Publication VI

G. Camarillo, I. Más Ivars, and P. Nikander. A Framework to Combine the Session Initiation Protocol and the Host Identity Protocol. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Pages 3051-3056, May 2008.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2008 proceedings.

A Framework to Combine the Session Initiation Protocol and the Host Identity Protocol

Gonzalo Camarillo, Ignacio Más and Pekka Nikander
Ericsson Research
Email:{gonzalo.camarillo, ignacio.mas.ivars, pekka.nikander}@ericsson.com

***Abstract*—The Session Initiation Protocol (SIP) is an application-layer protocol to establish and manage sessions. SIP provides user mobility by having user agents register their location with a server in the network. The Host Identity Protocol (HIP) is a shim layer between the network and the transport layers to establish and manage secure connections between hosts which may be mobile. In this paper, we propose a framework to combine SIP and HIP, and discuss the advantages of doing so. The advantages this framework provides relate to security, mobility, and multihoming. Additionally, we discuss how a P2P SIP (peer-to-peer SIP) system can be implemented on top of a HIP-based overlay network and the characteristics of such a system. We also offer some insights into an experimental prototype implemented as a proof-of-concept of the SIP-HIP interaction.**

***Index Terms*—SIP, HIP, P2P, P2PSIP**

I. Introduction

The Session Initiation Protocol (SIP) [1] is an application-layer rendezvous protocol that has been designed with flexibility, good scalability, and ease of implementation in mind. SIP is used for creating, modifying, and terminating sessions with many participants. A session is considered to be a set of senders and receivers that communicate with all their associated state information. SIP allows locating users and negotiating the parameters needed to establish sessions with them. Users are identified by SIP URIs, whose format is similar to the email address format (i.e., user@domain). SIP requires extra protocols to transport the multimedia data itself, such as the Real-time Transport Protocol (RTP) [2], and to describe and encode capabilities of session participants, such as the Session Description Protocol (SDP) [3]. SIP is loosely based on HTTP, which is arguably the most successful and widely used protocol on the Internet and has been designed in conformance with the end-to-end principle of system design: all the logic and state is stored in the end devices, which offers no single point of failure and scales well [4].

The Host Identity Protocol (HIP) [5] proposes a new layer in the TCP/IP protocol stack. This new layer is located below the transport layer and consists of cryptographically generated host identifiers. HIP provides opportunistic host authentication and end-to-end security, host mobility, and address multihoming. In HIP, each host is identified by a host identity tag (HIT), which is a hash of a public key. Before exchanging any user traffic, hosts perform a HIP exchange consisting of four messages. This exchange, which is referred to as the base HIP exchange, allows the hosts involved in the communication to prove to each other that they hold the private key corresponding to their respective HITs. After completing a base exchange, hosts exchange user traffic, which is protected with IPsec.

Upper-layer protocols such as TCP use HITs to identify upper-layer connections. HITs do not change throughout the duration of the connection, but the IP addresses associated with the HITs are allowed to change. This way, IP address changes due to mobility or multihoming are transparent to the upper-layer application, which only relies on the HITs to identify connections. In order to initiate a base HIP exchange with another host, a given host needs, in addition to the HIT of the remote host, an IP address to send its initial HIP packet to. Service identifiers, such as HTTP URIs, may be resolved into a HIT and an IP address using the Domain Name System (DNS). Once the service identifier is resolved, the hosts can perform the base HIP exchange.

The remainder of the paper is organized as follows: Section II describes the basic principles of the framework to combine SIP and HIP, and describes the mobility, security, and multihoming related properties of the framework. This section also describes how the framework applies to P2P SIP. Section III gives insights into our implementation experience. Finally, Section IV contains the conclusions of the paper.

II. Framework Architecture

A framework that integrates HIP and SIP needs to tackle both session (i.e., media) and SIP traffic. Figure 1 shows how both types of traffic are typically routed between endpoints. Session traffic is generally sent end-to-end while SIP traffic may traverse SIP proxies located between the endpoints. HIP is used to manage all connections between different entities. Sections II-A, II-B, and II-C deal with the mobility, security, and multihoming related properties of this framework respectively. Section II-D described how this framework applies to P2P SIP.

A. Mobility-related Properties

Both SIP and HIP can perform mobility management. However, rather than providing overlapping functionality, they provide complementary types of mobility management. We distinguish three types of mobility: user, session, and host mobility. User mobility provides a user with the ability to be reachable under the same identifier regardless of the user's location. Session mobility involves the transfer of a ongoing session between different devices. Host mobility occurs when

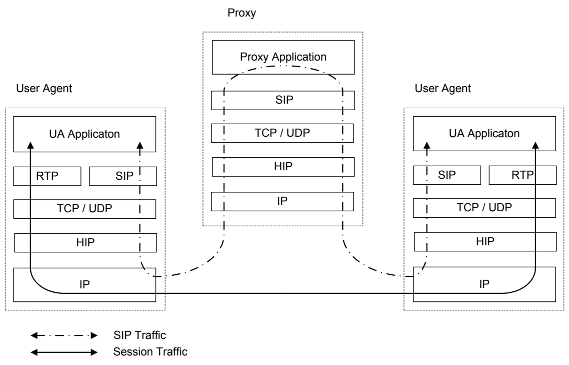

Fig. 1. Framework Architecture

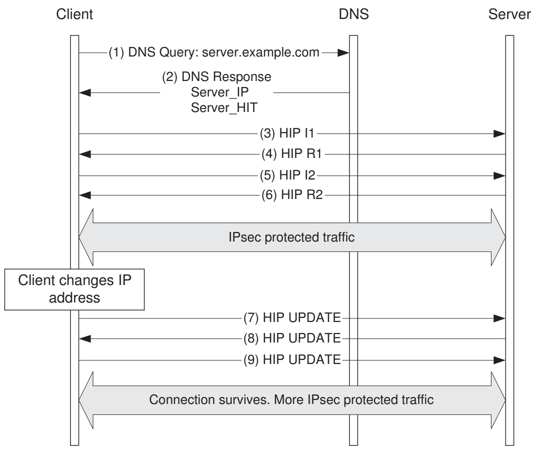
Client
DNS
Server
(1) DNS Query: server.example.com
(2) DNS Response Server_IP Server_HIT
(3) HIP I1
(4) HIP R1
(5) HIP I2
(6) HIP R2
IPsec protected traffic
Client changes IP address
(7) HIP UPDATE
(8) HIP UPDATE
(9) HIP UPDATE
Connection survives. More IPsec protected traffic

Fig. 2. Client-to-server communication using HIP

a host changes its IP address. We propose to use SIP to provide user and session mobility and to use HIP to provide host mobility.

HIP provides end-to-end host mobility. Figure 2 shows how a SIP client establishes a HIP connection with a SIP server. The client initially knows the server's fully qualified domain name (FQDN) because it appears in the URI to which the client will be sending SIP traffic. In order to obtain the HIT and the IP address of the server, the client queries the DNS (1), which provides the client with these data (2). At this point, the client initiates the base HIP exchange with the server (3-6) using the HIT and the IP address obtained from the DNS. At a later point, if the client moves to a new IP address, it performs a new HIP exchange (this time consisting of three messages [6]) with the server (7-9) using the same HIT and the same server's IP address as before. This new exchange informs the server that the client has moved to a new IP address. The TCP connection survives the IP address change because the HITs, which are used to identify the TCP connection, do not change and the TCP socket is bound to the HITs instead of to the IP addresses of the hosts.

Media stream mobility is handled in the same way. The only difference is that the client obtains the server's HIT and IP address through an SDP [3] offer/answer exchange [7], instead of by using the DNS. In [8], we propose an extension to encode HITs in SDP. A comparison of the efficiency achieved by implementing mobility in SIP using mobile IP (which is an alternative mechanism to implement host mobility) and HIP can be found at [9].

HIP mobility works well when a host or device is moving to a new location. However, when a user moves a session to a new device, the new device will have a different public-private key pair than the old device, and so, HIP cannot tackle mobility between them. This type of session mobility is better tackled by SIP.

A SIP user can move to a new endpoint without losing ongoing sessions. Users can use re-INVITEs or UPDATE requests to change endpoints without changing the SIP proxies in the path, or use the Replaces mechanism [10], which is a SIP extension in the form of a new header field, to establish a brand new session to replace the old one. The former mechanism has limited applicability to mobility (it is more often used to change the characteristics of a session without changing the endpoint handing the session) while the latter is more general.

SIP also provides user mobility. Users register their location with a SIP registrar. This way, a particular user is always reachable under the same identifier regardless of the user's location. This identifier can be an AoR (address of records), which is the publicly-known SIP URI at which a user is reachable.

Therefore, using SIP and HIP together allows providing a mobility management framework that includes host, user, and session mobility.

B. Security-related Properties

This section analyzes the security properties of this framework. The use of HIP resolves some issues related to denial of service (DoS) attacks that are present when plain SIP is used.

1) Denial of Service Prevention: Session establishment in SIP follows the offer/answer model [7]. The offerer generates an offer that contains, among other things, the IP address the answerer will be sending media to. The offer/answer model can be used to perform denial of service attacks against third parties. The offerer generates an offer with the IP address of the victim and the answerer, on reception of such offer, starts sending media to the victim. (The answerer can perform this attack as well.)

If the session consists of media sent over a connection-oriented transport protocol such as TCP, the victim does not respond to the unknown and unexpected connection establishment request (e.g., the initial SYN in TCP) and the connection is not established. However, if the session consists of media sent over RTP and UDP, the sender just overloads the victim with RTP packets (this attack is commonly referred to as the media hammering attack). The ICMP (non reachable) messages generated by the victim may or may not stop the attack. Firewalls in the path may discard them or the RTP library of the sender may simply ignore them. Our experiments indicate that some RTP libraries do actually ignore ICMP messages.

We have performed a set of experiments to test the DoS threat to different RTP stacks by having a VoIP SIP user agent send unsolicited RTP packets to a victim. We repeated the same experiment using two different victims: a NetBSD machine and a Linux machine. Both of them started generating ICMP (Destination Unreachable) packets in response to the attack. However, the rate at which these ICMP packets were generated was different in the two operating systems. While NetBSD generated an ICMP packet in response to every incoming RTP packet, Linux generated an ICMP packet for each incoming RTP packet only for the first six RTP packets. Afterwards, it only generated one ICMP packet per second.

The RTP library generating traffic on the VoIP client did not perform any action on reception of the ICMP messages. It continued flooding the victim with traffic. This behavior is typically justified because ICMP messages cannot be authenticated. Therefore, if an RTP library stopped sending traffic on reception of (unauthenticated) ICMP messages, it would be trivial to perform DoS attacks against such a library. Nevertheless, in this case this feature makes it impossible for the victim to stop the attack. Additionally, even though the RTP library never received any RTCP (Real-time Transport Control Protocol) [2] message from the remote endpoint, it did not stop generating traffic until we manually stopped the experiment.

The use of HIP prevents this type of attack because the victim does not accept the initial incoming HIP message. Additionally, a victim that implemented HIP and accepted the HIP connection for some reason could close it (using the CLOSE message) at any point.

Note that HIP provides strong cryptographically-based protection against this attack; the receiver of the media stream is securely identified before sending any media. Other protection mechanisms against this attack, such as the use of STUN [11] connectivity checks before sending media, provide a weaker protection.

A similar attack to the one just described can be performed using SIP third-party registrations. The attacker registers the victim's IP address as a new contact for the attacker's AoR. The attacker then ensures that a high number of SIP requests (e.g., a large number of instant messages) are sent to the attacker's AoR, all of which are routed to the victim's IP address. The use of HIP to protect SIP traffic prevents this attack as well.

2) Hop-by-Hop Security: SIP uses TLS (Transport Layer Security) [12] to provide hop-by-hop security. Depending on the SIP entities involved in a particular TLS connection, authentication is performed in different ways. In a TLS connection between a user agent and its outbound proxy, the outbound proxy typically provides its server certificate while the user agent is authenticated using digest authentication over the TLS connection [13]. This mechanism is used because user agents usually share a secret with their outbound proxy but do not have client certificates. Authentication in TLS connections between proxies is generally performed using certificates within the TLS connection establishment procedure. Once both ends of the TLS connection have been authenticated, they exchange traffic which is encrypted and integrity protected.

If a user agent wants to force proxies handling its SIP messages to always send them over TLS connections, it addresses its message to a SIPS URI. Proxies handling SIPS URIs always use TLS to send and receive SIP messages. This way, only the proxy servers routing SIP message sent to a SIPS URI have access to its contents. When SIP hop-by-hop security was designed two security mechanisms were analyzed: TLS and IPsec. The main reason why TLS was finally chosen over IPsec was that applications can easily control that a particular message is sent over a TLS connection. The application simply specifies the use of TLS when opening a transport socket (e.g., a TCP socket). However, ensuring that a particular message is sent over an IPsec connection is typically difficult and may require modifications to the kernel of the operating system. Applications do not typically have a way to access the security database that stores the policies related to the security associations the host has established or will establish. As a consequence, applications are typically unaware of whether or not their traffic is being protected by IPsec.

We propose to use HIP as a hook for SIP applications to establish IPsec connections with its peers. The fact that an application sends a message to a HIT, instead of sending it to an IP address, implies that the message will be IPsec protected. Therefore, all SIP traffic will be protected, not only requests (and their corresponding responses) sent to SIPS URIs, which mandate TLS protection. Consequently, even if a SIP request is addressed to a SIPS URI, a proxy can choose to send it forward using HIP instead of TLS. Proxies obtain the HIT and the IP address of a message's next hop by querying the DNS. Enabling proxies to handle HITs only requires minor modifications to the way SIP entities locate the next hop for a given message [14].

Like TLS, HIP supports certificate-based authentication. The server can include a CER attribute (a HIP attribute that can carry certificates) with its certificate in the R1 packet of the base HIP exchange. The client can also use a CER attribute to send its certificate in the I2 packet. So, HIP can be used in both, user-agent-to-proxy and proxy-to-proxy scenarios.

3) Alternative Means of Protection: Certain types of sessions are better protected using mechanisms other than IPsec, which is the mechanism HIP uses. An example of such a session type is a session consisting on real-time traffic transported using the Real-Time Transport Protocol (RTP) [2]. When RTP is used over low-bandwidth links, header compression mechanisms such as ROHC (Robust Header Compression) are typically employed between the user agent and its access router [15]. RTP headers are sent over the link in a compressed format in order to achieve a faster transmission (bandwidth savings are usually seen as just a positive side effect). At the end of the link, the RTP headers are uncompressed and passed to the application in their regular format.

Therefore, access routers implementing the compressor/decompressor functions need to have access to the RTP headers in order to compress them and uncompress them. At the same time, users sometimes want to have the contents of the RTP packets secret by encrypting them. SRTP (Secure RTP) [16] makes it possible to use header compression techniques and RTP-payload encryption simultaneously. SRTP allows users to encrypt the contents of their RTP packets (e.g., encoded voice) without encrypting the RTP headers. SRTP specifies an optional trailer to the RTP packets, which carries keying and authentication material. When only encryption is turned on (i.e., no authentication) it is possible not to use the trailer at all. In this case, SRTP does not introduce any overhead over plain RTP.

IPsec, however, encrypts RTP headers, which makes it impossible to use RTP header compression in conjunction with IPsec. In this way, IPsec is not as bandwidth-efficient as SRTP, which can take advantage of RTP header compression. This makes transmissions over low-bandwidth links slower. Therefore, RTP sessions are not a good candidate to use HIP, given that HIP traffic is always protected using IPsec. Nevertheless, the mobility and multihoming capabilities of HIP might be useful for this type of real-time traffic. Therefore, being able to use HIP without its implicit IPsec encryption could prove useful in scenarios that involve real-time sessions. Our future work includes studying the use of alternative security mechanisms for HIP.

C. Multihoming-related Properties

A SIP registration binds a user's AoR with the user's current location or locations (i.e., a user can be available at several user agents). It is important that the user is actually available at the registered locations because, otherwise, no SIP traffic will be delivered to the user.

In order to improve the reliability of SIP systems, user agents can be multihomed. A particular user agent can, for example, be reachable at two different IP addresses, each of which is fetched from a different service provider. This ensures that traffic will always reach the user agent.

The multihoming capabilities of HIP allow SIP user agents in this framework to be reachable at several alternative IP addresses at the same time. Note that this is a different property than the ones described in [17], which relate to multihomed edge proxies and registrars.

D. Peer-to-peer SIP

P2P SIP (peer-to-peer SIP) allows building SIP networks without using a dedicated SIP network infrastructure. The users' registration information (which includes the users' current location) is not stored in dedicated servers in the network. Instead, it is stored in a distributed manner among all the user agents in the network.

P2P SIP systems use DHTs (Distributed Hash Tables) to build such a distributed registration database. A DHT allows the nodes forming the DHT to store objects in a distributed fashion. The set of objects each node stores is determined by the algorithm defining the DHT. In any case, DHTs ensure that objects are evenly distributed among its nodes. That is, each node in a DHT is required to store roughly the same amount of information.

Objects stored in a DHT are accessed via keys. Each object has an associated key and nodes can search for that key in the DHT. In P2P SIP, an object stores the registration information of a user (e.g., the user's current IP address); the object's key is the hash of the user's AoR.

For example, a user agent that needs to establish a session with 'sip:bob@example.com' would calculate the hash (e.g., using SHA1) of 'sip:bob@example.com' and perform a search for the resulting hash in the DHT. The DHT would return the contact information (e.g., an IP address) needed to contact 'sip:bob@example.com'. The user agent uses this contact information to send a SIP INVITE request in order to establish the session.

Note that, effectively, P2P SIP replaces the DNS-based URI resolution function used in traditional SIP [14] with a new DHT-based function. That is why the user agent in the example does not query the DNS to resolve 'sip:bob@example.com'. Instead, it queries the DHT. (How to differentiate URIs that need to be resolved using the DNS or a DHT is part of our future work.)

Nodes in a DHT are identified by their peer IDs. This framework proposes to use the node's HIT as its peer ID. Using HIP to build overlay networks where nodes are identified by their HITs has been studied in [18] and [19].

Using HITs as peer IDs has set of associated advantages. Connection between nodes are handled by HIP, which provides security, mobility management, and multihoming to those connections. HIP also provides a secure binding between users' identities (i.e., their public keys) and their peer IDs. This binding allows nodes to securely prove they own a given peer ID.

Another advantage of using HIP in this context is that P2P SIP can be easily integrated into legacy SIP applications (i.e., SIP applications that do not support P2P SIP), shortening the P2P SIP developing time. IPv6 addresses and HITs have the same format [20]. This allows applications to use the same traditional socket APIs to establish HIP connections as they used to establish regular connections before. HIP handles those connections at a layer below transport and, thus, does not require applications to be modified. Overall, our implementation experience shows that a given legacy SIP application needs to be modified to a lesser degree in order to support P2P SIP when HIP is used than when similar functionality is implemented directly at the application layer (assuming that a HIP library is available). At present, there already are available HIP libraries for all major operating systems.

HIP-based overlay networks can also be used by applications other than P2P SIP. This makes HIP-based overlay networks attractive in multiapplication environments.

III. Implementation Experience

We implemented a proof-of-concept prototype in order to analyze the delay introduced by the use of HIP in a SIP architecture as proposed in this paper (see Figure 1). Our prototype allowed us to turn HIP support on and off in order to study the delay penalty introduced by HIP. This section presents our preliminary results. Our future work includes performing a large number of measures and a thorough statistical delay analysis.

Our test bed consisted of two SIP user agents and a registrar that also acted as a proxy. They were all set up on the same IPv6 subnet with very small round trip times between them (in the order of 2 ms). This set up is appropriate because the focus of the experiments was set on the processing delays introduced by HIP. Extrapolating our results to scenarios with longer round trip times should be trivial. All the nodes in our test bed were Dell Latitude D610 laptops with Intel Pentium Mobile at 1.86 GHz processors and 512 MB of RAM running Linux (2.6 kernel). The SIP user agents were based on the Minisip user agent (we modified minisip so that it could run on an IPv6 network). The SIP registrar and proxy was a SER (SIP Express Router) server. Our HIP implementation, consisting of around 25000 lines of C code, was installed in the three nodes. The network only carried traffic related to our experiments (i.e., it was empty otherwise) and the hosts did not run any other application besides our prototype implementation.

Two experiments were performed. Figures 3 and 4 show their call flows. Both experiments were first run as shown in the figures and then without using HIP in order to analyze the performance penalty introduced by HIP. UDP was the transport protocol used in all the experiments.

Figure 3 shows a SIP user agent registering at its registrar and receiving a MESSAGE request through the registrar, which acts as a proxy. The attempt by the user agent to send the REGISTER request triggers the HIP base exchange with the registrar. The size of the I1, R1, I2, and R2 messages of the exchange are 94, 654, 718, and 270 bytes respectively. The whole exchange took 250.3 ms to complete. The difficulty of the cryptographic puzzle in R1 was set to 10 and it took 105.9 ms for the user agent to resolve it (a puzzle of such difficulty requires an average of 512 SHA-1 operations to be resolved).

Once the HIP base exchange completed, the IPsec-protected REGISTER (5) was sent to the registrar. The total size of the frames (including link-layer headers) carrying the IPsec-protected REGISTER request and its 200 (OK) response was 1316 bytes, which is a 13.5 % higher than the same SIP transaction with no IPsec protection. It took 2.7 ms for the user agent to process an incoming IPsec-protected MESSAGE request (8). In comparison, it took 2.5 ms to process the same MESSAGE request with no IPsec protection. The total size of the frames (including link-layer headers) carrying the IPsec-protected MESSAGE transaction was 1492 bytes, which is an 11.9 % higher than the same SIP transaction with no IPsec protection.

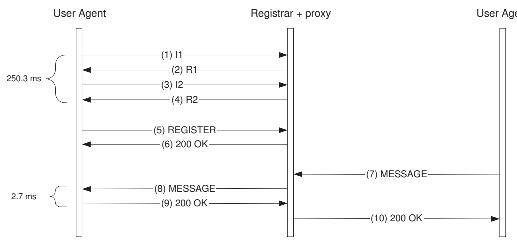

Fig. 3. SIP REGISTER and MESSAGE over HIP

Figure 4 shows the user agent that registered in Figure 3 processing an incoming IPsec-protected INVITE request. It took 317.7 ms for the user agent to perform the HIP base exchange with the remote user agent and to generate a 180 (Ringing) response to the INVITE. In comparison, it took 179.8 ms to generate the 180 (Ringing) when HIP was not used.

Our HIP stack typically initiates the HIP base exchange when the application sends data to a new destination. In this case, however, the HIP base exchange was performed before the user agent sent any data to the remote user agent. This is because Minisip uses connected UDP sockets. The user agent connected the socket on receiving the INVITE request in order to be ready to start sending media as soon as the user accepted the incoming call. Consequently, when the user agent started sending media packets to the remote user agent, the HIP base exchange was already done and no additional delay or media clipping was introduced by the use of HIP. The total size of the frames (including link-layer headers) carrying the IPsec-protected INVITE transaction (excluding the HIP base exchange) was 3688 bytes, which is a 10.4 % higher than the same SIP transaction with no IPsec protection. The total size of the frames (including link-layer headers) carrying audio encoded using the 8000 Hz ITU-T G.711 PCMU codec was 282 bytes, which is a 20.5 % higher than the same media packet with no IPsec protection. Table I summarizes the increases in packet size due to IPsec for different traffic types.

TABLE I
Packet size increase due to IPsec

Traffic Type	Overhead
REGISTER	13.5%
MESSAGE	11.9%
INVITE	10.4%
Media	20.5%

IV. Conclusions

In this paper, we have presented a framework to combine SIP and HIP. SIP-based services between HIP-enabled hosts

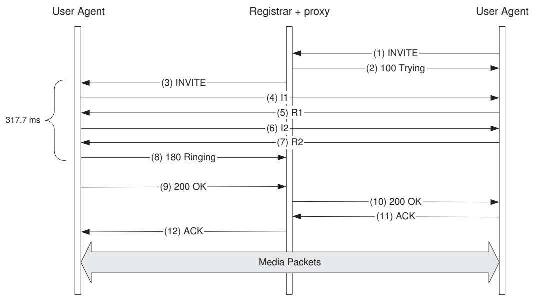

Fig. 4. SIP INVITE over HIP

can take advantage of the mobility, security, and multihoming capabilities of HIP. Additionally, some SIP-based services, although not all, benefit from the IPsec-based protection provided by HIP. In particular, user agents using RTP header compression may find alternative ways of securing traffic such as SRTP more convenient.

In our framework, HIP is used between entities of a SIP infrastructure as the transport mechanism. Thus, SIP hop-by-hop SIP security is extended to use IPsec, in addition to TLS, by using HIP to negotiate the security properties of the connections between SIP entities.

Using HITs as peer IDs in P2P SIP offers important advantages. In addition to providing a secure binding between user identities and peer IDs, the use of HIP can shorten development times for P2P SIP systems.

We have implemented a proof-of-concept prototype that demonstrates the concepts described in this article and offers some highlights on the different overheads caused by combining SIP and HIP together. These overheads, when weighted against the advantages the use of HIP provides, do not seem to represent any significant limitation for this framework.

REFERENCES

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," Internet Engineering Task Force, RFC 3261, Jun. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3261.txt

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, RFC 3550, Jul. 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3550.txt

[3] M. Handley, V. Jacobson, and C. Perkins, "SDP: Session Description Protocol," Internet Engineering Task Force, RFC 4566, Jul. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4566.txt

[4] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp. 277–288, November 1984.

[5] R. Moskowitz, "Host Identity Protocol," Internet Engineering Task Force, Internet-Draft draft-ietf-hip-base-06, Jun. 2006, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-hip-base-06.txt

[6] P. Nikander, "End-Host Mobility and Multihoming with the Host Identity Protocol," Internet Engineering Task Force, Internet-Draft draft-ietf-hip-mm-03, Mar. 2006, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-hip-mm-03.txt

[7] J. Rosenberg and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)," Internet Engineering Task Force, RFC 3264, Jun. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3264.txt

[8] H. Tschofenig, "Interaction between SIP and HIP," Internet Engineering Task Force, Internet-Draft draft-tschofenig-hiprg-host-identities-03, Mar. 2006, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-tschofenig-hiprg-host-identities-03.txt

[9] J. Y. So, J. Wang, and D. Jones, "SHIP mobility management hybrid SIP-HIP scheme," *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005*, 2005.

[10] R. Mahy, B. Biggs, and R. Dean, "The Session Initiation Protocol (SIP) Replaces Header," Internet Engineering Task Force, RFC 3891, Sep. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3891.txt

[11] J. Rosenberg, "Simple Traversal of UDP Through Network Address Translators (NAT) (STUN)," Internet Engineering Task Force, Internet-Draft draft-ietf-behave-rfc3489bis-03, Mar. 2006, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-behave-rfc3489bis-03.txt

[12] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," Internet Engineering Task Force, RFC 2246, Jan. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2246.txt

[13] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," Internet Engineering Task Force, RFC 2617, Jun. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2617.txt

[14] J. Rosenberg and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers," Internet Engineering Task Force, RFC 3263, Jun. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3263.txt

[15] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," Internet Engineering Task Force, RFC 3095, Jul. 2001. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3095.txt

[16] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," Internet Engineering Task Force, RFC 3711, Mar. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3711.txt

[17] C. Jennings and R. Mahy, "Managing Client Initiated Connections in the Session Initiation Protocol (SIP)," Internet Engineering Task Force, Internet-Draft draft-ietf-sip-outbound-03, Mar. 2006, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-sip-outbound-03.txt

[18] P. Nikander, J. Arkko, and B. Ohlman, "Host identity indirection infrastructure," *in Proceedings of The Second Swedish National Computer Networking Workshop 2004 (SNCNW2004), Karlstad University, Karlstad, Sweden*, November 2004.

[19] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme, "A node identity internetworking architecture," *in Proceedings of the 25th IEEE International Conference on Computer Communications, INFOCOM 2006*, April 2006.

[20] P. Nikander, J. Laganier, and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)," Internet Engineering Task Force, RFC 4843, Apr. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4843.txt