Publication I

G. Camarillo, T. Kauppinen, M. Kuparinen, and I. Más Ivar. Towards an Innovation Oriented IP Multimedia Subsystem. *IEEE Communications Magazine*, Vol. 45, No. 3, Pages 130-136, March 2007.

Towards an Innovation Oriented IP Multimedia Subsystem

Gonzalo Camarillo, Tero Kauppinen, Martti Kuparinen, and Ignacio Más Ivars, Ericsson Research.

ABSTRACT

This article proposes an approach to IMS policy control based on session policies that achieve transparent end-to-end session establishments between IMS terminals. The article identifies drawbacks in the current IMS policy control methodology, discusses how these drawbacks may negatively influence the potential of IMS to provide innovative services, and describes how the new approach overcomes these drawbacks. Our proposal offers modularity and scalability properties that enable operators to establish policies and modify existing ones without major changes in the IMS core. Thus, policies can be applied transparently to SIP dialogs between terminals and modified on the fly without tearing down ongoing dialogs. The article also discusses a test bed implementation that worked as a proof-of-concept.

INTRODUCTION

The Session Initiation Protocol (SIP) [1] is a text-based rendezvous protocol that provides user mobility and session establishment. SIP user mobility is based on registrations. Users register their current locations with the network, which uses this registration information to route incoming session requests to those users.

SIP session establishment is based on the offer/answer model [2], which is a two-way session description exchange between two SIP user agents. User agents involved in an offer/answer exchange obtain all the information required to establish a session between them.

The IP multimedia subsystem (IMS) is system architecture whose main signaling protocol is SIP. The goal of IMS is to provide service providers with a platform that facilitates the provision and management of a wide range of services. The success of service providers using IMS and consequently, the success of IMS as a whole, depends on how appealing those IMS services are to users.

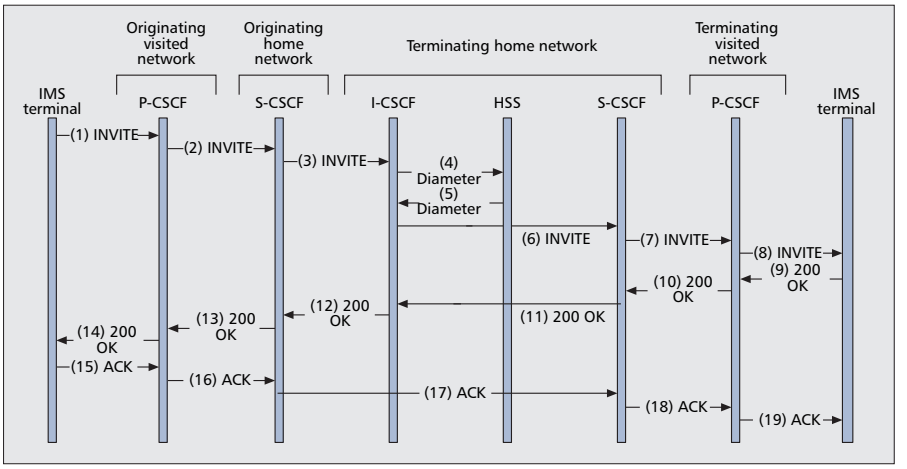
To make IMS services more appealing than services developed for other platforms, IMS enables service developers to integrate their services horizontally. Horizontal service integration is believed to produce faster service development times than the traditional vertical service integration, where a stand-alone module provides all of the functionality required by a particular service. A horizontally integrated service consists of a set of functions that work together to provide the functionality expected from that particular service. Given that many of these functions (e.g., user authentication) are common across different services, the same function can be reused by several services.

IMS provides these common functions and makes them available to any service built on top of it. This way, service developers are not required to re-implement these functions for each new service. Instead, they can simply focus on implementing the service logic specific to each particular service and rely on the common functionality provided by IMS. Common functions provided by IMS that are available to services built on top of it include capability negotiation, authentication, service invocation, addressing, routing, group management, presence, provisioning, session establishment, and charging.

However, to be successful, IMS must go beyond simply providing a fast service development environment. IMS also must foster the creation of innovative services. This article describes an alternative approach to IMS policy control that aims to foster innovation. Innovation is, arguably, the key to creating successful services that appeal to users. The approach removes the requirement to inspect the SDP (Session Description Protocol [3]) in the network nodes, which can be costly. In this way, new SDP extensions can be used to provide extended functionality and innovative services without incurring high network upgrade costs and delays.

The remainder of this article is organized as follows. We describe IMS session establishment, policy control implementation in IMS, and issues related to the IMS approach to policy control. We propose an alternative approach to policy control, referred to as session policies, which resolves those issues. We describe our experiences implementing session policies in an IMS environment. We outline our future work in this area.

 0163-6804/07/$20.00 © 2007 IEEE



■ **Figure 1.** *IMS session establishment.*

IMS Session Establishment

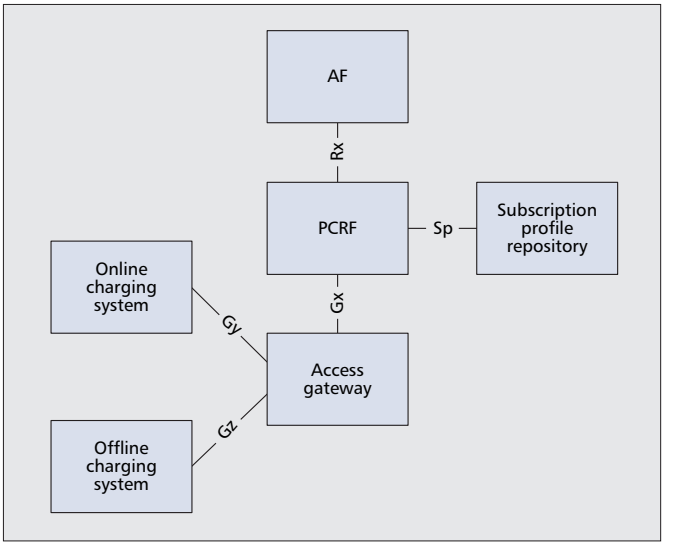
As stated previously, IMS session establishment is based on SIP. SIP provides session establishment through a two-way session description exchange called the offer/answer model [2]. A user agent generates a session description (the offer) that contains the information required to establish the session (e.g., IP addresses to transfer the media) and sends it to the remote user agent.

On receiving the offer, the remote user agent generates its own session description, which is referred to as the answer. Both the offer and the answer are written in a session description format that must be understood by both user agents. The default session description format is SDP. After the offer/answer exchange completes, the user agents can start exchanging media between them. At that point, the session is considered to be established.

Offer/answer exchanges can be mapped to a SIP three-way handshake (INVITE — 200 OK — ACK) in two ways. The INVITE carries the offer, and the 200 (OK) carries the answer, or the 200 (OK) carries the offer, and the ACK carries the answer. IMS supports both mappings to establish sessions.

Figure 1 shows a basic session establishment in IMS. Note that this session establishment does not use the optional preconditions extension [4] that is typically applied to basic session establishments. Its use is not relevant for this discussion and would add extra complexity to the message flow.

The flow in Fig. 1 shows two roaming terminals establishing a session between them. Both P-CSCFs (proxy-call/state control functions) are located in the visited domains. S-CSCFs are always located in the home domain of the users they serve. The network elements in Fig. 1 that relate to the offer/answer model are the P-CSCF (proxy-CSCF) and the S-CSCF (serving-CSCF). Both have access to all the offers and answers exchanged by the terminals.



■ **Figure 2.** *IMS policy and charging control architecture.*

Policy Control in IMS

IMS provides policy control. IMS domains can specify the types of media streams that are accepted (e.g., voice streams) and the types that are not (e.g., video streams). It also is possible to specify the sessions that are accepted based on characteristics of their media streams, such as the codecs used or the bandwidth requested for them.

IMS policy control and IMS charging control were implemented using separate architectures in 3GPP R5 (Release 5) and 3GPP R6. However, 3GPP R7 merged those architectures together. The policy and charging control (PCC) architecture defined in 3GGP R7 is the result of merging SBLP (service based local policy) and FBC (flow-based charging).

Figure 2 shows the 3GPP R7 policy and

A user cannot protect the integrity of or encrypt the session descriptions because the network may be required to modify them. Encrypted sessions are rejected by the network and integrity protection is broken by network nodes that are not actual attackers.

charging control architecture. Only the AF (application function), the PCRF (policy and charging rules function), and the access gateway are relevant to this discussion. The role of the AF can be performed by the P-CSCF or by an application server.

The PCRF receives information about the offer/answer exchanges between the terminals from the AF over the Rx interface. If the characteristics of the session being established are acceptable to the PCRF (based on the domain policy), the PCRF authorizes the session on the access gateway using the Gx interface. If the characteristics of the session are not acceptable to the PCRF, it instructs the AF to terminate the session using the Rx interface. Of course, in this case the PCRF does not authorize the session on the access gateway.

SIP and Policy Control

When the PCRF informs the AF that the session being established is not acceptable, the AF must inform the terminal trying to establish the session. The AF uses its SIP interface to do that. The way an AF informs a terminal that its session request is not acceptable depends on the mapping used between the offer/answer model and the SIP three-way handshake. For example, assume that the originating P-CSCF in Fig. 1 must inform the originating IMS terminal that its session request is not acceptable (the terminating P-CSCF would use identical mechanisms).

If the INVITE request (1) carried an offer, the P-CSCF would respond with a 488 (not acceptable here) response. This response would contain a session description of a session that would be acceptable. Such a session description could be used by the terminal to generate a new offer in a new INVITE request that would be acceptable for the network.

If the offer in the INVITE request (1) was acceptable, but the answer in the 200 (OK) response (13) was not, the P-CSCF would wait for the three-way handshake to finish. Immediately after that, it would generate two BYE requests, one to the originating terminal and one to the terminating terminal. In this case, the BYE request would not carry an indication of the type of session that would be acceptable to the network. Therefore, it would be difficult for the terminal to generate a new acceptable offer.

Additionally, the user behavior would be far from ideal. The session would be established and terminated immediately.

If the INVITE request (1) did not carry an offer, the 200 (OK) response would. If the offer in the 200 (OK) response or the answer in the ACK request (15) were not acceptable, the P-CSCF would behave as in the previous example. It would generate two BYE requests to terminate the session, with the same associated issues as before.

In addition to the behaviors just described, many actual deployments (generally using fixed access), have their P-CSCFs perform SDP rewriting. If a session description is not acceptable, the P-CSCF rewrites it before routing the message forward so that the resulting session description is acceptable. The P-CSCF can remove media streams, codecs, and so on.

SDP rewriting is also performed by P-CSCFs handling cellular terminals. The P-CSCF uses the single reservation flow (SRF) SDP extension to instruct the terminal about the PDP (packet data protocol) context to use for the media stream.

The characteristics of a session also can be unacceptable to one or both (originating and terminating) home domains. In this case, the S-CSCF performs the same actions as the P-CSCFs in the previous examples. The S-CSCF would use 488 (not acceptable here) responses or BYE requests to enforce its policy.

Issues with the Current Approach to Policy Control

There are issues with informing terminals about policy decisions in the ways described previously. As stated earlier, a terminal is not always informed why its session was rejected. The user of the terminal may not understand what to do to establish a new session that will be accepted. A user may also receive a successful-session-establishment indication and a session-terminated indication, one immediately after the other.

A user cannot protect the integrity of or encrypt the session descriptions because the network may be required to modify them. Encrypted sessions are rejected by the network and integrity protection is broken by network nodes that are not actual attackers. Additionally, a network cannot change the policies that apply to an ongoing session. For example, if a network decides that using video is no longer acceptable, it cannot inform the terminals about this fact. Its only option is to terminate a whole session by sending BYE requests to both terminals, and users will not know why the session terminated.

Although the previous issues that relate to user experience are important, this article focuses on issues affecting application and service developers. These issues affect whether IMS services can be innovative. Service developers currently are forced to use SDP as the only session description format. This is because network elements such as P-CSCFs and S-CSCFs require access to offers and answers.

New services cannot use SDP extensions that are not understood by the network because sessions using unknown extensions most likely will be rejected. Even if a user's home domain accepts sessions with unknown SDP extensions, other domains may not. This limits the users a user can communicate with to those in his or her domain.

Additionally, SDP rewriting can have unexpected interactions with extensions that are unknown to the entity performing the rewriting. SDP rewriting also makes it difficult for developers to debug their implementations because the network may modify their protocol messages without their knowledge.

This implies that service developers are limited to establishing sessions that can be described with SDP and the SDP extensions that are supported by the network. If the implementation of an innovative service requires the use of a new SDP extension or the use of a different session description format, the whole network would

require upgrading before this service could be provided. This would dramatically slow the introduction of such service and increase considerably the price to introduce it.

SESSION POLICIES

One of the design principles of SIP was to enable the creation of end-to-end services without requiring the upgrading of the network elements between endpoints. As discussed earlier, the current approach to IMS policy control breaks this principle.

Session policies provide a means for domains to communicate their policy to terminals and for terminals to provide domains with information about the sessions that they establish. There are two types of session policies: session-independent policies and session-specific policies. Session-independent policies are general policies that apply to all the sessions a terminal may attempt to establish. For example, a terminal may not be allowed to use video streams in its sessions. Session-specific policies only apply to a particular session. For example, a terminal may be required to group two of the session media streams (e.g., audio and video) into a single PDP context and transfer a third media stream over a different PDP context (e.g., instant messaging).

A domain typically requires information about the session being established by a terminal to provide it with session-specific policies for that session. The terminal informs the domain about the session and obtains the domain policies for that session. For example, a domain can use the information received from a terminal about the IP addresses it will use to open the gates of the access gateway or a pinhole in a firewall.

Some policies can be implemented as session-independent or as session-specific policies. For example, a domain may choose to provide terminals with a list of all the audio codecs the terminals are allowed to use as a session-independent policy. On the other hand, a domain could choose to be informed about the codecs a terminal intends to use in a session and inform the terminal which of the codecs are acceptable.

Some domains prefer to implement this type of policy as session-specific policies to avoid disclosing the entire domain policy to the terminal. These domains want to prevent competitor operators from copying their policies, which some providers believe to be a source of competitive advantage.

THE POLICY SERVER

Session policies define a logical entity referred to as a policy server. Terminals subscribe to the policy server using a SUBSCRIBE request and obtain information about the domain session-independent policies in NOTIFY requests.

Session-independent policies are a form of configuration information. Therefore, the event package used to transfer session-independent policies is the event package for the user agent profile delivery specified in [5].

Terminals also use SUBSCRIBE requests to obtain sessions-specific policies. When a terminal intends to establish a session, it sends a SUBSCRIBE request to its policy server. This request uses the event package for session-specific policies defined in [6]. The terminal includes an XML document describing the session it is about to establish in the body of its SUBSCRIBE request.

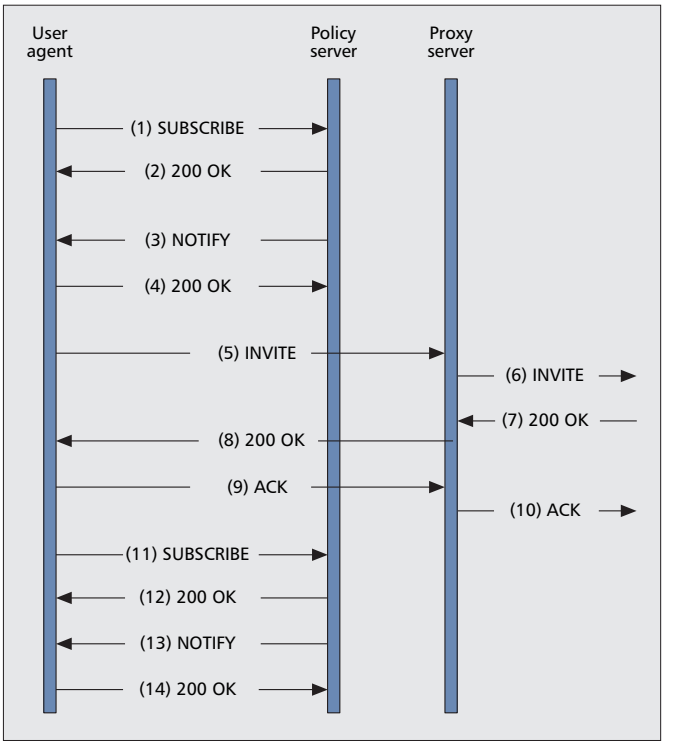
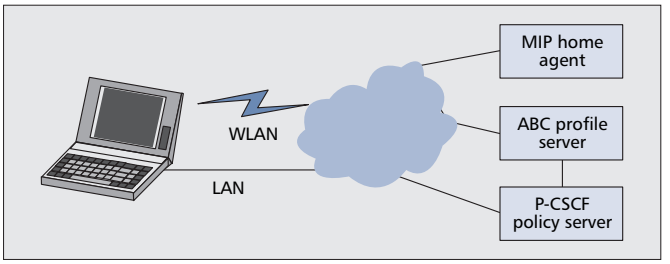


Figure 3. *Session-specific policies message flow.*

The policy server receives the SUBSCRIBE request and based on the XML document received and the user's profile, returns its policies in a NOTIFY request. If the terminal must change the characteristics of the session at some point, it sends a new SUBSCRIBE within the same SUBSCRIBE-initiated dialog to the policy server. If the policy server must send additional policies to the terminal at some point, it issues a new NOTIFY request.

Figure 3 shows an example of a message flow including session-specific policies. The user agent sends the policy server a SUBSCRIBE request (1). The request contains an XML body describing the session the user agent intends to establish. The policy server returns a NOTIFY request (3) that contains an XML body describing the policies applicable to the session.

The user agent generates an INVITE request (5) with an offer that complies with the policies received from the policy server. After the user agent receives an 200 (OK) response (8) with an answer, it sends a new SUBSCRIBE request (11) with a new XML body to the policy server. The new XML body informs the policy server about the session parameters contained in the answer (e.g., agreed codecs, IP addresses, media streams accepted and rejected, and so on.).



■ **Figure 4.** *Implementation architecture.*

SESSION ESTABLISHMENT DELAY

As discussed earlier, session-independent policies are a form of configuration data. Therefore, they are obtained at configuration time before any session can be established. Consequently, they do not add any session establishment delay.

If session-specific policies are implemented as described in Fig. 3, they increase slightly the session establishment delay in some cases. As shown in Fig. 3, a round-trip time between the user agent and the policy server is required before the initial INVITE request can be sent.

Note that sessions that are not initially acceptable to the network do not suffer any additional delay because of session policies. This is because, even when session policies are not implemented, there is an INVITE/488 (not acceptable) exchange prior to the actual session establishment. This exchange takes the same time as the initial SUBSCRIBE/NOTIFY exchange used by session policies.

Fortunately, it is possible to optimize the flow in Fig. 3 to eliminate the extra delay introduced to sessions that are directly acceptable to the network. Such optimization requires coordination between the policy server and the proxy server in the figure. The user agent sends the SUBSCRIBE request (1) and the INVITE request (5) in parallel. If the session is acceptable to the policy server, it informs the proxy server, which routes the INVITE request forward. If, on the other hand, the session is not acceptable, the policy server instructs the proxy server to return a 488 (not acceptable here) response. The user agent receives session-specific policies in a NOTIFY request from the policy server as usual.

Things are different on the terminating side. Session-specific policies introduce a round-trip time to the establishment of sessions that are directly acceptable to the network. This round-trip time between the user agent and the policy server is required between the reception of the INVITE request and the generation of the 200 (OK) response by the user agent. The user agent sends a SUBSCRIBE request and receives a NOTIFY request with policies during this round-trip time.

The same one round-trip delay is introduced to sessions that are not directly acceptable to the network. However, when session policies are not used, those sessions are terminated by the network. Session policies allow those sessions to be established.

The delay introduced to some sessions is the price to pay for having both policy control and end-to-end negotiations. However, when it comes to the future of IMS as a successful service enabling platform, it seems much more important to introduce new services quickly and inexpensively than to provide very low establishment delays in every situation.

Note that a domain that implemented all its policies based on session-independent policies still must use session-specific policies to be informed of the characteristics of a given session. For example, information such as the IP addresses used in a session might be required to open the gates of the access gateway.

SESSION POLICIES IN IMS

The addition of session policies to IMS requires a logical function in the IMS architecture: the policy server. The policy server has a SIP-based interface to the terminals and a Diameter-based interface to the PCRF.

The functionality of the interface between the policy server and the PCRF is similar to the current Rx interface between an AF and the PCRF (Fig. 2). The PCRF makes policy decisions based on the information provided by the policy server, and the policy server informs the terminals about those decisions.

IMS sessions between roaming users involve four different domains, as shown in Fig. 1. The originating domains (home and visited) provide session policies to the originating terminal. The terminating domains provide session policies to the terminating terminal. Therefore, a terminal must consult its visited and home policy servers before attempting to establish a new session. However, even though the terminal consults two policy servers, both consultations can be performed in parallel to avoid additional delays. Interestingly, if both the visited network and the home network find a particular session unacceptable, the session-policies-based approach achieves a lower session establishment time than the currently specified approach to IMS policy control.

When the current approach is used, the P-CSCF rejects the initial INVITE request from the terminal. The terminal modifies its session description according to the 488 (not acceptable) response received, and issues a new INVITE request. This INVITE request is then rejected by the S-CSCF with a new 488 (not acceptable) response.

If session policies were used instead, the terminal obtains (in parallel) the policies from both domains before generating its initial INVITE request. This way, it saves one round-trip time.

Alternatively, session policies from both domains can be consolidated into a single policy. The terminal consults only the visited policy server, which applies its policies and consults the home policy server. The home policy server applies further policies, if required, and informs the visited policy server, which sends the terminal the consolidated policies for the session.

IMPLEMENTATION EXPERIENCE

This section describes our proof-of-concept implementation of session policies in an IMS

network. One of the goals of this implementation is to enable the network to modify its policies in the middle of an ongoing session. When the policies change, the network informs the terminal, and the terminal adapts to the new policies, typically by sending a re-INVITE request.

Note that, currently, the IMS architecture does not provide a means for the network to inform terminals about policy changes that apply to ongoing sessions, as discussed earlier.

A network can change the policies applicable to a given ongoing session based on a set of events. We chose to implement a multi-access environment where our terminals could move between different accesses. The accesses had different characteristics and thus, different policies associated to them. For example, video streams were not allowed on certain accesses. Therefore, when a terminal moved to a new access with different policies, it was informed by the network about those new policies.

Figure 4 shows the architecture of our implementation. It represents the network of an operator that provides IP connectivity over two different access technologies and IMS connectivity. The two access technologies chosen for our implementation were an 802.11-based WLAN and an Ethernet-based LAN. The mobility between both accesses was handled using Mobile IP (MIP).

We based our implementation on the 3GPP R5 IMS architecture, where the P-CSCF and the PDF (policy decision function) were co-located (the 3GPP R7 PCRF is an evolution of the PDF). Consequently, we co-located the P-CSCF and the policy server in a single node.

Access management was based on the ABC (always best connected) architecture proposed in [7]. Our terminals could detect when one of their link layers came up (e.g., the terminal gained WLAN coverage). At that point, they informed the ABC profile server about all of their available accesses and the accesses used at that point.

When a terminal chose to use a new access, it informed the ABC profile server, which in turn, informed the policy server. As a result, the policy server sent new policies to the terminal.

In our test bed, the terminals consisted of Linux-based laptops with software-based IMS clients on them. We were required to modify the software-based IMS clients to implement session policies and the ABC framework.

The policies used in our experiments consisted of allowing audio and video streams over one access but only audio streams over the other. When a terminal changed access, it received new policies. Based on the new policies received, the terminal sent a re-INVITE, adding or removing the video stream.

As expected (we did not implement the optimization described earlier), we observed higher session establishment delays for session requests that were acceptable to the network. The extra delay was caused by the initial SUBSCRIBE/NOTIFY exchange between the terminal and the policy server. In this case, the NOTIFY request simply informed the terminal to establish the session because it was acceptable.

Session requests that were not acceptable to the network did not suffer any additional delay because of session policies, as discussed earlier.

Our implementation shows that currently defined IMS policy control functions can be implemented based on session policies. Additionally, functions that are not supported by the current IMS policy control architecture, such as policy modifications for ongoing sessions, also can be implemented based on session policies.

Future Work

Our future work includes standardizing the elements of session policies in the IETF (Internet Engineering Task Force). A proposal for a framework for session policies can be found at [8]. A proposal for the event package to deliver session-specific policies discussed previously can be found at [6]. A proposal for an XML format to carry policy information can be found at [9]. A set of policy-related functions that are currently based on SDP rewriting and could benefit from session policies can be found at [10]. Our future work also includes proposing the addition of policy control based on session policies into IMS specifications.

Conclusions

This article describes how session policies could be used to implement policy control in IMS. Using session policies instead of the currently specified techniques for policy control would make IMS a better platform for service innovations.

Session policies remove the requirement to inspect and understand session descriptions in the network. Therefore, the use of session policies would enable service providers to offer innovative services based on new SDP extensions or other session description formats without requiring them to upgrade their networks. This way, IMS services would reach their users faster and at a lower price.

The price to pay to implement session policies is a slightly higher session establishment delay for sessions that would have been acceptable to the network in the first place. Although providing innovative services is a key to the success of IMS, optimizing session establishment times to a minimum in every possible scenario is not seen as essential by many users.

The use of session policies would enable service providers to offer innovative services based on new SDP extensions or other session description formats without requiring them to upgrade their networks. Thus, IMS services would reach their users faster and at a lower price.

References

[1] J. Rosenberg *et al.*, "SIP: Session Initiation Protocol," IETF RFC 3261, June 2002, at http://www.rfc-editor.org/rfc/rfc3261.txt

[2] J. Rosenberg and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)," IETF RFC 3264, June 2002, http://www.rfc-editor.org/rfc/rfc3264.txt

[3] M. Handley, V. Jacobson, and C. Perkins, "SDP: Session Description Protocol," IETF RFC 4566, July 2006, http://www.rfc-editor.org/rfc/rfc4566.txt

[4] G. Camarillo, W. Marshall, and J. Rosenberg, "Integration of Resource Management and Session Initiation Protocol (SIP)," IETF RFC 3312, Oct. 2002, http://www.rfc-editor.org/rfc/rfc3312.txt

[5] D. Petrie, "A Framework for Session Initiation Protocol User Agent Profile Delivery," IETF Internet draft, draft-ietf-sipping-config-framework-08, Mar. 2006, work in progress, http://www.ietf.org/internet-drafts/draft-ietf-sipping-config-framework-08.txt

[6] V. Hilt and G. Camarillo, "A Session Initiation Protocol (SIP) Event Package for Session-Specific Session Policies," IETF Internet draft, draft-ietf-sipping-policy-package-02, Oct. 2006, work in progress, http://www.ietf.org/internet-drafts/draft-ietf-sipping-policy-package-02.txt
[7] E. Gustafsson and A. Jonsson, "Always Best Connected," *IEEE Wireless Commun.*, vol. 10, no. 1, 2003.
[8] V. Hilt, G. Camarillo, and J. Rosenberg, "A Framework for Session Initiation Protocol (SIP) Session Policies," IETF Internet draft draft-ietf-sip-session-policy-framework-00, Oct. 2006, work in progress, http://www.ietf.org/internet-drafts/draft-ietf-sipping-session-policy-framework-00.txt
[9] V. Hilt, G. Camarillo, and J. Rosenberg, "A User Agent Profile Data Set for Media Policy," IETF Internet draft, draft-ietf-sipping-media-policy-dataset-02, Oct. 2006, work in progress, http://www.ietf.org/internet-drafts/draft-ietf-sipping-media-policy-dataset-02.txt
[10] J. Hautakorpi *et al.*, "Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments," IETF Internet draft, draft-ietf-sipping-sbc-funcs-00, Nov. 2006, work in progress, http://www.ietf.org/internet-drafts/draft-ietf-sipping-sbc-funcs-00.txt

BIOGRAPHIES

GONZALO CAMARILLO (Gonzalo.Camarillo@ericsson.com) received his M.Sc. degrees in electrical engineering from the Royal Institute of Technology, Stockholm, Sweden, and from Universidad Politecnica de Madrid (Spain). He heads the Advanced Signaling Research Laboratory at Ericsson in Finland. He has co-authored, among other standards, the SIP specification (RFC 3261). He is the IETF liaison manager to 3GPP and currently co-chairs the SIPPING and HIP working groups in the IETF. He is the author of the books *SIP Demystified* and *The 3G IP Multimedia Subsystem (IMS)*. His research interests include signaling, multimedia applications, transport protocols, and network security

TERO KAUPPINEN (tero.kauppinen@ericsson.com) received his M.Sc. degree in computer science from the University of Helsinki, Finland. He has a strong background in Linux, Internet protocols, and programming on both the kernel and user levels. He is currently working as a research scientist at Ericsson Research in Finland. His current focus is on IPv6 related research and prototyping.

MARTTI KUPARINEN (martti.kuparinen@ericsson.com) is currently working as a research scientist at Ericsson Research in Finland. He has a strong background in UNIX, Internet protocols, and programming. He has been working with various IPv6 related issues since the late 1990s. His main focus is building various prototype systems.

IGNACIO MAS IVARS (ignacio.mas.ivars@ericsson.com) received his M.Sc. degrees in electrical engineering from the Royal Institute of Technology, Stockholm, Sweden, and Universidad Politecnica de Madrid, Spain. He obtained his Technology Licentiate degree from the Royal Institute of Technology. He works in the Service Layer Technology Department of Ericsson Research, Stockholm. He has authored and co-authored several papers on admission control and quality of service on the Internet. His research interests include admission control, quality of service, multimedia transport, signaling, and network security.