A Policy Management Framework for Flow Distribution on Multihomed End Nodes

Koshiro Mitsuya
Keio University, Japan
mitsuya@sfc.wide.ad.jp

Romain Kuntz
Louis Pasteur University, France
kuntz@lsiit.u-strasbg.fr

Shinta Sugimoto
Keio University, Japan
shinta@sfc.wide.ad.jp

Ryuji Wakikawa
Keio University, Japan
ryuji@sfc.wide.ad.jp

Jun Murai
Keio University, Japan
jun@wide.ad.jp

ABSTRACT

A multihomed node has several paths with its correspondent, maintained by several multihoming protocols. The decision to route a packet over a specific path relies on filter rules, which result from the comparison between the path's characteristics and the user policy. Multihoming protocols or their implementations provide various user interfaces to configure the filter rules. However, there is currently no method to describe user policy in terms of cost, bandwidth, delay and other network characteristics, and to compare this policy with the path characteristics. We thus propose in this paper a new framework for policy management for flow distribution, which offers a user interface to define policies and generates filter rules for each multihoming protocol. We first sort out the requirements for users in the multihomed environment. By reviewing some of the most important multihoming protocols and implementations, we show that they do not match all those requirements. We then propose a new policy management framework that fits those very requirements.

1. INTRODUCTION

Thanks to the recent outstanding development of various wireless access technologies, a node is likely to be equipped with multiple network interfaces. This is especially true for mobile nodes, which often embeds multiple wireless technologies such as GPRS, 3G, IEEE802.11 and Bluetooth. In order to improve the user's experience, recent research has focused on how such a multihomed node could switch between or simultaneously use these accesses. Hence, many enhancements to TCP/IP have been proposed to address this issue, in both mobile and non-mobile environments: Multiple Care-of Addresses (MCoA) registration [17] for Mobile IPv6 (MIPv6) [4], the Host Identity Protocol (HIP) [9], Site Multihoming by IPv6 Intermediation (SHIM6) [10], and the Stream Control Transmission Protocol (SCTP) [14].

In the near future, we can imagine that a node will run several multihoming protocols, as each of them has its own benefits and none of them can cover all situations. For example, a fixed node may run the MCoA protocol in order to communicate with a multihomed Mobile Node, as well as the SCTP protocol in order to use efficiently its multiple addresses to communicate with other peers in the Internet.

The user or application that would like to perform, for example, load sharing among the available paths has to configure the flow distribution mechanism available on the node according to its desires. A packet to be sent to a peer is then processed by this flow distribution mechanism, and according to its decision, transmitted via one of the multiple available paths.

However, in order to perform flow distribution, current multihoming protocols or their implementations usually provide a filtering mechanism that is dependent from the protocol specification. If the user wants to specify to which path a flow should be routed, this usually translates in a protocol-specific rule processed by the filtering mechanism. Multihomed nodes are lacking a method for policy management on top of the multihoming protocols to describe user policy in terms of desired cost, bandwidth, delay, or jitter, to be independent from the protocol used on that node. This policy would then be compared with the available path characteristics, and the filtering mechanism configured according to the result of that comparison.

In this paper, we thus propose a new policy management framework for flow distribution, which offers a user interface to define policies and generates filter rules for each multihoming protocol. we first explain the assumed multihomed environment and then sort out the requirements for applications or users that would like to benefit from this environment. By reviewing some of the most important multihoming protocols and implementations, we show that a gap exists with those requirements. We thus propose the policy management framework that would fill the existing gap between specifications, implementations, and the previously defined requirements.

2. FUTURE MULTIHOMED ENVIRONMENT

In this section, we explain the multihomed environment that we envision to deploy in the near future, and define roles of a flow distribution mechanism in such an environment.

2.1 Terminology

We first define the following terms used in this paper:

Requester A requester is a local or remote user or application that solicits the node's flow distribution mechanism with a set of policies.

MobiArch'07, August 27–31, 2007, Kyoto, Japan.

Flow A flow is an unidirectional traffic stream defined by a set of selectors (for example, the IP source and destination address, the source and destination port, etc.).

Policy A policy associates flows with the desired path's characteristics. This represents the user's will for each flow.

Policy Rule A Policy Rule associates policies with a set of conditions. This represents what policies must be applied when the conditions are fulfilled.

Filter Rule A filter rule associates a flow to an action (the output path to choose, drop the flow, etc.). This is typically the output from the policy management mechanism once the user's policies have been processed.

2.2 Flow distribution on multihomed end nodes

The overview of the envisioned multihomed environment is shown in Fig. 1. Various multihoming protocols (further described in Sec. 4) have been or are being standardized, each of them addressing different multihoming scenario and targeting specific goals: for example, MCoA provides multiple path to a Mobile IPv6 node for ubiquity purposes while SHIM6 tends to solve site-multihoming especially for end-system fixed nodes. The figure represents a node running several of those multihoming protocols and establishing multiple paths with several correspondents.

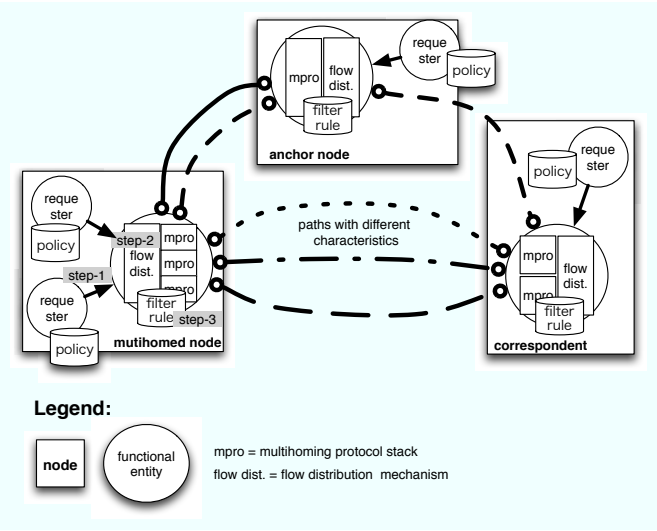

Figure 1: Future multihomed environment

A node can get many benefits from being multihomed: fault tolerance, ubiquitous access and load-sharing are usually the advantages pointed up by multihoming solutions. The last one is of most interest in this paper as this can be achieved through flow distribution. Each path maintained on a node by its multihoming protocol(s) would have different characteristics in terms of cost, bandwidth, delay and jitter. The requester will try to distribute its flows among these available paths by submitting a set of policies (Fig. 1 step-1) to the policy management framework. This one processes it (step-2), resulting in the filter rules (step-3). Those filter rules are then used as an input to the requester's OS-specific filtering framework (socket API, packet filtering framework, etc.) that will take care of the packet routing through the correct output interface. Filter rules may also be exchanged between peers, in order for each node involved in the communication to tell its preferences to each of its correspondents.

In essence, flow distribution is achieved by comparing the requester's policy and the characteristics of each available paths: flow distribution is a succession of actions that implements such a Quality of Service (QoS). The key notions of a QoS mechanism are *flows*, *service contracts*, and *flow management* [1]. *Flows* characterize the production, transmission and eventual consumption of a flow associated QoS. *Service contracts* are binding agreements of QoS levels between users and providers of the QoS. *Flow management* provides the monitoring and preservation of the contracted QoS levels.

A flow distribution mechanism would comply with these notions. Thus, our approach to implement a policy management framework for flow distribution on multihomed end nodes is detailed as follows:

- Similarly use the concept of flow because it is widely accepted among multihoming protocols moreover Internet Traffic Engineering. This corresponds to the *Flows*.
- Provide a user interface to submit user's policy into the flow distribution system. As further detailed in Sec. 4, none of current multihoming protocols provide that. This corresponds to the *service contracts*.
- Make the system generating filter rules that meet user's policy, thus requesters do not need to mind about the kind of flow management being used on the node. This also corresponds to the *service contracts*.
- Use the filtering frameworks provided by implementations of multihoming protocols. These protocols are lying on a number of layers and planes thus it is hard to provide a single generic filtering framework. This corresponds to the *flow management*.

Requirements for policy management that results from this analysis are exposed in the next section.

3. REQUIREMENTS FOR A POLICY MANAGEMENT MECHANISM

We split the approach explained in Sec. 2 into the following functional requirements that such a policy management mechanism (hereafter called "the system") would have to fulfill:

R1 Policy description: the requester must be able to describe its policy using a language or an interface defined by the system. Such an interface must match the following requirements:

- **R1.1** The requester must be able to describe each flow or group of flows and associate them with its desired path characteristics.
- **R1.2** Description of the policy must be done in a multihoming protocol-independent manner: it must not depend on some of the multihoming protocol's identifier or characteristics.

R2 Multiple requesters: several concurrent requesters must be able to submit their policies to the same system. As each requester may be local or remote to the host on which the system operates, a transport solution to exchange the policies must be available.

R3 Policy resolution: each of the requester's policy must be resolved by the system into at least one filter rule, or if need arises into an error reported to the requester. The system may

also provide a default path for policies that do not match any of the available path. It may also be able to resolve conflicts when contradictory policies arise, and remove duplicated policies.

R4 Filter rule description: the filter rule produced by the system must include all the necessary information to be used as an input to the OS-specific frameworks (socket API, packet filtering framework, etc.) that will take care of the packet routing through the correct output interface.

R5 Filter rules transport: as some of the filter rules may target a peer host, a transport solution to exchange filter rules must be available in the system.

R6 Multiple filter rules processing: the system must be able process multiple filter rules, and if need arises, to report errors to the source of the filter rule.

R7 Security: transport of policies and filter rules as mandated in requirements 2 and 5 must be done in a secure manner, and ensure host authentication, policy and filter rules encryption and data integrity.

4. REVIEW OF MULTIHOMING PROTOCOLS AND THEIR IMPLEMENTATIONS

We review in this section some of the most important multihoming protocols and their implementations. We then confront them to the previously defined requirements.

4.1 Mobile IPv6 based

Protocol Overview

Mobile IPv6 [4] and its extension NEMO Basic Support (NEMO BS [2]) have been standardized by the IETF to allow a node or a network to remain reachable at the same IPv6 address and prefix while moving around in the Internet. In order to provide Multihoming features to the IPv6 mobility support, the Multiple Care-of Addresses Registration (MCoA [17]) protocol is currently being discussed by the IETF. It allows the mobile node to maintain multiple concurrent paths with its correspondents, thus ensuring durable and wide-area access to the Internet. A node using MCoA is always reachable at a unique permanent IPv6 address (the Home Address, used as an identifier) while maintaining several temporary addresses (the Care-of Addresses, used as locators), representing the real location of the node in the Internet. As the locators can change over the time, each path is identified with a Binding Unique Identification (BID) number.

Several proposals at the IETF MONAMI6 Working Group define how filter rules can be exchanged between two nodes (at least one being mobile and multihomed): some are very specific to Mobile IPv6 as they define the mechanism inside the Mobile IPv6 signaling [13], some others try to be independent from the Mobile IPv6 specification, while using some of the MCoA features ([7] and [8]). These proposals address all of the previously defined requirements except R1.2, but as they are still a work in progress, they are subject to change.

Implementation Overview

SHISA, a MIPv6/NEMO Basic Support implementation for BSD [12], supports MCoA for NEMO Basic Support. In order to maintain multiple paths on a mobile node, each path is represented by an operating-system specific $mtun$ tunnel interface. Each $mtun$ interface is bound to a physical interface. By default, all traffic is sent to one of the $mtun$ interfaces (for example, $mtun0$), but the user can specify some filter rules thanks to one of the Operating System's packet filtering mechanism (e.g IP Filter[1] or PF[2]). For example, the following PF rules send all the traffic from the 2001:db8:0:f011:: prefix (**from 2001:db8:0:f011::/64 to any**) via the interface bound to $mtun0$ (**route-to mtun0**), except SSH traffic (**port 23**) that is sent through the interface bound to $mtun1$ (**route-to mtun1**):

```
pass out route-to mtun0 inet6
     from 2001:db8:0:f011::/64 to any

pass out route-to mtun1 inet6
     from 2001:db8:0:f011::/64 to any port 23
```

NEPL[3], a NEMO Basic Support implementation for Linux, has been extended to support MCoA[4]. Policy routing is achieved with the packet marking capability of the netfilter framework[5]. In order to send a flow via a specific path, packets are marked with the BID of this path. No output interface names are ever used to define routing policies, and another part of the system maintains the relationship between the mark and the path (the actual output interface) to be used to send the matched packets.

For example, the following rule marks with the integer value 10 (**-j MARK –set-mark 10**) all the SSH traffic (**-p 23**) whose IPv6 source address matches the prefix 2001:db8:0:f011::/64 (**–source 2001:db8:0:f011::1**).

```
ip6tables -A PREROUTING -t mangle
          -p 23
          --source 2001:db8:0:f011::/64
          -j MARK --set-mark 10
```

All packets matching this rule will then be routed via the path bound to the BID 10. Further details about the MCoA for NEPL implementation design can be read in [6].

With both of these implementations, the user cannot define policies and must deal directly with the OS-specific packet filtering tool to configure the system's flow distribution. Furthermore, the rules installed by the user are very protocol-dependent, as the output interface or the BID has to be specified.

4.2 SHIM6 based

Protocol Overview

SHIM6 [10] is a protocol which aims to provide multihoming support for IPv6, which is based on a concept of Identifier-Locator split. The primary goal of SHIM6 is to provide locator agility to upper layer protocols without imposing any load on the global routing infrastructure. The upper Layer Identifier (ULID) is an identifier which is presented to upper layer protocols and thus used by applications to identify a communication endpoint. Inside the SHIM6 IP sub-layer, ULID is mapped to one or more locators that are globally routable unicast addresses. The mapping is stored in a data structure called SHIM6 context which can be established between end-hosts by a 4-way handshake, and a context can be uniquely identified by ULID pair. Once a context is established, a given flow can be multiplexed or demultiplexed by the SHIM6 IP sub-layer.

How applications can leverage the multihoming support provided by SHIM6 is outside the scope of the protocol specification. One

[1]http://coombs.anu.edu.au/~avalon/
[2]http://www.openbsd.org/faq/pf/
[3]http://www.mobile-ipv6.org
[4]http://software.nautilus6.org/MCoA/
[5]http://www.netfilter.org

viable approach would be to introduce a kind of policy database inside the SHIM6 IP sub-layer, which would store flow information and its associated action. The policy database would be looked up during the outbound packet processing to check if the IP packet needs to be multiplexed by SHIM6. Additionally, an application may set its preferences on locators for both local and remote node by using the socket options defined in [5].

Thanks to the SHIM6 API, flow distribution can be performed per socket, which allows specification of both local and remote preferred locators on a SHIM6 node. This addresses the requirements R1.1 and R5.

Implementation Overview

The UCL SHIM6 implementation[6] for the GNU/Linux Operating System is still a work in progress and does not offer yet any mechanism to control the shim. However, it plans to support the SHIM6 API [5] in the future.

Another implementation [11] for the GNU/Linux OS uses a different approach using the queue handler for IPv6 that allows packets to be queued in userspace for further processing with the Linux Netfilter framework. Netfilter rules takes care of adding the SHIM6 extension header and swapping the source and destination addresses with their ULID.

Those implementations do not support the SHIM6 API yet, and one relies on the OS-specific packet filtering tool and the SHIM6 ULID to configure the flow distribution.

4.3 HIP based

Protocol Overview

Host Identity Protocol (HIP) [9] is a protocol designed based on a concept of ID/Locator separation which aims to improve security, mobility and multihoming capability of end hosts in the Internet. In the HIP architecture, a new namespace is created for identifying an endpoint of a communication. A Host Identifier (HI) is is used by the upper layer protocols to identify communication endpoint, which is a public key of the host. An IP address simply represents a location of the host and is used to route IP packets from the source to the destination. In the HIP architecture, two communicating peers establish a HIP context by running 4-way message exchange. Once a HIP context is established, data packets exchanged by the peers are protected by Encapsulation Payload Protocol (ESP) based on the shared secret. Each end can authenticate its peer by using public key cryptography. Thanks to the IPsec, HIP provides protection against Denial-of-Service attacks and supports data integrity and optionally data confidentiality. In HIP, an association between a HI and one or more locators can be flexible, i.e., communication endpoints remain available regardless of the changes of locators. In this way, HIP provides capability of mobility and multihoming to end hosts.

There is no framework defined for HIP to exchange policy and filter rules between the communicating peers. A multihoming shim API [5], however, defines a set of APIs for control of locator management and reachability protocol.

Implementation Overview

HIPL[7] is a HIP implementation for Linux. HIPL provides an API to configure the initial locator manually, but it does not yet provide an interface to track locator updates: applications cannot know when a locator changes, is added or removed. It is thus impossible to perform flow distribution with the current state of this implementation.

4.4 SCTP based

Protocol Overview

The Stream Control Transmission Protocol (SCTP [14]) is a new transport layer protocol. The SCTP common header includes a Verification Tag field (whose value is chosen by each endpoint during the association phase), and a 32 bit checksum field. Those values are used to verify that the SCTP packet belongs to the current association and is not an old or stale packet from a previous one. Thanks to this association, an SCTP multihomed host can be reached via more than one IP address.

In order to associate an SCTP endpoint with multiple addresses, $sctp\_bindx()$ is introduced in [15]. The addresses associated with a socket are the eligible transport addresses for the endpoint to send and receive data. The endpoint also presents those addresses to its peers during the association process.

Thanks to the use of a socket API, SCTP fulfills the requirement R1.1.

Implementation Overview

The LKSCTP (Linux Kernel SCTP) Project[8] is an implementation of SCTP and its associated protocols. It is now part of the mainline Linux kernel code. It supports the sockets API extensions [15] designed for applications that would like to benefit from the multiple paths offered by SCTP. A library function (not defined in this API) may also help the application to pick a source and destination address for the communication. Such a library function could thus step in the path selection on an SCTP node.

4.5 Summary

All multihoming protocols surveyed in this section are based on a identifier/locator separation concept with a shim that takes care of multiplexing and demultiplexing. This is the minimum common feature of those multihoming protocols. All of the protocols or implementations provide a filtering framework and let requesters choose a locator by configuring filter rules. However, we have noticed that two approaches to configure filter rules exist at different layers. One, explicitly defined by the specification, is extending the socket API. Another one, more specific to implementation, is using the existing Operating System's packet filtering mechanism such as PF or NetFilter. More research is thus needed if we want to have a common filtering framework for all the multihoming protocols. However this topic out of the scope in this paper, and as we will discuss in Sec. 5.2, we base our approach on the existing filtering frameworks.

A socket is associated with the local and remote IP addresses and ports. Similarly, the packet filtering mechanism allows to describe flows with a set of selectors. Therefore, most of the protocols fulfill requirement R1.1. However, those approaches usually associate the flow to a system or protocol-oriented path identifier (for example, the BID for MCoA). Thus none of the reviewed protocols fulfill the requirement R1.2.

Some work on flow distribution for multihomed mobile nodes (especially using Mobile IPv6) has already been achieved in the IETF MONAMI6 Working Group. Our proposition presented in this paper has started within MONAMI6 to be later extended to any multihomed system. As a result, Mobile IPv6 enhancements can in theory fulfill the requirements from R2 to R7. There has not

[6] http://gforge.info.ucl.ac.be/projects/shim6

[7] http://hipl.hiit.fi/hipl/

[8] http://lksctp.sourceforge.net

been much work done in other protocols than Mobile IPv6 for the support of flow distribution. Therefore, they hardly fulfill any of the requirements set; socket APIs for locator management are defined for HIP and SHIM6, however, there is no common framework for describing the filter rules and further exchanging those information over network between communicating peers. In SCTP, there is also a rich API for path maintenance and address management for multihoming environment, however, it is outside the scope of the API or the protocol how peers exchange filter rules.

When using various multihoming protocols at different layers at the same time, it is necessary to consider how the flow can be properly multiplexed by a given shim layer (the order and the protocol to be applied) during the outbound or inbound packet processing at the end-system. Because the design totally depends on the OS architecture, we can hardly define a generic flow distribution architecture. We thus propose in the next section a policy management framework that allows users to describe their demands in terms of the needed network characteristics. This mechanism is not going to modify the existing shim layer, but is running on top of the given shim layers by using the flow filtering mechanism specific to the Operating System or the multihoming protocol. Such a framework would fill the gaps of each current protocol and implementations, to process the user's policy and distribute flows among the multiple available paths. This new framework could be applied to the current protocols as well as being used as a guideline for the future ones.

5. A NEW POLICY MANAGEMENT FRAMEWORK FOR FLOW DISTRIBUTION

5.1 Policy data set

The first goal of this new framework is to provide to the requester a generic language to define a common policy data set whatever the protocol or operating system is running on the node.

The data structure of the policy data set is shown in Fig. 2. It is defined as an aggregation of policy rules. Each policy rule is made up of a set of conditions associated to one or several policies. A policy rule specifies what policies must be applied when a set of associated conditions are met [16].

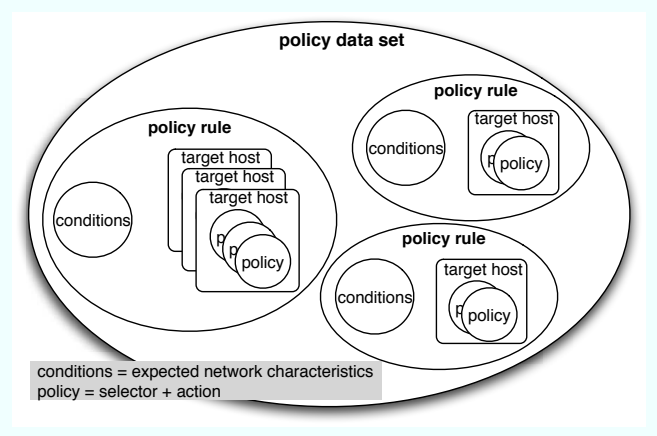

Figure 2: Data structure of the policy data set

The conditions refer to the characteristics wished by the requester. If the conditions match the current characteristics on the node, the policies associated to these conditions are elected. These characteristics are expressed in terms of bandwidth, cost, delay, jitter etc. Multihoming protocols usually uses protocol-specific identifier to define the paths they maintain. Thus, when multiple protocols are running on a single system, common identifiers could be used for simplicity and transparency to the requester. Network characteristics are thus interesting descriptors, as they can be used to describe any kind of path whatever the underlaying protocol is used.

Policies associated to the conditions can be defined for several target hosts. The target host could be the local host, or its peers, identified with their permanent IP address. Each policy associates some selectors (for example the source and destination address, the source and destination ports, the protocol number, etc.) with an action and a lifetime. The action refers to the desired characteristics of a path that should be used to send the flow.

The exact definition of such selectors and path characteristics are still to be defined, but it should be as detailed as possible to describe with accuracy a flow and all the characteristics of a path. Although the actual language to define this data set is a future work, the requirement R1 would be fulfilled by this data set.

5.2 Policy management framework

The second goal of this framework is to process the policy data set by using the existing filtering frameworks provided by the Operating System or the multihoming protocol implementation. For that purpose, this new framework defines several modules (Fig. 3):

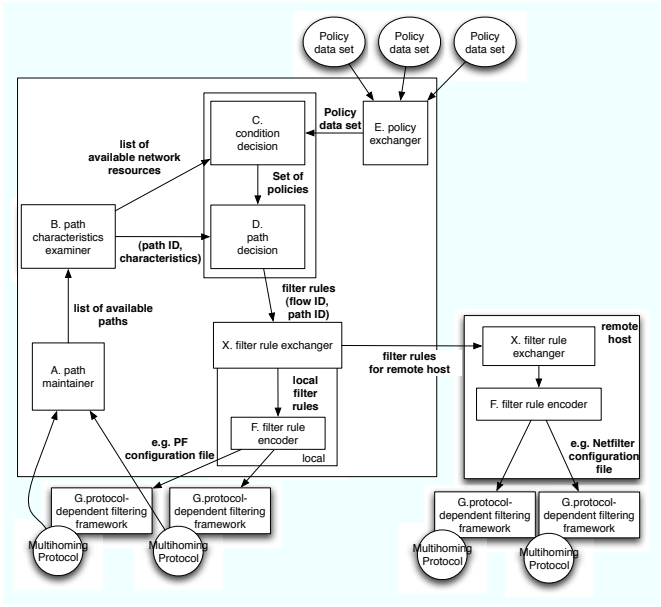

Figure 3: The proposed policy management framework for flow distribution

- The path maintainer (Fig. 3, A) is the multihoming protocol that takes care of establishing and maintaining multiple paths between the node and its peers: MCoA, SHIM6, SCTP etc. The output of this module is a list of the available path, each path being identified by a pair of a protocol identifier (to be defined) and a protocol-specific path identifier (for example the BID for MCoA).

- The path characteristics examiner (B) takes care of building a list of characteristics proper to each path. It takes as an input the list of the available path from the path maintainer module. The path characteristics could be defined in terms

of available bandwidth, costs, average delay and jitter, or any other information that qualifies the path. The exact definition of such characteristics is still to be defined. The output of this module is a list of characteristics for each path, associated with their identifier.

- The condition decision module (C) and the path decision module (D) take care of processing the policy data set previously defined. More precisely:
  - The condition decision module (C) receives from the requesters (local, or remote via the Policy Exchanger (E)) a policy data set. It first merge them to fulfill the requirement R2. Thanks to the list of available network resources received from the path characteristics examiner (B), the module then search for the matching conditions, select the associated policies and transmit them to the path decision module (D).
  - The path decision module (D) makes the relation between the condition decision module (C) and the path characteristics examiner (B). It takes as an input both the set of policies from the former and the list of characteristics/path identifier from the latter, and confront them. For each policy, the path decision module selects a path that matches the policy's action. It thus provides as an output a list of filter rules, that associates a path identifier for each flow. By this way, R3 would be fulfilled. Although definition of the language to describe the output is a future work, R4 would be fulfilled. R6 must be considered during this processing.

- The filter rule exchanger (X) gets the filter rules from the path decision module. Filter rules for the local host are translated to the system-specific packet filtering framework by the filter rule encoder (F), and then installed on the system (G). Filter rules can also be announced to the applications via an API to help them to make their path decision. This modules thus strongly relies on the tools provided by the multihoming protocol and the operating system on which the protocol is operated. Filter rules for remote hosts can be exchanged with the peers using a secure protocol. By this method, R5 will be fulfilled. R7 must be considered when we design the transport protocol.

The Policy Exchanger (E) and the Filter Rule Exchanger could be implemented in the same way as described by the Common Open Policy Service protocol (COPS [3]). The local host getting policy data sets from the requesters would play the role of the Policy Decision Point (PDP) for a given communication, and the target host installing the resulting filter rules would be the Policy Enforcement Point (PEP).

Upon an event (for example, one path is not available anymore), the information can be transmitted from the path maintainer up to the path decision module that will process the policy data set again and provide new filter rules adapted to the current path characteristics.

6. CONCLUSION

In this paper, we introduced a scenario towards a policy management framework on top of several multihoming protocols, and sorted out the 7 requirements that such framework has to meet in order for applications or users to benefit from the multihomed environment. By reviewing the existing multihoming protocols and their implementations, we show that a gap exists with these requirements. We then outlined a new policy management framework for flow distribution and discussed how it fulfills the requirements. As our next step, we would like to improve this draft framework and evaluate it. For this purpose, we plan to implement it and confirm that it matches our expectations and requirements.

7. REFERENCES

[1] A. Campbell, G. Coulson, F. Garcia, D. Hutchison, and H. Leopold. Integrated Quality of Service for Multimedia Communications. In *IEEE INFOCOM*, San Francisco, USA, April 1993.

[2] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Network Mobility (NEMO) Basic Support Protocol. Request For Comments 3963, IETF, January 2005.

[3] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (Common Open Policy Service) Protocol. Request For Comments 2748, IETF, January 2000.

[4] D. B. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. Request For Comments 3775, IETF, June 2004.

[5] M. Komu, M. Bagnulo, K. Slavov, and S. Sugimoto. Socket Application Program Interface (API) for Multihoming Shim. Internet Draft draft-ietf-shim6-multihome-shim-api-02.txt, IETF, March 2007. Work in progress.

[6] R. Kuntz and J. Lorchat. Building Fault Tolerant Networks using a multihomed Mobile Router: a Case Study. In *Asian Internet Engineering Conference (AINTEC) 2006*, Bangkok, Thailand, November 2006.

[7] C. Larsson, H. Levkowetz, H. Mahkonen, and T. Kauppinen. Flow Bindings in Mobile IPv6 and Nemo Basic Support. Internet Draft draft-larsson-monami6-filter-rules-02.txt, March 2007. Work in progress.

[8] K. Mitsuya, K. Tasaka, R. Wakikawa, and R. Kuntz. A Policy Data Set for Flow Distribution. Internet Draft draft-mitsuya-monami6-flow-distribution-policy-03.txt, February 2007. Work in progress.

[9] R. Moskowitz, P. Nikander, P. Jokela, and T. R. Henderson. Host Identity Protocol. Internet Draft draft-ietf-hip-base-07.txt, February 2007. Work in progress.

[10] E. Nordmark and M. Bagnulo. Level 3 multihoming shim protocol. Internet Draft draft-ietf-shim6-proto-07.txt, IETF, November 2006. Work in progress.

[11] K. Park, H. Cho, I. Jang, T. You, and S. Lee. Implementing SHIM6 Protocol. Internet Draft draft-park-shim6-implementation-00.txt, IETF, October 2006. Work in progress.

[12] K. Shima, K. Mitsuya, R. Wakikawa, T. Momose, and K. Uehara. SHISA: The Mobile IPv6/NEMO BS Stack Implementation Current Status. In *Asia BSD Conference (ASIABSDCON)*, Tokyo, Japan, March 2007.

[13] H. Soliman, N. Montavont, N. A. Fikouras, and K. Kuladinithi. Flow Bindings in Mobile IPv6 and Nemo Basic Support. Internet Draft draft-soliman-monami6-flow-binding-04.txt, February 2007. Work in progress.

[14] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. J. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. Request For Comments 2960, IETF, October 2000.

[15] R. Stewart, Q. Xie, L. M. H. Yarroll, K. Poon, and M. Tuexen. Sockets API Extensions for Stream Control Transmission Protocol (SCTP). Internet Draft

draft-ietf-tsvwg-sctpsocket-14.txt, IETF, December 2006. Work in progress.

[16] G. Stone, B. Lundy, and G. Xie. Network policy languages: a survey and a new approach. *Netowrk, IEEE*, 15(i):10–21, 2001.

[17] R. Wakikawa, T. Ernst, K. Nagami, and V. Devarapalli. Multiple Care-of Addresses Registration. Internet Draft draft-ietf-monami6-multiplecoa-02.txt, March 2007. Work in progress.