");
       return (new MRPolicyAgent());
       };
} class_mrpolicyagent;

void POLICYTimer::expire(Event *e){ a_->timeout(TIMER_POLICY);}

void RandomTimer::expire(Event *e){ a_->randomtimeout(0);}

MRPolicyAgent::MRPolicyAgent():Agent(PT_POLICY1), ptimer_(this), rtimer_(this) {
         s_seqno_=0;
         r_seqno_=1;
         LIST_INIT(&isl_head_);
         LIST_INIT(&cnentry_);
         LIST_INIT(&mnnentry_);
         LIST_INIT(&request_);
         LIST_INIT(&response_);
         InterfaceStateEntry *i1=new InterfaceStateEntry(1,1);
         i1->insert_entry(&isl_head_);
         InterfaceStateEntry *i2=new InterfaceStateEntry(2,2);
         i2->insert_entry(&isl_head_);
         printf("create MRPolicyagent2");
         flag=1;
         pnum=0;
};

int MRPolicyAgent::command(int argc, const char*const* argv){
     if(argc == 2) {
            if(strcmp(argv[1], "start") == 0) {
              printf("***start***");
              return TCL_OK;
              }

     };
     if(argc ==3) {
     printf("the command is ");
     printf(argv[1]);
          if(strcmp(argv[1], "bad") == 0) {

                             createpolicyrequest(2, atoi(argv[2])); //just for testing, in order to force all
policy traffic flow in the link 2-5-3-1
                             update_istable_disable(atoi(argv[2]));
                             //select(2, atoi(argv[2])); for testing
                             //createpolicyrequest(2, atoi(argv[2]));
                             return TCL_OK;
                 }
           if(strcmp(argv[1], "good") == 0) {


                             update_istable_enable(atoi(argv[2]),uplinkcosttable[atoi(argv[2])]);
                             //select(1, atoi(argv[2])); for testing
                             createpolicyrequest(1, atoi(argv[2]));
                             return TCL_OK;
                 }

   };
    return (Agent::command(argc,argv));
}

void MRPolicyAgent::recv(Packet* p, Handler *h){ //mr only send info with type 1,2,or 3;and receive info with
type 4
     float cost;
     InfoEntry* entry;
     NodeEntry* i;
     hdr_policy* hdrp = hdr_policy::access(p); //get the policy header
     hdr_ip* hdrip =hdr_ip::access(p);
     if (hdrp->ptype()==PREQ){  //
        if ((hdrp->seqno())== r_seqno()){ // check the sequence number
            InfoEntry *info=head(hdrp->infoentry());
            for (;info;info=info->next_entry())
           {
               switch(info->infotype_)
               {
                case 1:                          // from MR to HA, new interface
                             cost=uplinkcosttable[info->interfaceid_-1];          //according to interface id
and coa, retrieve the cost information stored in HA,and form the response message;
                             if (cost<=0){entry= new InfoEntry(info->infoid_, 4,info->interfaceid_,0,0.0);}//
didnot find the related cost information
                             else {entry=new InfoEntry(info->infoid_,1,info->interfaceid_,0,cost);}//reserved t
is not used in this case

                             update_istable_enable(info->interfaceid_, downlinkcosttable[info->interfaceid_-
1]);
                             select(1,info->interfaceid_);
                             break;
                     case 2: // some interface is unavailable_
                             entry=new InfoEntry(info->infoid_,2,info->interfaceid_,0,0.0);

                             update_istable_disable(info->interfaceid_);
                             select(2,info->interfaceid_);
                             break;
                     case 5:
                             update_istable_cost(info->interfaceid_,info->cost_);  // from HA to MR
                             entry=new InfoEntry(info->infoid_,2,info->interfaceid_,0,0.0);
                             //InterfaceStateEntry ise=lookup_isentry(info->interfaceid_-1);

                             select(5,info->interfaceid_);
                             break;
                     case 6:  //it is information about cnlist          ;from MR to HA
                             i=info->nodeentry_;;
                             if (info->reservedT_==0 )
                             { for (;i;i=i->next_entry()){  //update cnlist on HA
                             i->insert_entry(& cnentry_);
                             }}else {for (;i;i=i->next_entry()){
                             i->remove_entry();
                             }}
                             entry=new InfoEntry(info->infoid_,2,info->interfaceid_,0,0.0);
                             //
                             break;
                case 7: //it is informtion for mnn list           ; from MR to HA
                             i=info->nodeentry_;
                             if (info->reservedT_==0 )
                             { for (;i;i=i->next_entry()){
                             i->insert_entry(& mnnentry_);
                             }}else {for (;i;i=i->next_entry()){
                             i->remove_entry();
                             }}
                             entry=new InfoEntry(info->infoid_,2,info->interfaceid_,0,0.0);
                             break;

                  }
                 entry->insert_entry(&response_);
          }
                     int addr=hdrip->saddr();
                     lastsentpacket_=set_policy_packet(PRES, response_, sizeof(response_), addr,0);
                     lastresponse_=response_;
                     LIST_INIT(&response_);
                     send_packet(lastsentpacket_);
        }else {   Packet *p=set_policy_packet(PRES, lastresponse_,sizeof(lastresponse_),1,1);
                 send_packet(p);
                  //for repeated request,resend the repsonse message
                 }
     }else if (hdrp->ptype()==PRES){
                     printf("MR receives a response policy message;\n");
                     printf("%d\n",hdrp->seqno());
                     printf("%d\n",s_seqno());
                     if ((hdrp->seqno()== s_seqno())&&(flag==0)){ // check the sequence number
                     InfoEntry* info=head(hdrp->infoentry());
                     for (;info;info=info->next_entry())
                     {
                     switch(info->infotype_)
                     {
                             case 1://information about the given interface. handle this in the same way as
case 4 of request message
                             update_istable_cost(info->interfaceid_,info->cost_);
                             break;
                             case 2:
                     //do nothing,
                             break;
                             case 4: // error message, do nothing.here
                             break;
                     }
                     }
                     if (ptimer_.status() == TIMER_PENDING){
                             ptimer_.cancel();// by weiwei
```

```
                              }
                              flag=1;// indicates the respected response is received and next request is allowed
to sent
                   pnum=pnum+1;
         if (pnum==500) exit(0);
         float waittime=Random::uniform(0,1);
         rtimer_.sched(waittime);
     }
      //for repeated reponse, just ignore it
     };
     Packet::free(p);
};
void MRPolicyAgent::createpolicyrequest(int i, int id){

         InfoEntry *entry=new InfoEntry(1,i,id,0,0.0);
         entry->insert_entry(&request_);
         int addr=1;//HA's address is 1;MR is 2;
         printf("the size of info is %d\n", sizeof(request_));
         lastsentpacket_=set_policy_packet(PREQ, request_, sizeof(request_), addr,0);
         while(1){
             if (flag==1) {send_packet(lastsentpacket_);
                           ptimer_.sched(TIMER_POLICY);
                              lastrequest_=request_;
                              LIST_INIT(&request_);
                              flag=0;//indicate one request is sent out and response is respected
                              break;
                              }
         }
};
void MRPolicyAgent::update_istable_enable(int id, float cost){
         InterfaceStateEntry *in=ishead(isl_head_);
         for (;in;in=in->next_entry())
         {  if (in->id_ ==id){ in->cost_ =cost; in->available_ =true;}
         }
}

void MRPolicyAgent::update_istable_cost(int id, float cost){
         InterfaceStateEntry *in=ishead(isl_head_);
         for (;in;in=in->next_entry())
         {  if (in->id_ ==id){ in->cost() =cost;}
         }
}

void MRPolicyAgent::update_istable_disable(int id){
         InterfaceStateEntry *in=ishead(isl_head_);
         for (;in;in=in->next_entry())
         {  if (in->id_ ==id){ in->available_ =false;}
         }
}

int  MRPolicyAgent::selectinterface(int eventtype,int id){
      InterfaceStateEntry * in=ishead(isl_head_);
      InterfaceStateEntry * ise=ishead(isl_head_);;

      for(;ise;ise=ise->next_entry()){
      if (ise->id_==id) break;}
      switch(eventtype)
      {
        case 1: if (in->next_entry()==NULL) //if it is the only available_ interface, select it
          {selected=ise;
           }
         else if( ise->cost_< (selected)->cost_)
                   {selected=ise;}
         break;
        case 2:
        case 3://eventinterface(selected) becomes down or quality turns bad,
          //reselect from all available_ and unmarked interfaces. if none of them has good quality,use the
first one in the table;
         selected=new InterfaceStateEntry(0,0);
         for (;in;in=in->next_entry())
         {if (in->available_==true) {selected=in;}};//first find one that is avaible
         for (;in;in=in->next_entry()){
        if ((in->cost_<selected->cost_)&&(in->available_==true)) { selected=in;}}//then select the cheap one
         break;
        case 4:// eventinterface's quality turns good, if it is cheaper than the selected one, use the new one
         if (ise->cost_< (selected)->cost_)
         {selected=ise;}
         break;
        case 5://cost changes,
         if (ise->id_!=(selected)->id_)
                  {if (ise->cost_<(selected)->cost_)
                  {selected=ise;}}
         else {
                           selected=in;
                           for (;in;in=in->next_entry()){
                       if ((in->cost_< selected->cost_)&&(selected->available_==true)) {selected=in;}}
                  }
         break;
         }
      return selected->id_;
};

void MRPolicyAgent::send_packet(Packet *p){  //refers to mipv6.cc in mobiwan
                  send(p,0);
                  return;
};

// Policy Packet= common header+ip header +policy header
Packet* MRPolicyAgent:: set_policy_packet(PolicyType type,struct infoentry inf, int infosize, int dst, int
resend) { // the dynamic size of information is also a parameter, and dst is used for ip header
         Packet* p=allocpkt();
         struct hdr_cmn *hdrc=HDR_CMN(p);
         struct hdr_ip *hip=HDR_IP(p);
         hip->daddr()=1;
         hip->dport()=0;
         hip->saddr()=2;
         hip->sport()=0;
         struct hdr_policy* hdrp =hdr_policy::access(p);
         hdrp->ptype() =type;
         hdrp->infoentry()=inf; //June6
         if(resend==0){
         if (type==PREQ) {
                  hdrp->seqno()=++s_seqno_;
         }else{ hdrp->seqno()=r_seqno_; r_seqno()++; }}
         if (resend==1){
         if (type==PREQ) {
                  hdrp->seqno()=s_seqno_;
         }else{ hdrp->seqno()=r_seqno_-1;  }
         }
         printf("the s-seqno is %d\n",s_seqno_);
         hdrc->size()= POLICY_HEADER_DEFAULT_SIZE+infosize+IP_HDR_LEN;

         return (p);
};

void MRPolicyAgent::timeout(int t){ //resending requset needs timer; resending response does not
         //send_packet(lastrequest_);
         //for testing
         Packet *p=set_policy_packet(PREQ, lastrequest_,sizeof(lastrequest_),1,1);
         //for testing end
         send_packet(p);
       ptimer_.resched(TIMER_POLICY);//
       flag=0;
       printf("send previous request");
};

void MRPolicyAgent::randomtimeout(int){
 Tcl& tcl=Tcl::instance();
if (timeflag==0){ tcl.evalc("$mragent good 1");
          timeflag=1;
}
else {tcl.evalc("$mragent bad 1");
```

```
        timeflag=0;}
}

void MRPolicyAgent::select(int eventtype,int id){
         int s=selectinterface(eventtype,id)  ;
         Tcl& tcl=Tcl::instance();
         if (s==1){
         tcl.evalc(" [$mr_ get-module Manual] add-route-to-adj-node -default $ar1_");};

         if (s==2){
         tcl.evalc(" [$mr_ get-module Manual] add-route-to-adj-node -default $ar2_");};

         if (s==0){printf("no interface is available at this moment");
                  } //it means no interface is available at this moment, and i should do something to disable
these real routes
};
int hdr_policy::offset_;
```

HA Agent

```
#include <string.h>
#include <stdio.h>
#include "object.h"
#include "node.h"
#include "policy.h"
#include "ip.h"
#define TIMER_POLICY 0.2

static class PolicyHeaderClass : public PacketHeaderClass {
public:
       PolicyHeaderClass() : PacketHeaderClass("PacketHeader/Policy", sizeof (hdr_policy)){
                            bind_offset(&hdr_policy::offset_);
       }
} class_policyhdr;

class HAPolicyAgent : public Agent
{
         public:
         HAPolicyAgent();
         int s_seqno_;  //sequence number for policy information initiated by itself
         int r_seqno_;  //sequence number for policy information initiated by its partner
         int& s_seqno(){return s_seqno_;}
         int& r_seqno(){return r_seqno_;}
         struct isentry isl_head_; //interface state entry
         struct nodeentry cnentry_;  //recording all the rejected cn
         struct nodeentry mnnentry_; //recording all the rejected mnn
         struct infoentry response_; //recording the last sent response message for retransmission
         struct infoentry request_;
         Packet *lastsentpacket_;
         Packet *lastreceivedpacket_;
         struct infoentry lastresponse_; //recording the last sent response infomessage for retransmission
         struct infoentry lastrequest_;
         InterfaceStateEntry* selected;
         float uplinkcosttable[2];//only two interfaces..
         float downlinkcosttable[2];
         POLICYTimer ptimer_;
         int flag;

         void select(int eventtype, int id );
         int selectinterface(int eventtype, int id );
         void timeout(int);
         InfoEntry *head(struct infoentry head){return head.lh_first;}
         void createpolicyrequest(int i, int id, float cost);
         void send_packet(Packet *p);
         void set_policy_header(Packet *p);
         void send_ack(Packet *p);
         void send_req(Packet *p);
         void recv(Packet *p, Handler *h);
         InterfaceStateEntry* update_isl(Packet*);
         InterfaceStateEntry* ishead(struct isentry head) {return head.lh_first;}
         void update_istable_enable(int id, float cost);
         void update_istable_cost(int id, float cost);
         void update_istable_disable(int id);
         Packet*  set_policy_packet(PolicyType type,struct infoentry inf, int infosize, int dst, int resend);
         protected:
         virtual int command(int argc, const char*const* argv);
} ;

static class haagentClass : public TclClass {
public:
       haagentClass() : TclClass("Agent/HAPolicyAgent") {printf("HAPolicyagentclass");};
       TclObject* create(int, const char*const*) {
       printf("create HAPolicyagent1");
       return (new HAPolicyAgent());
       };
} class_hapolicyagent;

HAPolicyAgent::HAPolicyAgent():Agent(PT_POLICY2), ptimer_(this) {
         s_seqno_=0;
         r_seqno_=1;
         LIST_INIT(&isl_head_);
         LIST_INIT(&cnentry_);
         LIST_INIT(&mnnentry_);
         LIST_INIT(&response_);
        LIST_INIT(&request_);
         uplinkcosttable[0]=1.0;
         uplinkcosttable[1]=2.0;
         downlinkcosttable[0]=1.0;
         downlinkcosttable[1]=2.0;
         InterfaceStateEntry *i1=new InterfaceStateEntry(1,1);
         i1->insert_entry(&isl_head_);
         InterfaceStateEntry *i2=new InterfaceStateEntry(2,2);
         i2->insert_entry(&isl_head_);
         //lastreceivedpacket_=set_policy_packet(PRES, response_, sizeof(response_), 0);
         //lastsentpacket_=set_policy_packet(PREQ, request_, sizeof(response_), 0);
         flag=1;
         printf("create HAPolicyagent2");
};

int HAPolicyAgent::command(int argc, const char*const* argv){
     if(argc == 2) {
            if(strcmp(argv[1], "start") == 0) { //just for testing, could be delete later
                  printf("***start***");
                  return TCL_OK;
             }
     }
     if (argc == 4) {
            if(strcmp(argv[1], "uplinkcostchange") == 0) {  //for uplinkcost change event, one message should
be created and sent to MR, for downlinkcost change event, HA just simply changes its interfacestate table and
selects the suitable interface
                  createpolicyrequest(5, atoi(argv[2]), atoi(argv[3]));
                  uplinkcosttable[atoi(argv[2])]=atoi(argv[3]);
                  return TCL_OK;
             }
             if (strcmp(argv[1], "downlinkcostchange") == 0) {
                  update_istable_cost(atoi(argv[2]),atoi(argv[3]));
                  downlinkcosttable[atoi(argv[2])]=atoi(argv[3]);
                  select(5, atoi(argv[2]));
                  return TCL_OK;
             }
     }
     return (Agent::command(argc,argv));
};

void HAPolicyAgent::createpolicyrequest(int i, int id, float cost){//"i" is the type;

         InfoEntry *entry=new InfoEntry(1,i,id,0,cost);
         entry->insert_entry(&request_);
         int addr=1;//HA's address is 1;MR is 2;
         lastsentpacket_=set_policy_packet(PREQ, request_, sizeof(request_), addr,0);
         while(1){
             if (flag==1) {send_packet(lastsentpacket_);
```

```
                         ptimer_.sched(TIMER_POLICY);
                            lastrequest_=request_;
                            LIST_INIT(&request_);
                            flag=0;//indicate one request is sent out and response is respected
                            break;
                            }
         }
};

void HAPolicyAgent::recv(Packet* p, Handler *h){ //mr only send info with type 1,2,or 3;and receive info with
type 4
     printf("HA received one policy message\n");
     Tcl& tcl = Tcl::instance();
     float cost;
     InfoEntry* entry;
     NodeEntry* i;
     hdr_policy* hdrp = hdr_policy::access(p); //get the policy header
     hdr_ip* hdrip =hdr_ip::access(p);
     if ((hdrp->ptype()) == PREQ ) {  //
         printf("HA received one request policy message\n");
         printf("%d\n",hdrp->seqno());
         printf("%d\n",r_seqno());
         if ((hdrp->seqno())== r_seqno()){ // check the sequence number
         printf("the seq no is correct");
         struct infoentry weiwei=hdrp->infoentry();
           InfoEntry *info=head(hdrp->infoentry());
           for (;info;info=info->next_entry())
          {
              switch(info->infotype_)
              {
               case 1:                            // from HA to HA, new interface
                          cost=uplinkcosttable[info->interfaceid_-1];          //according to interface id
and coa, retrieve the cost information stored in HA,and form the response message;
                          if (cost<=0){entry= new InfoEntry(info->infoid_, 4,info->interfaceid_,0,0.0);}//
didnot find the related cost information
                          else {entry=new InfoEntry(info->infoid_,1,info->interfaceid_,0,cost);}//reserved t
is not used in this case
                          printf("HA recieves a policy message which indicates a good interface\n");
                          update_istable_enable(info->interfaceid_, downlinkcosttable[info->interfaceid_-
1]);
                          //select(1,info->interfaceid_); for testing
                          break;

                  case 2: // some interface is unavailable_
                          entry=new InfoEntry(info->infoid_,2,info->interfaceid_,0,0.0);
                          update_istable_disable(info->interfaceid_);
                          printf("HA recieves a policy message which indicates a bad interface\n");
                          //select(2,info->interfaceid_);  for testing
                          break;
                  case 5:
                          update_istable_cost(info->interfaceid_,info->cost_);  // from HA to HA
                          entry=new InfoEntry(info->infoid_,2,info->interfaceid_,0,0.0);
                          select(5,info->interfaceid_);
                          break;
                  case 6:  //it is information about cnlist          ;from HA to HA
                          i=info->nodeentry_;;
                          if (info->reservedT_ ==0 )
                          { for (;i;i=i->next_entry()){  //update cnlist on HA
                          i->insert_entry(& cnentry_);
                          }}else {for (;i;i=i->next_entry()){
                          i->remove_entry();
                          }}
                          entry=new InfoEntry(info->infoid_,2,info->interfaceid_,0,0.0);
                          //
                          break;
               case 7: //it is informtion for mnn list           ; from HA to HA
                          i=info->nodeentry_;
                          if (info->reservedT_==0 )
                          { for (;i;i=i->next_entry()){
                          i->insert_entry(& mnnentry_);
                          }}else {for (;i;i=i->next_entry()){
                            i->remove_entry();
                            }}
                            entry=new InfoEntry(info->infoid_,2,info->interfaceid_,0,0.0);
                            break;
                   }
                        entry->insert_entry(&response_);
          }
                   int addr=hdrip->saddr();
                   lastreceivedpacket_=set_policy_packet(PRES, response_, sizeof(response_), addr,0);
                   lastresponse_=response_;
                   LIST_INIT(&response_);
                   send_packet(lastreceivedpacket_);
        }else if((hdrp->seqno())== (r_seqno()-1)){          //for repeated request,resend the repsonse message

              Packet *p=set_policy_packet(PRES,lastresponse_,sizeof(lastresponse_),2,1);
         send_packet(p); //send the response or last response message
         }
         }else if (hdrp->ptype()==PRES){
                   if ((hdrp->seqno()== s_seqno())&&(flag==0)){ // check the sequence number
                   InfoEntry* info=head(hdrp->infoentry());
                   for (;info;info=info->next_entry())
                   {
                   switch(info->infotype_)
                   {
                            case 1://information about the given interface. handle this in the same way as
case 4 of request message
                            update_istable_cost(info->interfaceid_,info->cost_);
                            break;
                            case 2:
                            //do nothing,
                            break;
                            case 4: // error message, do nothing.here
                            break;
                   }
                   }
                            if (ptimer_.status() == TIMER_PENDING){

                   ptimer_.cancel();}
                   flag=1;
                   };
                            }
         //for repeated reponse, just ignore it
              Packet::free(p);
};

void HAPolicyAgent::update_istable_enable(int id, float cost){
         InterfaceStateEntry *in=ishead(isl_head_);
         for (;in;in=in->next_entry())
         {  if (in->id_ ==id) { in->cost_ =cost; in->available_ =true;}
         }
         }

void HAPolicyAgent::update_istable_cost(int id, float cost){
         InterfaceStateEntry *in=ishead(isl_head_);
         for (;in;in=in->next_entry())
         {  if (in->id_ ==id){ in->cost() =cost;}
         }
         }

void HAPolicyAgent::update_istable_disable(int id){
         InterfaceStateEntry *in=ishead(isl_head_);
         for (;in;in=in->next_entry())
         {  if (in->id_ ==id){ in->available_ =false;}
         }
         }

int  HAPolicyAgent::selectinterface(int eventtype,int id){
      InterfaceStateEntry * in=ishead(isl_head_);
      InterfaceStateEntry * ise;
      ise=in;
      for(;ise;ise=ise->next_entry()){
```

```
    if (ise->id_==id) break;}
    switch(eventtype)
    {
      case 1: if (in->next_entry()==NULL) //if it is the only available_ interface, select it
       {selected=ise;
       }
       else if( ise->cost_< (selected)->cost_)
             {selected=ise;}
       break;
      case 2:
      case 3://eventinterface(selected) becomes down or quality turns bad,
       //reselect from all available_ and unmarked interfaces. if none of them has good quality,use the
first one in the table;
       selected=in;
       for (;in;in=in->next_entry()){
      if ((in->cost_<selected->cost_)&&(in->available_==true)) { selected=in;}}
       break;

      case 4:// eventinterface's quality turns good, if it is cheaper than the selected one, use the new one
       if (ise->cost_< (selected)->cost_)
       {selected=ise;}
       break;
      case 5://cost changes,
       if (ise->id_!=(selected)->id_)
             {if (ise->cost_<(selected)->cost_)
             {selected=ise;}}
       else {
                    selected=in;
                    for (;in;in=in->next_entry()){
                 if ((in->cost_< selected->cost_)&&(selected->available_==true)) {selected=in;}}
             }
       break;

       }
    return selected->id_;
};

void HAPolicyAgent::send_packet(Packet* p){  //refers to mipv6.cc in mobiwan
     send (p,0);
};

// Policy Packet= common header+ip header +policy header
Packet* HAPolicyAgent:: set_policy_packet(PolicyType type,struct infoentry inf, int infosize, int dst, int
resend) { // the dynamic size of information is also a parameter, and dst is used for ip header
    Packet* p=allocpkt();
    hdr_cmn *hdrc=HDR_CMN(p);
    hdr_ip *iph=HDR_IP(p);
    hdr_policy* hdrp =hdr_policy::access(p);
    hdrp->ptype() =type;
    hdrp->infoentry()=inf;
    if (resend==0){
    if (type==PREQ) {
       hdrp->seqno()=++s_seqno_;
    }else{ hdrp->seqno()=r_seqno_; r_seqno()++; }}
    if (resend==1){
       if (type==PREQ) {
             hdrp->seqno()=s_seqno_;
       }else{ hdrp->seqno()=r_seqno_-1;  }
       }

    hdrc->size()= POLICY_HEADER_DEFAULT_SIZE+infosize;
    iph->daddr()=2;
    iph->dport()=0;
    iph->saddr()=1;
    iph->sport()=0;

    return (p);
};

void HAPolicyAgent::timeout(int t){ //resending requset needs timer; resending response does not
       Packet *p=set_policy_packet(PREQ,lastrequest_,sizeof(lastrequest_),2,1);
```

```
         send_packet(p);
         //send_packet(lastsentpacket_);
      ptimer_.resched(TIMER_POLICY);//
};

void HAPolicyAgent::select(int eventtype,int id){
         int s=selectinterface(eventtype,id)  ;
         Tcl& tcl=Tcl::instance();
         if (s==1){
         tcl.evalc(" [$ha_ get-module Manual] add-route-to-adj-node -default $ar3_");};
         if (s==2){
         tcl.evalc(" [$ha_ get-module Manual] add-route-to-adj-node -default $ar4_");};
};
```