Implementing and Using HIP with IPv6

Mika Kousa
Helsinki University of Technology, Espoo

Until recent years hosts in the Internet have been mostly stationary. Static hosts used static addresses. Recently this situation has changed dramatically due to rapidly increased number of mobile hosts. These mobile hosts might even have multiple connections to the outside world. Another hugely grown area in networks is security. Especially mobile hosts require security. Current widely used protocols have not been designed with security, mobility, and multihoming in mind. This has required new protocols to be developed.

This paper presents how Host Identity Payload tries to address all of these issues by creating a new Host Identity namespace and a protocol which takes advantage of this namespace. Also other advantages and disadvantages of HIP are discussed. Some issues on transition phase from plain IPv6 to IPv6 with HIP support are covered.

1. INTRODUCTION

Current Internet's core, Internet Protocol version 4, is about to become insufficient in the near future due to extreme growth of the number of connected hosts. The next new version of the Internet Protocol is IPv6, which has been actively researched and developed lately, fixes problems IPv4 has and adds benefits IPv4 did not have at all.

New and complex protocols have to be studied well in order to make them popular and make transition phase from previous version smooth. Practise has shown that protocols which were considered to be well defined might still contain errors or other serious issues, therefore it is important to research alternative ways to implement the essential parts of the new protocol. Lack of alternative implementations for a protocol might cause world wide problems or security threats if the error in protocol is found a long time after the protocol has already spread widely. Because IPv6 is not yet in widespread use, the time for this research is as soon as possible before IPv6 really starts gaining more momentum.

This paper shows how Host Identity Payload (HIP) achieves some of the issues IPv6 considers important such as mobility and multihoming and what is needed in transition phase from plain IPv6 to IPv6 with HIP support.

Chapters in this paper are divided as follows: chapter 2 presents an overview of HIP, chapter 3 discusses what advantages HIP provides and what disadvantages HIP has. Transition phase is covered in chapter 4 and some conclusions are given in chapter 5.

2. OVERVIEW OF HIP

This chapter discusses reasoning why HIP was developed by showing problems in current namespaces. An overview of the HIP protocol is also presented.

2.1 Current namespace issues

There are currently two major namespaces in the Internet: Internet Protocol addresses and Domain Name Service (DNS) names [DNS]. IP addresses are used to name networking interfaces. DNS names are designed to be easily remembered by humans instead of IP addresses which are better suited for computers. Both schemes are used in many different ways, such as address based access control.

Recent development has shown some problems. Semantic overloading (use of these addresses in many different situations) and additions to functionality have complicated these namespaces [ARCH]. More and more of the hosts are becoming mobile. When a host visits a network and disconnects, its interfaces may get different addresses when it reconnects again to the network later.

If transport layer is tightly bound to IP address, it makes separate development of transport and internetworking layers hard.

Domain names provide hierachically assigned names for hosts, which diminishes anonymity.

In short, the problems are dynamic readdressing of hosts (mobility), lack of anonymity and lack of authentication for systems and datagrams.

To overcome these problems, we would need a names-

Author's contact information: mkousa@cc.hut.fi

Published in *Internet Protocols for Mobile Computing - Seminar on Internetworking, Autumn 2002*, by Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory. The complete publication can be found at http://www.tml.hut.fi/Studies/T-110.551/2002/papers/December/index.html.
ISBN 951-22-6272-X - ISSN 1455-9749 - HUT - TML - TML-C9

pace which could be used in end-to-end operations independent of the evolution of the internetworking layer and throughout internetworking layers. This would support mobility. Authentication mechanisms would benefit from cryptographically based namespace. Creation of addresses from this namespace locally without requiring registration provides anonymity.

Next chapter 2.2 gives a basis for the discussion and chapter 3 presents how HIP actually addresses these problems.

2.2 HIP namespace

2.2.1 *Host Identity.* HIP creates a new namespace called Host Identity (HI). Host Identities are usually associated to the operating system's IP networking stack.

Two types of Host Identities exist: "well known" public HI's (can be published in DNS or some other known place) and anonymous. Every host has at least one HI, client hosts have usually both types of HI's.

Host Identities are cryptographic in nature. The best Host Identities are the public key part of an asymmetric key-pair. Host Identities based on a public key can be used in HIP packet authentication and for protection from man-in-the-middle attacks.

Host Identities are potentially long, so they are not used directly in Internet protocols. Instead, they are passed to HIP protocol payload, which is discussed in section 2.3. HIs can be stored e.g. in DNS or LDAP [LDAP] directories.

Because there are many different public key algorithms and key lengths in which full Host Identity could be used, HI is not ideal to be used as such e.g. as a packet identifier or as an index in some HIP related table. Since protocol headers have usually fixed length addresses, another presentations of full Host Identity have been defined [ARCH] to be used in protocols. They are discussed next.

2.2.2 *Other Representations of Host Identity.* A Host Identity Tag (HIT) is a 128 bit long hash over Host Identity. Fixed length eases protocol coding, control of packet sizes, and it has a format suitable for underlying protocols despite of identity technology used.

Local Scope Identity (LSI) is a 32 bit long local representation of HI. LSIs are designed mainly for IPv4.

2.3 Host Identity Payload protocol

The new namespace requires a new protocol, the Host Identity Payload protocol. HIP protocol is used for exchanging namespace identities and continuity between those hosts independent of the networking layer [MOS]. HIP protocol payload is transferred on a layer called Host Layer. Host Layer is located conceptually between the internetworking (Internet Protocol layer) and transport layers (e.g. TCP) [ARCH].

The most important function HIP protocol performs is the base exchange, which performs end-to-end host authentication between initiator and responder hosts. It establishes also Security Associations for ESP [ESP]. Base exchange is discussed in section 2.3.1.

Three other type of packets are needed for controlling the properties of an existing connection or some other HIP related information between the hosts. These packets are described in section 2.3.2.

2.3.1 *Base Exchange.* Base exchange is similar to the TCP connection establishment procedure [TCP]. TCP has a three way handshake to establish state between two hosts.

HIP protocol needs four packets to establish state between two hosts. Base exchange has been carefully designed to avoid possible Denial Of Service and Man in the middle attacks. TCP and UDP [UDP] have also protection when HIP enables use of ESP via Security Associations. This model is shown in figure 1.

First packet called I1 packet is just a request from the initiator to the responder to start the exchange. This packet might be spoofed, so responder does not perform any time consuming operations on this packet. Duplicate packets received in a short period of time are discarded to avoid packet flooding attacks.

Second packet is called R1 which is sent as a reply to the I1 packet. This packet contains responder's public value of Diffie-Hellman key [DIFFIE] and information on responder's supported encryption algorithms and ESP modes. Additionally, responder creates a puzzle which the initiator must solve in order to make a successful base exchange. Puzzles are designed in such a way that solving them requires significantly more time than creating them. If responder has many simultaneous connection attempts, it can simply create a harder puzzle. Responder could also set puzzle difficulty based on its level of trust of the initiator. Precomputed puzzles and re-using R1 packets provide additional security to avoid resource consuming DoS attacks. When the packet is created it is signed using Diffie-Hellman algorithm and the result of the signature is appended to the packet.

Third packet, I2, contains initiator's public value of Diffie-Hellman key, initiator's selection of encryption algorithm supported by the responder, encrypted Host Identity of the initiator using selected encryption algorithm, Security Parameter Index for responder's ESP, and the answer to the puzzle. Signature of the packet is appended to the packet.

The fourth and the last packet is R2, which contains initiator's SPI and signature over the packet.

Retrieval of Host Identities must be secure. The HIP specifications require that implementations must support at least DNSSEC [DNSSEC] for authentication of Host Identities. HIs of type public key should be stored in a DNS KEY RR with the protocol set to HIP [ARCH]. When initiator receives R1 packet, it retrieves responder's HI from a signed DNS zone and uses it to validate the packet. Responder does the same when it receives I2 packet. Host must validate signatures when it receives R1, I2 or R2 packet. If initiator's HI is anonymous responder host may decide not to accept base exchange to avoid Man-in-the-middle attack.

After the base exchange is finished successfully, both hosts have authenticated themselves to each other and set up their Security Associations. After this, connection continues using ESP.

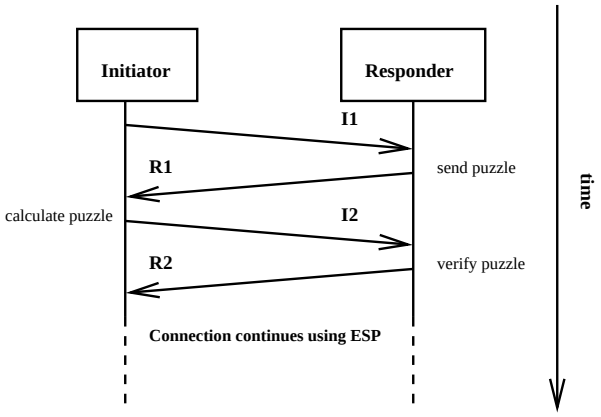

Fig. 1. Base Exchange

2.3.2 *Other packets in HIP protocol.* Efficient usage of HIP protocol requires some controlling packets. These packets are used to update the state of the HIP protocol. HIP protocol defines three packets: BOS (the HIP Bootstrap Packet), NES (the HIP New SPI Packet), and REA (the HIP Readdress Packet).

BOS is used in situations where initiator can not retrieve responder's address from DNS or some other repository. This might be the case when HIP is used e.g. in DHCP [DHCP]. BOS is a signed packet which contains announcer's full Host Identity and Host Identity Tag (HIT) and possibly a signature of announcer's HI. This packet is usually broadcast periodically.

NES provides a new SPI to the peer. NES packet can also provide a new Diffie-Hellman key to be used within HIP connections.

Readdressing packet, REA, is needed to notify its partners of an address change e.g. due to renewal of a DHCP lease [DHCP], Internet reconnect or some mobility event. REA packet payload contains IPv4 type of address (resource record A in DNS) or IPv6 type of address (resource record AAAA in DNS).

As can be seen, the protocol is quite simple yet it provides enough functionality.

Now that we have introduced the Host Identity Payload, its concepts and architecture on a detailed level enough, we continue with some networking related issues HIP tries to solve or ease. Since every complex protocol has surely some downsides, we also discuss potential problems with HIP. Next chapter lists advantages and disadvantages when using HIP.

3. HIP ADVANTAGES AND DISADVANTAGES

This chapter shows how HIP approaches several important networking concepts such as multihoming and mobility using previously described architecture of HIP. In addition to these, other advantages and disadvantages are also listed.

A notable advantage of using Host Identities and HIP protocol is that they decouple internetworking and transport layers. Transport is bound to the Host Identity. This makes independent development and evolution of these layers possible. This decoupling also eases implementation of HIP based multihoming and mobility as is shown next.

3.1 Multihoming with HIP

A typical multihomed host has more than one network interface, each of which are connected to different networks (a multihomed host may also have only one network interface if it has two IPv6 network prefixes). These interfaces have therefore different addresses. Due to different addresses, a multihomed host seems to be located in different locations within the network. Operating system keeps track of mapping between the address and corresponding interface.

With HIP namespace, mappings are created between IP addresses of all interfaces and a single Host Identity. Every IP address is mapped to the same HI. IP addresses present a topological location within the network, and the HI is used as the end-point. Now the multihomed host looks like it is located in one place when communicating with HIP.

3.2 Mobility with HIP

A system is considered mobile if its IP address can change dynamically for any reason like an address received from DHCP [ARCH].

3.2.1 *Initiator Mobility.* Initiator iniated mobility is simple. Responder can accept a HIP or ESP packet (whose SPI is created by HIP) from anywhere. IP address is then used only for routing the packets back. Initiator may also send readdressing (REA) packets (see chapter 2.3.2) to the responder to tell the initiator's new location.

3.2.2 *Responder Mobility.* In responder mobility, the initiator needs to know the location of the responder. This method uses a rendezvouz server. The responder updates its IP address continuously to the rendezvouz server using readdressing (REA) packets (see chapter 2.3.2). Before that the responder must set up a HIP based Security Association with rendezvouz server. The

rendezvouz server must have received responder's Host Identity from a trusted source in order to avoid attacks which might provide wrong information on the mapping between responder's HI and IP address. The initiator sends its initial HIP packet destined to the responder to the rendezvous server which forwards the packet onward to the responder's current location. After these packets are exchanged directly between the initiator and the responder, and responder mobility is handled like initiator mobility. Rendezvous server is discussed in more detail in [HIPIMP].

3.3 Other Advantages of HIP

3.3.1 *Host Identity Related*

—Host Identities provide a consistent name for a host regardless of how it connects to the Internet.

—Cryptographically based public key -like names are hard to spoof assuming they are retrieved securely.

—HIP separates the routing and host namespaces with no effect on routability. Role of the IP address changes to simply a packet forwarding namespace.

—Host Identities are interface independent.

3.3.2 *Security and Networking Related*

—Protocol is very simple and bandwidth conservative.

—Secure authentication of hosts with base exchange (chapter 2.3.1).

—Fast key exchange and setting of Security Associations for IPSec ESP transport mode with HIP protocol. This might diminish the need for complex external infrastructures such as IKE [IKE] even though HIP does not provide all the same functionality as IKE.

—Rendezvouz server does not need a lot of processing power (only a couple of packets per request). One server could easily handle a large number of hosts.

3.4 Disadvantages of HIP

3.4.1 *Host Identity Related*

—Decoupling of internetworking and transport layers and adding a new host layer might require changes to the existing APIs and applications. This is discussed in more detail in chapter 4.4.

—Locally created anonymous Host Identities makes resolvability very difficult for other hosts.

3.4.2 *Security and Networking Related*

—Encryption and decryption requires extra data processing costs for systems. This present a problem for low performance devices such as handhelds, mobile devices etc (see [KOMU]).

—Use of DNS increases because Host Identities can be saved to DNS records. Increased DNS record sizes and DNS updates cause increased bandwidth use and storage requirements. New records create larger packets for replies, therefore TCP might have to be used instead of UDP when querying DNS for HI, causing higher latency.

—ESP protected data transfer makes creation of security policies harder. Firewalls can not peek into encrypted packet and decide if the packet should be allowed to pass or not [V6OPS].

—Changing HIP to support multicasting may require a substantial amount of work.

After all, every networking protocol has also its human point of view. Computer networks are designed for people, not just for machines which can learn any new protocol within a microsecond. Learning curve for new things is high or low depending on the person, but it still exists.

4. TRANSITION PHASE FROM IPV6 TO IPV6+HIP

Let us assume now that someone who is using plain IPv6 protocol is interested in trying out how IPv6 with HIP support is used in the real world.

This chapter presents required issues when HIP is considered to be used; groundwork (protocol implementation) and transition phase from IPv6 to IPv6+HIP.

HIP is still under development and therefore considered experimental. Documentation consists mostly of Internet Drafts, there are not yet any officially accepted RFCs on HIP. Initial implementation work has been started by several groups, but by the time of writing this paper there are no ready protocol implementations which are in full conformance with the drafts.

4.1 Requirements for Transition Phase

When speaking of network protocols, transition phase is not something which can be done overnight whether the change is big or small. Even the smallest changes to existing use of networking protocols in a system require careful planning by the network administrators. Careless preparation and execution of transition phase might lead to a major catastrope and huge costs due to reverting to the previous situation before the transition was started.

4.2 Similarities with Other Transitions

Transition phase from IPv6 to IPv6+HIP has similar issues and considerations as transition from IPv4 to IPv6 [TRA]. Successful transition presents two major requirements: compatibility and implementation.

—*compatibility:* One of the most important things in transition phase is compatibility between hosts. Compatibility makes transition easier when HIP hosts remain compatible with non-HIP hosts. As is seen with transition from IPv4 to IPv6, previous estimates and plans for IPv6 deployment all over the world have not realized in practise, and it is still not known when IPv6 will be the most used protocol in the Internet. It is assumed that the number of hosts supporting HIP will grow very slowly due to current state of HIP development, therefore it is important to maintain interoperability between hosts which support HIP and hosts which do not support HIP as long as necessary. One way to achieve compatibility is to use dual layering method descibed in section 4.2.1.

—*Implementation:* Full interoperability requires that protocol implementations must be complete. Use of

incomplete protocols might cause problems and erroneous situations when they are used with complete implementations.

4.2.1 *Dual Layering.* When moving from IPv4 to IPv6 while maintaining compatibility between different hosts, a technique called dual stack is used. In dual stack operating system's networking code provides support for both protocols. This makes simultaneous IPv4 and IPv6 connections possible without them interfering each other. Similarly, an IPv6 host which has support for HIP must not prevent normal use of IPv6 when it needs to create a plain IPv6 connection to a host which does not have support for HIP. When HIP enabled host tries to use HIP with non-HIP host, non-HIP host should send back an ICMP "Destination Protocol Unreachable" message [ICMP].

4.3 Requirements for Other Protocols

New namespaces usually present new requirements for other protocols than for the protocol for which the namespace was developed on. A typical example of this is the Domain Name System (DNS) which is used for mapping between IP addresses and hostnames. IPv4 addresses are stored into DNS using a resource record of type A. IPv6 addresses cannot use the same resource record as IPv4, so a new resource record had to be created which is known as the AAAA record.

HIP requires changes to DNS databases, too, but it does not require a new resource record to be created. Instead, public HIs are stored in a KEY resource record with the protocol set to HIP. When Host Identity Tags (HITs) are stored into DNS, they have have all flags set to ZERO, protocol set to HIP, and algorithm set to HIT128. Anonymous HIs are not stored in DNS [MOS].

Interoperability requirement (see chapter 4.2) presents a new requirement for resolver libraries. Resolver libraries must be capable of handling both types of addresses, e.g. A and AAAA when transitioning from IPv4 and IPv6. Resolver issues are discussed below in chapter 4.4.3. After these essential issues are discussed, we present an example program in section 4.5 which shows how simple converting existing IPv6 programs can be.

4.4 Requirements for APIs

New protocols almost always require semantic and syntactical changes to the existing applications and application programming interfaces (API) those applications use. These modifications involve at least network related data structures (chapter 4.4.1) and name resolution functions (chapter 4.4.3). Different ways to implement HIP support for APIs are discussed in chapter 4.4.2.

4.4.1 *Socket Address Structures.* One of the most common concepts in network programming are socket address structures which are used for representing addresses [STE]. Every address family has its own socket structure. In C language, they are known as structure sockaddr_*xxx*, where *xxx* refers to the address family used, e.g. *sockaddr_in6* is used for storing Internet protocol version 6 addresses. Socket address structures are passed to network related function calls. Examples of these calls are name service, socket creation, connection establishment, and data transfer calls.

4.4.2 *Transparent and Non-transparent API.* Adding a support for a new protocol can be achieved in two ways, either transparently or non-transparently. Transparency means that the syntax of the functions will have no changes, there will be only semantic modifications inside the functions. This makes the actual operation of the function invisible to the program developers, they can continue using the function as they have been using it before the modification thus easing the transition. Transparency if often hard to implement because programs must execute the same way before and after they are changed to use the new function. In non-transparent mode, support for the new protocol is built explicitly.

Non-transparent HIP implementation would need a new socket address structure. This creates a drawback: every socket related function would have to be inspected and changed accordingly to support the new HIP socket address structure.

If we look at the specification of HIT (chapter 2.2.2), we note that HIT has the same length as IPv6 addresses (128 bits). Additionally, HIP drafts define two formats for HIT addresses, which can be separated by looking at the first two bits of the HIT. These two formats are designed to avoid the most commonly occurring IPv6 addresses in RFC 2373 [ADDRARCH]. More details on these HIT formats can be found from [MOS].

These two issues are enough for the operating system's network stack to distinguish between normal IPv6 and HIT addresses, thus making the transparent API for HIP possible. When HIP connections are used, a HIT address will be placed instead of an IPv6 address. When the operating system checks the address type, it decides whether the connection proceeds using HIP or normal IPv6. This way we can even reuse existing socket address structure of IPv6 (*sockaddr_in6* in C language).

Actual support for applications in initial implementation phase is achieved by recompiling program source code with modified versions of patched libraries which have HIP support. Programs which do not use DNS or old binary distributions will not most probably provide support for HIP.

4.4.3 *Name Resolution Services.* When contacting remote hosts, we usually use human readable names for hosts instead of numerical addresses. Addresses returned by DNS depend on resolver library's configuration. With IPv4/IPv6 dual stack, the host must be capable of handling both types of addresses received from the DNS. The resolver may perform filtering and/or reordering on the results (perhaps due to a policy set by an administrator) before the addresses are given back to the application which initiated the request. Despite of the the policy, it is important that the application can control if address filtering is used or not.

Resolver modification is needed for HIP, too, because HITs are stored in DNS. Additionally, if HIT addresses are saved in other places, explicit support for retrieving them must also be implemented. By default, HIP host's resolver would prefer HIT addresses and return them be-

fore normal IPv6 addresses. Name resolvers must also have support for DNSSEC because HIP specifications require this for secure key retrieval. Application might also request only HIT addresses if it does not want to accept insecure connections using plain IPv6.

4.5 Example Program Using HIP

Next we present a simple example using C programming language which shows that converting an existing IPv6 application to use HIP is not necessarily hard and can be done with very small effort. Error handling in the program is not shown for clarity.

This example program uses a modified version of *getaddrinfo* function (see [BAS] for more information) for name resolution. *getaddrinfo* is given an additional flag (AI_HIP) which tells it to return only HIT addresses to the application. If this flag was not given, returned address depend on host policy. After name is found program execution is the same as it would have been using IPv6 addresses.

```
struct addrinfo hints;
struct addrinfo *res;
int fd;

hints.ai_flags = AI_HIP;
hints.ai_family = AF_INET6;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = 0;

getaddrinfo("hip.host.org", "echo", &hints, &res);
fd=socket(res->ai_family, res->ai_socktype,
          res->protocol);
connect(fd, res->ai_addr, res->ai_addrlen);
/* normal data transfer here: send(), recv(), .. */
close(fd);
freeaddrinfo(res);
```

However, there might be some problems using this approach. Assume that resolver query finds HIT address for a host, and initiator makes a new HIP connection to remote host. Now, if responder host does not support HIP, the connection will fail. Initiator must then decide whether it tries again insecurely without HIP or cancels the connection attempt.

4.6 Deployment Phase

The best way to start the HIP deployment is to make incremental steps. Usability of new protocols in the real world needs to be proven before larger scale deployment of the protocol will take place. Initial user experiences are critical when the protocol is considered to be taken into wider use.

One reason why transition to IPv6 has been slow is that IPv6 needs a lot of changes to the existing systems. Some older routers do not support IPv6, but only IPv4. Fully IPv6 based Internet requires that those old routers are changed to new ones. Upgrading creates increased costs, so infrastructure owners are not willing to upgrade unless they have to.

HIP does not need as much drastic changes if the infrastrucure for IPv6 exists already. Because HIP operates on top of internetworking layer, existing routing infrastructure can remain functional.

4.6.1 *Small Scale Deployment.* Getting initial HIP implementations to work will require a considerable amount of technical knowledge. Typical testing requires installation of kernel patches and complex manual configuration phases. The main objective in this state is to provide functionality, not usability. Therefore HIP will initially most probably be adopted by small technologically orientated persons, such as security enthusiasts.

Large organizations might wait for these initial user experiences and see whether HIP would be usable within their organization, perhaps by setting up a small test network which is not connected to the production level systems.

DNS server configuration is not necessarily needed, hosts could use static tables for addresses such as */etc/hosts* file on UNIX like operating systems.

4.6.2 *Larger Scale Deployment.* Larger scale deployment is not possible before there are full HIP implementations for many common operating systems and HIP has proven its usability in small networks. Large organization might set up IPv6 tunnels between its different networks and prefer HIP if previous experiences proved to be beneficial.

DNS configuration is needed to support connections from hosts from other networks if organization decides to place HIP hosts into public network. Rendezvouz server needs to be set up for mobile users. Network administrators need to define policies on using HIP.

Busy HIP hosts may end up with scalability related problems due to performance issues caused by increased amount of encrypting and decrypting traffic and handling the base exchange. This must be noted when HIP is considered to be used in large public networks where the number of potential HIP clients is high.

5. CONCLUSIONS

HIP seems to be a promising protocol which provides many advanced functionalities with only adding a new simple namespace and a simple protocol. Different reprentations of Host Identities eases their applicability to different protocols, as we have shown the case with IPv6. HIP protocol provides mobility, multihoming, authentication and security in a clean manner, thus making some other existing technologies into lesser use or even obsolete. Advantages of HIP clearly outstand the disadvantages.

Transition phase from IPv6 to IPv6+HIP is not as hard as it would seem to be at the first sight. Current applications may need to be patched for HIP support, but that will be a straightforward task to do.

By the time of writing, several HIP implementations have been started and the specifications are continuously reviewed and improved. HIP will most probably be one of most interesting development areas in computer networks in the near future.

REFERENCES

[ADDRARCH] Deering S., Hinden R., *RFC2373: IP Version 6 Addressing Architecture*, July 1998, URL: ftp://ftp.funet.fi/rfc/rfc2373.txt

[ARCH] Moskowitz, R., *Host Identity Payload Architecture*, work in progress, an Internet-Draft, January 2001, URL: http://homebase.htt-consult.com/~hip/draft-moskowitz-hip-arch-02.txt

[BAS] Bound J., Gilligan R. E., McCann J., Stevens W., Thomson S., *Basic Socket Interface Extensions for IPv6*, September 2002, URL: ftp://ftp.funet.fi/internet-drafts/draft-ietf-ipngwg-rfc2553bis-07.txt

[DHCP] Droms R., *RFC2131: Dynamic Host Configuration Protocol*, March 1997, URL: ftp://ftp.funet.fi/rfc/rfc2131.txt

[DIFFIE] Rescorla E., *RFC2631: Diffie-Hellman Key Agreement Method*, June 1999, URL: ftp://ftp.funet.fi/rfc/rfc2631.txt

[DNS] Mockapetris P., *RFC882: DOMAIN NAMES - CONCEPTS and FACILITIES*, November 1983, URL: ftp://ftp.funet.fi/rfc/rfc882.txt

[DNSSEC] Eastlake D., *RFC2535: Domain Name System Security Extensions*, March 1999, URL: ftp://ftp.funet.fi/rfc/rfc2535.txt

[ESP] Atkinson R., Kent S., *RFC2406: IP Encapsulating Security Payload (ESP)*, November 1998, URL: ftp://ftp.funet.fi/rfc/rfc2406.txt

[HIPIMP] Moskowitz, R., *Host Identity Payload Implementation*, work in progress, an Internet-Draft, January 2001, URL: http://homebase.htt-consult.com/~hip/draft-moskowitz-hip-impl-01.txt

[ICMP] Postel J., *RFC792: Internet Control Message Protocol*, September 1981, URL: ftp://ftp.funet.fi/rfc/rfc792.txt

[IKE] Carrel D., Harkins D., *RFC2409: The Internet Key Exchange (IKE)*, November 1998, URL: ftp://ftp.funet.fi/rfc/rfc2409.txt

[KOMU] Komu M., *Host Identity Payload in Home Networks*, April 2002

[LDAP] Howes T., Kille S., Wahl M., *RFC2251: Lightweight Directory Access Protocol (v3)*, December 1997, URL: ftp://ftp.funet.fi/rfc/rfc2251.txt

[MOS] Moskowitz, R., *Host Identity Payload And Protocol*, work in progress, an Internet-Draft, October 2001, URL: http://homebase.htt-consult.com/~hip/draft-moskowitz-hip-05.txt

[STE] Stevens W. R., *UNIX Network programming*, Prentice-Hall, 1997, ISBN-0-13-4900012-X

[TCP] Postel J., *RFC793: Transmission Control Protocol*, September 1981, URL: ftp://ftp.funet.fi/rfc/rfc793.txt

[TRA] Gilligan R. E., Nordmark E., *Transition Mechanisms for IPv6 Hosts and Routers*, work in progress, an Internet-Draft, July 17 2002, URL: ftp://ftp.funet.fi/internet-drafts/draft-ietf-ngtrans-mech-v2-01.txt

[UDP] Postel J., *RFC768: User Datagram Protocol*, 28 August 1980, URL: ftp://ftp.funet.fi/rfc/rfc768.txt

[V6OPS] Savola P., *Firewalling Considerations for IPv6*, work in progress, an Internet-Draft, September 2002, URL: http://www.ietf.org/internet-drafts/draft-savola-v6ops-firewalling-00.txt