Software Architecture of a Dynamically Configurable IP Layer

Shinta Sugimoto[†*], Ryoji Kato[†], Toshikane Oda[†]
Ryuji Wakikawa[‡], Keisuke Uehara[*], Jun Murai[‡]

[†]Ericsson Research Japan, Nippon Ericsson K.K.
[‡]Faculty of Environment and Information Studies, Keio University
*Graduate School of Media and Governance, Keio University

Abstract

Today, the IP layer within the TCP/IP stack not only performs IP routing but also manipulates IP datagrams in various ways to meet requirements such as security, mobility and multihoming. Due to the contradiction between the original concept of the Internet and IP sub-layers, utilization of the IP sub-layers in the current systems is only possible in a system-oriented way. In this paper, the limitation of the current TCP/IP system concerning utilization of IP sub-layers is demonstrated, and new software architecture for overcoming the problems is presented. The proposed architecture enables applications to utilize features provided by IP sub-layers selectively, yet preserves Internet transparency and requires no change to protocols.

1. Introduction

Today, the IP layer not only routes datagrams but also manipulates the datagrams in various ways, such as IP encapsulation, encryption or even re-writing of the IP header. Such manipulations are performed at the network layer to meet various requirements. In this paper, we call the mechanisms that implement such manipulation as *IP sub-layers*. Examples of IP sub-layers include IPsec[6][7], Mobile IP[8][9], and SHIM6[5]. Each IP sub-layer is designed to perform an intended function, to fulfill a set of given requirements, with a care not to introduce any side effects to the existing components of the TCP/IP protocol suite.

Although evolution of IP technology gives a wider variety of choices for the upper layer protocols, it is difficult for applications to selectively utilize IP sub-layers. More specifically, applications cannot make the best use of functionalities provided by the IP sub-layers to meet their various requirements. The restriction comes mainly from the mismatch between the basic design of the Internet architecture (i.e., Internet is a single logical address space) and the various requirements from upper layer protocols for the IP layer (i.e., mobility, multihoming, and security).

In the current system, the alternatives are either 1) to force applications to implement code which calls APIs dedicated for a given IP sub-layer, or 2) to let the system decide what IP transformations should be applied to the IP flow. However, neither of the above approaches fulfills the requirements of the upper layer protocols. In the former approach, re-writing of applications may be required to make use of specific APIs, or applications may be required to have detailed information about the IP layer such as a list of available IP addresses and so on. In the latter approach, the source address selection mechanism inside the kernel cannot reliably distinguish each flow.

The objective of this paper is to analyze the current IP sub-layer architecture to identify the problems and to provide a new architecture to overcome the problems. As a result, applications become capable of using IP sub-layers more effectively to fulfill their requirements and also using more than one IP sub-layers at a time in a flexible manner. In other words, we will provide a new mechanism to make it possible that the layering of IP sub-layers can be re-configured and customized to meet each of different requirements of the applications. For this purpose, we propose a new functional component called *IP Broker*, which centrally coordinates interactions of IP sub-layers taking an application's requests into account. We also propose a newly defined API which is provided by the IP layer to upper layer protocols so that the applications can convey their requirements to the IP layer.

The remainder of this paper is structured as follows. In Section 2, common characteristics of IP sub-layers are presented. Section 3 gives the problem statements, showing the limitation of the current system and the necessity of coordination inside the IP layer. In Section 4, the proposed architecture and design principles are presented. Section 5 presents detailed logic of path resolution which is a procedure to sort out what IP transformation is applied to an IP flow in what order.

2. Background and Motivation

2.1. IP Sub-layers

Today, the IP layer consists of various IP sub-layers to meet the various requirements from the upper layer protocols. Figure 1 depicts a functional structure of the IP layer with a number of IP sub-layers. The figure shows a comprehensive view of the IP layer on an end host. We believe that the figure represents a common view within the IETF community, since the layering model is consistent with several documents from IETF[5][13]. As shown, the IP sub-layers are hierarchically arranged and there is a boundary which horizontally divides the IP layer into two parts. The top half consists of IP sub-layers that are processed solely on the end host, whereas the bottom half may be processed at the end host and/or intermediate nodes.

Tunneling

Tunneling is a technique commonly used in the network layer. It is accomplished by encapsulating an original IP datagram by prepending a new IP header. From the network layer point of view, tunneling is a reformulation of an IP packet and thus it is logical that the IP packet goes through the IP layer again, which in turn may result in having another IP transformation being made to the IP packet. IP sub-layers such as Mobile IPv6, IPsec, and SHIM6 employ a form of tunneling.

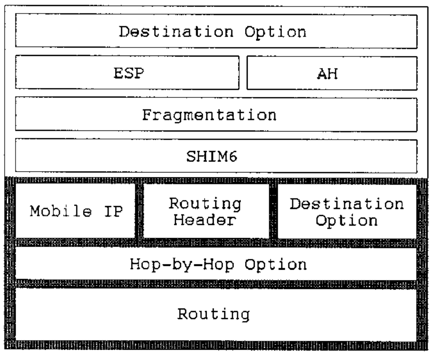

Figure 1: Reference model of IP layer

It is possible that various combinations of IP transformation is applied to a given IP flow. The order of IP sub-layer may not necessarily follow the order of layering of IP sub-layers presented in Figure 1. For instance, even though SHIM6 lays above Mobile IPv6 in the reference model, it does not prohibit an end host from applying SHIM6 to the flow of Mobile IPv6. That is, an IP packet flow which had been transformed by Mobile IPv6 can be subsequently transformed by SHIM6 provided that each IP sub-layer is configured properly.

Figure 2 illustrates several scenarios where more than one IP sub-layer is involved in the end-to-end IP flow between a pair of hosts. The figure shows different combinations of IP sub-layers arranged in different orders to meet the different requirements of the upper layer protocols. In some cases, one or more middle boxes are involved when necessary. Although it is not common to have such complex configurations today, we envision that such usage will be useful to satisfy a wide variety of requirements in the future network environments.

From the above discussion, we conclude that the following design requirement shall be met:

- *Design requirement 1: The IP layer is so designed that combinations and orders of IP sub-layers can be flexibly re-arranged in order to meet different requirements of individual IP flows in various IP networking scenarios.*
- *Design requirement 2: The IP layer shall arrange the IP sub-layers taking the network environment into account*

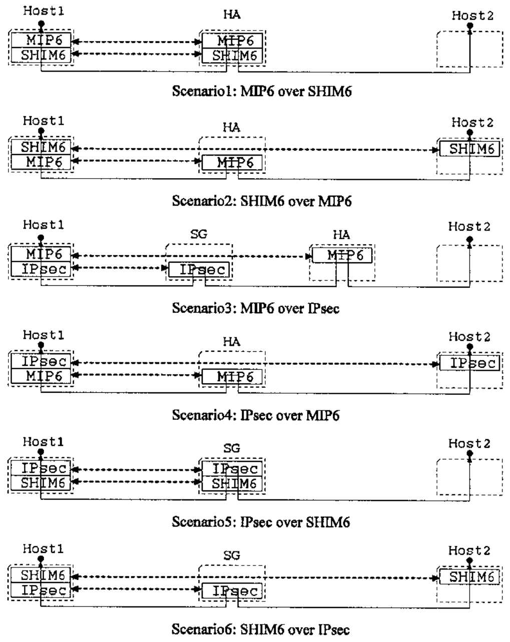
Host1
HA
Host2
MIP6
SHIM6
Scenario1: MIP6 over SHIM6
Scenario2: SHIM6 over MIP6
SG
IPsec
Scenario3: MIP6 over IPsec
Scenario4: IPsec over MIP6
Scenario5: IPsec over SHIM6
Scenario6: SHIM6 over IPsec

Figure 2: Utilization of IP sub-layers

2.2. Internet Transparency

The Internet was originally designed in a way that it is a single universal logical address space [16]. That is, the Internet should be seen by the upper layer protocols as a single logical address space, on which IP datagrams can flow from a sender to receiver unaltered. This fundamental characteristic of the Internet is referred to as "Internet transparency." Basically, the whole system of the Internet including DNS, transport layer protocols, applications, and API is designed based on this assumption. IP sub-layers have been carefully designed so that Internet transparency is preserved. That is, IP sub-layers are kept transparent to the upper layer protocols and thus explicit mechanisms for utilizing IP sub-layers have not been provided, by design. Requiring applications to utilize a specific API to control a specific IP sub-layer may lead to loss of Internet transparency. Thus absence of API for utilizing IP sub-layers is a deliberate design choice. However, RFC 3542[14] specifies how IPv6 extension headers, namely routing header, hop-by-hop option, and destination option header may be accessed by applications.

However, actually, need for IP sub-layer is neither universal nor uniform to the upper layer protocols. There is a need for applications to utilize IP sub-layers to meet their specific requirements. For example, a VoIP application would require session continuity by mean of mobility function in the IP layer, while a Web application would require network layer security rather than mobility support. Like this, characteristics of sessions employed vary and demands for support provided by IP sub-layers are totally application dependent.

Given the requirement for preserving Internet transparency, the present situation is that matching IP sub-layers to IP flows is mainly done implicitly by means of source address selection. The socket API framework allows applications to not specify the source IP address to set up a communication. That is, an application can leave the source address unspecified when establishing a connection. In such a case, the source IP address is selected inside the kernel (i.e., IP layer) according to the given policy and rules [3]. However, this mechanism cannot distinguish different requirements from respective applications and thus utilization of IP sub-layers is done in a system-oriented way rather than an application-oriented way.

From the above discussion, we conclude the following design requirements:

- *Design requirement 3: The IP layer should provide IP transformation services/capabilities to the upper layer protocols without loss of Internet transparency.*
- *Design requirement 4: The IP layer should apply IP transformation to a given IP flow according to the requirements given by the respective application.*

3. Problems

3.1. Limitation of Utilization of IP Sub-layers

There are mainly two approaches for determining what IP transformation should be made for a given IP flow: explicit and implicit approaches.

In the explicit approach, an application proactively selects the IP transformation to be applied to the IP flow. In order to so, an application is required either to use a specific API for utilizing the IP sub-layer or to have detailed knowledge of the IP sub-layer. By using a feature-enriched API (e.g., RFC 3542[14] or BTNS[11][12]), an application can specify what IP transformation shall be made to the IP flow. For instance, RFC 3542 defines extensions to the socket API with which applications can send or receive IP datagrams with IPv6 extension headers. The information specified by the API can be stored in the socket instance, and is taken into account for the subsequent IP packet processing. In the other method, application indicates a set of parameters which is interpreted by the IP layer to designate the specific IP transformation which the application has selected. One of the real deployments is that the application is aware that use of a specific source IP address will designate that the IP flow is to have specific IP transformation applied inside the IP layer. For instance, the application may choose a Mobile IP home address as its source IP address if it requires IP mobility support by Mobile IP sub-layer in the IP layer.

In the implicit approach, the kernel determines the necessary IP transformations to be applied to a given IP flow. This is advantageous for applications since they remain agnostic about the IP layer. In such a case, there is no need for application to have much knowledge of the IP layer, and it leaves the source IP address unspecified in the communication setup. As mentioned earlier, the choice of source address may govern the selection of IP transformation to be made for a given IP flow. In fact, the algorithm defined in RFC 3484[3] for source address selection takes the Mobile IP sub-layer into account; the rule gives preference to Mobile IP home address over normal IP addresses. This implies that Mobile IP is turned on by default, unless an application specifies a source IP address other than the home address.

Although the two approaches mentioned above marginally solve the issue of applying an IP transformation to IP flow, neither of them fully meets the aforementioned design requirements for the following reasons. In the explicit approach, an application is required to get heavily involved in the choice of IP transformation or to have detailed information about IP layer. Therefore, the approach does not fully meet the design requirement 3. In addition, it is too much for applications to monitor the network environment which may be dynamically changing in order to determine what IP transformation is needed. Thus the design requirement 2 is not met. On the other hand, in the implicit approach, there are few requirements for applications. However, selection of IP transformation is made by the kernel by means of source address selection without taking any requirements of applications into account. Thus the approach does not fully meet the design requirement 4.

3.2. Lack of Coordination inside IP Layer

As mentioned in Section 2, there are different IP sub-layers and each has different roles and characteristics. It is possible that more than one IP transformation need to be applied to a given IP flow. In such case, a serialized IP transformation needs to be applied to an IP flow in right order, being consistent to the context information of each IP sub-layers. However, in the current design of the IP layer, such arrangement of IP sub-layers is difficult for the following reasons.

First of all, the source address of the IP flow, the ultimate endpoint of the communication, needs to be determined taking all the necessary IP transformation to be applied to the IP flow. This implies that a serialized IP transformation needs to be resolved prior to the selection of the communication endpoint. Thus the source address selection mechanism deployed in the current TCP/IP systems cannot make the right choice of the local communication endpoint.

An IP sub-layer needs information about the subsequent IP sub-layer that is to apply IP transformation to the IP flow. More specifically, if the subsequent IP sub-layer requires a specific template to apply the respective IP transformation, the template should be understood by the previous IP sub-layer. It is not necessarily true that packet processing to be done at each IP sub-layer is independent from each other. In some case, an IP sub-layer requires information about other IP sub-layers, in particular, the subsequent IP sub-layer which is to apply IP transformation to the IP flow. Therefore, the current TCP/IP system where there is no coordination inside the IP layer does not sufficiently meet the design requirement 1.

3.3. Summary

As discussed in Section 3.1, preserving the Internet transparency and facilitating utilization of IP sub-layers in application-oriented way are two contradicting issues. It is worthwhile to look deeper into the heart of the problem. The root cause of the contradiction is related to the basic design of the Internet, i.e., identifiers and locators are represented as IP addresses[1]. The network based on the design works well in a traditional network configuration such that a host is connected to the Internet with a single network interface and the host never moves. However, problems arise once this assumption does not hold. Today, there are millions of mobile hosts connected to the Internet and large number of multihomed sites. Threats of miscellaneous DoS attacks caused by spoofing network layer information are emerging. Respective efforts made inside the IP layer are countermeasure to each problem; mobility, multihoming, and security.

With regard to the solution space for tackling these problems, there are mainly two approaches; clean slate approach and incremental approach. The former is a ground-up effort to build brand-new network architecture which replaces the Internet[2]. On the other hand, the latter aims to improve the Internet by developing lacking features or by making necessary modifications to the existing systems. Hence the two approaches are antithetical to each other and there are pros and cons. In the clean slate approach, there is greater chance to design the new architecture in a cleaner way taking all the identified issues into account. However, it is hard to deploy, obviously. On the other hand, the incremental approach allows the existing systems to continue to work, yet solves each issues incrementally. However, there is less hope for solving the issues in architecturally right way. In other words, there is a danger of making the Internet as a set of large number of patches, which has a limited performance.

In this paper, we take the incremental approach to solve the problems mentioned earlier in the hope of improving the current situation with minimum cost. Since we re-use existing protocols (IP sub-layers) in our approach, there is no built-in design of ID/Locator separation within the overall architecture. Instead, we introduce a coordinator inside the IP layer that makes arrangement of disjointed piece of IP sub-layers. The coordinator makes the necessary arrangement inside the IP layer, according to a comprehensive picture of a set of IP transformation to be applied to the IP flow.

4. Proposed Software Architecture for IP Layer

In order to satisfy the aforementioned design requirements, we propose a new architecture for IP layer. In this section, the outline of the proposed architecture and design principles considered are presented.

4.1 Design Principles

As for the design principles, we put much value on simplicity and extensibility. Simplicity means less work for kernel, i.e., total amount of work required to achieve the design goals of the proposed software architecture should be kept minimum. Extensibility means that kernel programmers should be able to add new features (e.g. new IP sub-layers) to the IP layer without any drastic changes to the software architecture.

4.2 Overview of Proposed Architecture

In order to achieve the design goal, bringing flexibility to the IP layer of TCP/IP stack of end host, we designed new software architecture for IP layer based on the design requirements and the design principles. Figure 3 illustrates the system overview. A new functional component called *IP Broker* is introduced inside the IP layer, which centrally coordinates resolution of IP transformation for respective IP flows. In our architecture, hierarchical structure of IP sub-layers is removed, in principle, and integrated use of IP transformation becomes possible under the control of applications. The IP Broker arranges the IP stack for each application taking the requirements from respective applications into account.

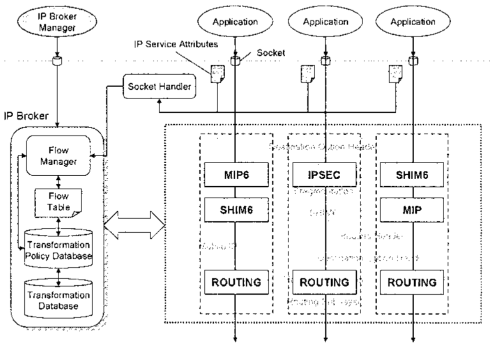

Figure 3: Overview of proposed software architecture for IP layer

The following are the most important design choices made:

1. Introduce a centralized coordinator inside the IP layer which arranges the IP sub-layers
2. Resolution of IP transformation is done based on requirements from applications and network configuration

In the absence of the coordinator, it is fairly difficult to apply more than one IP transformation to an IP flow. In the conventional software architecture of the IP layer, it is all up to the configuration of IP sub-layers made by the system administrator. This implies that the IP layer is not dynamically configurable. Moreover, proper selection of the source IP address is hardly made without the conclusive picture of what IP transformation is applied to a given IP flow. Hence we conceive that the centralized coordinator inside the IP layer is needed. The second design decision is related to the information on which resolution of IP transformation is made by the IP Broker. The ultimate role of IP Broker is to satisfy the needs of applications. For instance, a message "*I want a stable connection!*" is an example of requirement given by an application. Such a requirement indicates that the application wants its IP flow to survive whatever network incidents take place (e.g. change of network attachment point, change of local IP address due to re-homing). The conventional IP layer has never been responsive to such high level requests from applications. In our architecture, determination of IP transformation to be applied to a given IP flow is made by the IP Broker based on requirements from applications and the network configuration.

With the newly defined API, applications can request the IP Broker to provide certain IP services, which may result in applying a set of IP transformation to the IP flow. The API conveys the requirements from the application represented by a uniform data set called *IP service attributes*. Hence the IP service attribute associated with a socket is the key determinant for IP Broker to decide what IP transformation should be made to the associated IP flow. On the left top of Figure 3 sits a dedicated middleware called *IP Broker Manager*, which interfaces IP Broker directly. The roles of the IP Broker Manager are to convey network environment information to the IP Broker, and deliver the IP service attributes to the IP Broker acting on behalf of applications which do not support the newly defined API. Hence the proposed architecture can accommodate both legacy and new applications.

5. Logic of IP Broker

In this section, key design elements of the proposed architecture are presented. We explain the kind of input the IP Broker gets from applications and the system administrator, how an end-to-end path is calculated, and how connection setup is done.

5.1 IP Service Attributes and API

We define a set of information that describes characteristics of an IP flow called *IP service attributes* listed in Table 1. An IP service attribute describes a high level and primitive requirement from the application to the IP flow. The IP service attributes are defined as socket options as shown in Table 2 where each socket option corresponds to each IP service attribute presented in Table 1. The newly defined socket options are set or get by applications with setsockopt() or getsockopt() routines, respectively. Basically the API requires only a flag indicating whether or not to activate a given an IP service attribute or not. Hence the socket options take an integer as an argument of storing option value. For instance, a video streaming client application may give the requirement "Stable Connection" since its session lifetime is normally long. Accordingly, the socket handler inside the kernel stores the information in the socket instance and informs the IP Broker of the need for stable connection for the associated IP flow. If the application does not implement the newly defined API the IP Broker Manager directly informs the IP Broker of the IP service attributes required for certain applications.

Table 1: IP service attributes

Name	Description	Relevant IP sub-layer
Stable Connection	Whether or not the application wants the session to be stable, i.e., the session should have a capability to survive regardless of whatever network incidents take place	Mobile IP, SHIM6
Secure Connection	Whether or not the application wants the session to be secure. Security properties include data confidentiality, data integrity, and data origin authenticity.	IPsec (BTNS)

Table 2: Socket options for IP service attributes

Option name	Level	Option value
IP_ATTR_STABLE	IPv4, IPv6	int
IP_ATTR_SECURE	IPv4, IPv6	int

5.2 Network Modeling

In the proposed architecture, it is necessary for IP Broker to see the picture of the network configuration in order to figure out the best IP transformation to be applied to a given IP flow. Figure 4 is an example of network model. As shown, a network model is represented in a directed graph. In the model, any node that is involved in the routing and/or transformation of the IP datagram is denoted as a vertex. A vertex can also represent a set of routers (i.e., subnets). Additionally information is given to a vertex indicating what IP transformation the node can offer. One or more IP addresses can be assigned to a vertex, which are candidate IP addresses that end host can use as a communication endpoint. Normally an IP address which is specified as a communication endpoint is topologically situated at the end host itself, i.e., the IP address is assigned to the vertex representing the communicating host. However, sometimes it is possible that the IP address is assigned to a vertex which performs a certain IP transformation inside the network. Middleboxes such as Mobile IP home agents and IPsec security gateways[15] exploit this tunneling technique, and thus an inner IP address assigned to the client is topologically situated at the subnet of middlebox.

Figure 4 gives an example of network configuration where a host (Host1) is served by multiple Mobile IP home agents, HA1 and HA2. The upper half of the figure shows the actual network configuration whereas the lower half of the figure shows the network model. The vertex labeled 'INT' represents the Internet, where the peer (Host2) is connected. As shown, there are three IP addresses available for Host1, IP1@Host1, IP2@HA1, and IP3@HA2. Mobile IP home agent creates the illusion that a client is always connected to the home network, and thus Host1 is under an illusion that it is continuously connected to the home networks served by HA1 and HA2, respectively. In this sense, Host1 is considered to be a multihomed host.

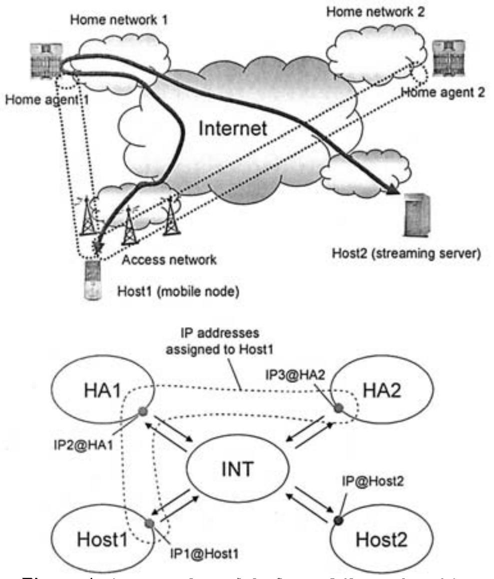

Figure 4: A network model of a multihomed mobile network

A network entity such as a Mobile IP home agent is pre-deployed and requires an agreement established between the server and the client prior to the service offering. Another example of pre-deployed network service is IPsec in authenticated mode. In a VPN, a typical deployment of IPsec, client and security gateway requires pre-deployed information, namely an identifier and associated credentials for authentication. On the other hand, there are some IP services that do not need any pre-deployed information. For instance, SHIM6[5] is designed in a way that the protocol runs between a given pair of peers that have no prior security association. BTNS[10], which is an unauthenticated mode of IPsec, does not require any pre-deployed information for peers to establish an IPsec security association.

To summarize, in the network model, in addition to the network configuration, information related to IP sub-layers that come with pre-deployed information should be given as input to the IP Broker. The input consists of 1) adjacent matrix representing the network topology in a directed graph, 2) IP service attribute associated with each vertex, and 3) associated IP address(es).

5.3 Path Resolution

The flow manager performs path resolution based on the IP service attributes requested by the application and the information about the network configuration. A path is a sequence of nodes which perform any IP transformation to the IP flow on the path between the sender and receiver. Normal IP routers that perform plain IP forwarding are omitted for simplicity. Hence the path represents not only topological path but also IP transformation to be applied to a given IP flow. Note that the path resolution is unidirectional. The result of path resolution may trigger setup of an IP sub-layer context in some case. The path resolution is done in the following four steps.

1. **Path search** – Find all possible unidirectional path from the sender to receiver
2. **Selection of local communication endpoint** – Determine which IP address should be used as local communication endpoint in each calculated path
3. **Selection of path** – Select path taking IP service attribute into account
4. **Multihoming** – Apply end-to-end multihoming context if necessary

The first step executes path search to sort out all the possible paths. Path search is done in a simple way, based on permutations of middleboxes and selection of multihomed paths. The number of paths is the product of 1) summation of permutation *nPm* (choosing m elements out of a permutation whose size is n), 2) the number of the Internet connectivity that sender has, and 3) the number of the Internet connectivity that receiver has. This means that the number of calculated paths grows exponentially with the number of middleboxes. However, we conceive that the algorithm does not cause any serious problem since a realistic number of middleboxes is 3-4 at most. The summation of *nPm* (where m is larger than 0 and less or equal than n) is 16 (n=3) and 65 (n=4), respectively. Multiple number of Internet connectivity of sender or receiver indicates that the end host is multihomed. As a result of path search, normally there are multiple candidate paths calculated. Accordingly, the paths are carefully selected taking various conditions into account. Figure 5 shows an example of path search resulted from a network condition given in Figure 4. As shown, there are five paths (Path1 to Path5) discovered by the path search.

```
Path1: Host1* → Int → Host2
Path2: Host1* → Int → HA1* → Int → Host2
Path3: Host1* → Int → HA2* → Int → Host2
Path4: Host1* → Int → HA1* → Int → HA2* → Int → Host2
Path5: Host1* → Int → HA2* → Int → HA1* → Int → Host2
```

Figure 5: An example of path search result

Table 3: Path attributes relevant to path resolution

Path	Local communication endpoint (LCE)	Redundant flag	Candidate path	Additional multihomed path
Path1	IP1@Host1	False	False	**True**
Path2	IP2@HA1	False	**True**	N/A
Path3	IP3@HA2	False	**True**	N/A
Path4	IP3@HA2	**True**	False	False
Path5	IP2@HA1	**True**	False	False

Following path search, selection of local communication endpoint (LCE) is made for each path. The selection is made by examining the IP address along the path from the peer to the sender host. That is, each vertex on the path is verified, in turn, if there is any IP address available for the sender host. If any IP address is found, the IP address is chosen as a local communication endpoint.

In the next step, selection of path is made carefully taking IP service attribute into account. In the example shown in Figure 4 and Figure 5, if an IP service attribute IP_ATTR_STABLE is specified by the application, Path1 is judged invalid since it does not satisfy the need for IP mobility support. In the example, Path4 and Path5 are considered to be inferior to Path2 and Path3, respectively. This is because the same type of IP transformation (IP Mobility) is applied to the flow twice, which increases the number of hop counts unnecessarily. In the example of Figure 4 and Figure 5, an IP transformation to be done by a single Mobile IP home agent is enough for satisfying the needs of stable connection requested by the application. In this paper, a set of the paths that are judged valid during the path selection is called *candidate path set*. In the example, Path2 and Path3 are included in the candidate path set.

The final step, multihoming support, is taken when necessary. If an application requests a stable connection and there are multiple paths included in the candidate path set, necessary settings and configuration is made for multihoming support. It is assumed that the multihoming support is accomplished by applying SHIM6 to a given IP flow. From each of candidate path set, a local communication endpoint is taken and a set of candidate upper layer identifiers (ULID) is formulated. Accordingly, each of the paths from the set of invalid paths is examined to see if there are any additional paths available. Note that multihoming support may increase the number of paths that peers can use. Figure 6 shows the pseudo-code for the additional multihome path search. For each path in the candidate path set, local communication endpoint is compared with that of each invalid path. If it is different, the path is considered as an additional multihomed path. The local communication endpoint from the path is added to the locator list associated with the ULID.

```
1: for all Path_c in Candidate Path Set do
2:   for all Path_i in Invalid Path Set do
3:     if LCE of Path_i is different from LCE of Path_c and
4:        redundant flag of invalid path is not set then
5:         mark Path_i as additional multihomed path
6:     end
7:   end
8: end
```

Figure 6: Pseudocode for searching for additional multihomed paths

5.4 Connection Setup

After path resolution is completed, the IP Broker needs to take some necessary procedures so that the application can setup a connection. The following procedures should be taken according to the results of path resolution:

1. Determine source address for the IP flow
2. Update transformation databases

The source address of a given IP flow is determined by the IP Broker according to the results of path resolution. The local communication endpoint of the selected path should be specified as the source IP address of the IP flow. If multihome support is provided by IP Broker, then the source address of the IP packet should be the local ULID of the SHIM6 context to be created. The source address selection made by IP Broker is substantially different from the conventional source address selection mechanism. The most significant difference is that source address selection is done in an integrated fashion, by taking into account application's requirement, available IP addresses, and network configuration. Therefore, it becomes possible for the IP Broker to meet multi-dimensional requirements at a time. Secondly, the selection is made per application. In the conventional approach, the selection is made by the uniform rules that are applied irrespective of requirements of application.

The second procedure is to dynamically apply settings and configuration necessary for applying IP transformation according to the results of path resolution. More specifically, transformation databases need to be updated so that IP transformation can be applied to the IP flow aligned with the path information. Since the pre-deployed information cannot be modified or changed by the IP Broker, the main task of the IP Broker is to arrange the IP sub-layers that require no pre-deployed information. In the example presented in Figure 4 and Figure 5, the communicating peers are recommended to establish a SHIM6 context to make the best use of multiple paths. Therefore, necessary settings should be made so that SHIM6 can initiate context establishment according to the results of path resolution.

6. Related Work

The Host Identity Protocol (HIP)[4] suggests a substantial design change of Internet architecture by separating identifier and locator of the Internet objects. In the HIP architecture, a new namespace for identifier is introduced and identifiers are presented to the upper layer protocols for the use of communication endpoints. Thanks to the separation of identifier and locator, the architecture can accommodate various kinds of network scenarios such as mobility and multihoming. In addition, leveraging the self-certifying identifier, host can claim the ownership of its identifier without relying on any infrastructure. Thus, HIP is a promising technology in the long run and a clean way to solve the problems of the Internet. However, downside of the protocol should also be considered. Due to the host-centric design, transition is hardly achieved unless the majority of the end hosts become HIP ready. A globally scalable infrastructure bootstrapping the mapping of identifier and locator is needed in communication setup. Hence there are some necessary steps to take for HIP to deploy. The proposed architecture, in comparison to HIP, aims to solve the miscellaneous problems at the network layer by re-using existing protocols rather than creating a new protocol.

7. Conclusion

In this paper, we demonstrated the limitations of the current TCP/IP system concerning utilization of IP sub-layers. There are contradicting roles for the IP layer; to provide a universal single address space to the upper layer protocols and to provide different IP services to respective IP flows to meet various requirements. Due to this contradiction, it is difficult for applications to make the best use of IP sub-layers in the current architecture. Hence we argue that a new architecture for the IP layer is required.

The proposed new architecture for the IP layer aims to improve the current situation by reaching a middle ground; it keeps the transparency of the Internet, yet allows applications to utilize IP sub-layers selectively. A new functional component called IP Broker brings flexibility to the IP layer, i.e., combination and order of IP transformation can be flexibly arranged in accordance with the requirements given by application and the network configuration. The acquisition of flexibility is a significant step for end hosts on the Internet to meet multi-dimensional requirements from the applications to the network layer.

The core feature of IP Broker is path resolution, by which the combination and order of IP transformation to be applied is determined based on the IP service attributes and the network environment information. The path resolution is comprised of four steps: 1) path search, 2) selection of local communication endpoint, 3) path selection, and 4) adaptation of end-to-end multihoming.

8. Acknowledgement

The authors would like to express sincere thanks to Professor Rodney Van Meter of Keio University and Dr. Pekka Nikaneder of Ericsson Research in Finland for providing valuable comments.

References

[1] J. Noel Chiappa, “Endpoint and Endpoint Names: A Proposed Enhancement to the Internet Architecture,” 1999.
[2] GENI.net: Global Environment for Network Innovations, http://www.geni.net
[3] R. Draves, “Default Address Selection for Internet Protocol version 6 (IPv6),” IETF RFC 3484, February 2003.
[4] R. Moskowitz, P. Nikander, P. Jokela, T. Henderson, “Host Identity Protocol,” draft-ietf-hip-base-09.txt, work-in-progress, October 2007.
[5] E. Nordmark, M. Bagnulo, “Shim6: Level 3 Multihoming Shim Protocol for IPv6,” draft-ietf-shim6-proto-08.txt, work-in-progress, May 2007.
[6] R. Atkinson, “Security Architecture for the Internet Protocol,” RFC 1825, August 1995.
[7] S. Kent, K. Seo, “Security Architecture for the Internet Protocol,” RFC 4301, December 2005.
[8] C. Perkins, B. Patil, P. Roberts, “IP Mobility Support for IPv4,” RFC 3344, August 2002.
[9] D. Johnson, C. Perkins, J. Arkko, “Mobility Support in IPv6,” RFC 3775, June 2004.
[10] N. Williams, M. Richardson, “Better Than Nothing Security: An Unauthenticated Mode of IPsec,” draft-ietf-btns-core-05.txt, work-in-progress, September 2007.
[11] M. Richardson, N. Williamas, M. Komu, S. Tarkoma, “IPsec Application Programming Interfaces,” draft-ietf-btns-c-api-01.txt, work-in-progress, July 2007.
[12] M. Richardson, “An interface between applications and keying systems,” draft-ietf-btns-abstract-api-00.txt, work-in-progress, June 2007.
[13] D. Thaler, “A Comparison of IP Mobility-Related Protocols,” draft-thaler-mobility-comparison-02.txt, work-in-progress, October 2006.
[14] W. Stevens, M. Thomas, E. Nordmark, T. Jinmei, “Advanced Sockets Application Program Interface (API) for IPv6,” RFC 3542, May 2003.
[15] B. Carpenter, “Middleboxes: Taxonomy and Issues,” RFC3234, February 2002.
[16] B. Carpenter, S. Brim, “Internet Transparency,” RFC 2775, February 2000.