SHIP – SIP HTTP INTERACTION PROTOCOL
Proposing a Thin-client Architecture for IMS Applications

Joachim Zeiß, René Gabner, Sandford Bessler
Telecommunications Research Centre Vienna (ftw.) Donau-City-Straße 1, A-1220 Vienna, Austria

Marco Happenhofer
Vienna University of Technology Favoritenstraße 9/388, A-1040 Vienna, Austria

Keywords: Distributed mobile computing, HTTP, IMS thin client, Mobile client s/w virtualization, Mobile widgets, SaaS, SIP

Abstract: IMS is capable of providing a wide range of services. As a result, terminal software becomes more and more complex to deliver network intelligence to user applications. Currently mobile terminal software needs to be permanently updated so that the latest network services and functionality can be delivered to the user. In the Internet, browser based user interfaces assure that an interface is made available to the user which offers the latest services in the net immediately. Client software is virtualized using script and widget technologies, which allow user interfaces to run on different hardware platforms and operating systems. Our approach, called SHIP, combines the benefits of the Session Initiation Protocol (SIP) and those of the HTTP protocol to bring the same type of user interfacing to IMS. SIP (IMS) realizes authentication, session management, charging and Quality of Service (QoS), HTTP provides access to Internet services and allows the user interface of an application to run on a mobile terminal while processing and orchestration is done on the server. A SHIP enabled IMS client only needs to handle data transport and session management via SIP, HTTP and RTP and render streaming media, HTML and Javascript.. Furthermore, the SHIP architecture allows new kinds of applications, which combine audio, video and data within a single multimedia session.

1 INTRODUCTION

The IP Multimedia Subsystem (IMS) (3GPP, 2009) is expected to provide convergent applications to mobile terminals using the Session Initiation Protocol (SIP) (Rosenberg et al., 2002). The IMS architecture envisions mobile terminals supporting SIP. SIP messages are routed from a user terminal across the IMS infrastructure to a specific application server in the system. A service application or component is triggered which, in turn, orchestrates service enablers (such as presence, location, etc.).

Besides SIP, currently, additional protocols are required to be supported by an IMS capable client: the XML Configuration Access Protocol (XCAP) (Rosenberg, 2007) for configuration of XML documents on XML Document Management Server (XDMS), the Message Session Relay Protocol (MSRP) (Campbell et al., 2007) for stateful messaging, and of course HTTP (Fielding et al., 1999) – for accessing the services via web front-ends. From the observation above, we see the need to simplify the interactions with service applications, to reduce the number of required protocol stacks and the amount of processing in the terminal.

The lack of IMS terminals on the market is a major obstacle for IMS/NGN roll-out. Like existing SIP clients, current IMS clients support in general a non-extensible set of applications like VoIP, video, chat, presence, and address book functions. Interlinking the SIP and HTTP protocols, as presented in this paper, however, will provide the basis for an open model of SIP/IMS terminals. With this model new services and enablers become available to the mobile terminal instantly; there is no need to deploy additional client software or even upgrade the operating system to support new network capabilities and services.

ZeiÃ§ J., Gabner R., Bessler S. and Happenhofer M.
SHIP âĂŞ SIP HTTP INTERACTION PROTOCOL - Proposing a Thin-client Architecture for IMS Applications.
DOI: 10.5220/0001840600210028
In *Proceedings of the Fifth International Conference on Web Information Systems and Technologies (WEBIST 2009)*, page
ISBN: 978-989-8111-81-4

We propose a thin client architecture for IMS applications by introducing a distributed MVC software pattern (Gamma et al., 1995). The user interface components (view and control) run on the terminal and interact with processing components (model) executed on the application server. The components communicate via HTTP session controlled by SIP, concurrently with voice or video transport inside the same dialog.

Generally, the SHIP solution follows the Software as a Service (SaaS) model already envisioned by Bennett et al. (Bennett et al., 2000) to unburden the user of software maintenance, ongoing operation, and support. Our architecture enables on-demand pricing of applications, a better protection of intellectual property for the software vendor and the network operator to control services and act as application service provider.

The main objective of this work is to present a solution for a better exploitation of the SIP signaling towards a seamless, secure, user friendly and provider-efficient deployment of IMS services. Such a solution would satisfy a number of wishes of network operators and users:

- User micro-portal: IMS subscribers can connect to a user-friendly, appealing personalized homepage in which relevant information and their commonly used functions are presented. The users can manage their own profile data, application preferences and privacy settings.
- Single sign-on: a user authenticated to IMS doesn't need to authenticate again to perform actions for them she is authorized or consume services to which she is subscribed.
- Web based user interface: in order to avoid the development and installation of client applications, web and web 2.0 technology could be used in the browser.
- Asynchronous (HTTP push) events dispatched by applications towards user terminals are supported.

The SHIP (SIP/HTTP interaction protocol) architecture is a blending of the SIP and HTTP protocols. IMS authentication is used to manage HTTP sessions within a SIP dialog. This approach leads to a novel, web based, universal IMS terminal.

1.1 Related Work

Addressing the single-sign-on feature, 3GPP has proposed the Generic Bootstrapping Architecture (GBA) (3GPP, 2008) (M. Sher, T. Magedanz, 2006). This framework introduces two new functions called Bootstrapping Server Function (BSF) and Network Authentication Function (NAF). The BSF has access to the Home Subscriber Server (HSS) and can download user profiles and credentials. The NAF offers services to the User Equipment (UE). In order to use the service, the UE has first to authenticate on the BSF and the NAF can later check the UE authentication. This requires an implicit authentication at the network; our approach realizes the authentication explicit.

The Sun JSR 281 specification (Java Specification, 2006) describes the IMS API that enables IMS applications to be installed on a JSR281-enabled device. The API assumes the existence of basic capabilities such as Push-to-Talk over Cellular (PoC), presence, instant messaging, etc. as well as the access to SIP headers and message bodies. Complementary to our approach, the use of JSR 281 implies the development and deployment of each new application on the terminal. Thus often CPU-intensive and complex orchestration of IMS enablers has to be done at the client's terminal.

In the EU FP6 project Simple Mobile Services (SMS) the authors propose to transport serialized messages coded in JSON (scripts) via SIP messages, taking advantage of the asynchronous character of SIP. This approach uses the signaling infrastructure (SIP proxies) to transport user data. Our approach uses the infrastructure only to establish a session and transports the user data point to point and reduces so the load on the SIP infrastructure.

1.2 Structure of this Paper

The rest of the paper is organized as follows: section 2 describes the proposed system architecture, section 3 explains the mechanisms to integrate bi-directional HTTP transport into a SIP session, section 4 explains the implementation of the SHIP concept on clients and servers and section 5 describes a design example for a typical SHIP based application. Section 6 concludes our work and gives further research directions.

2 SYSTEM ARCHITECTURE

In the following we assume that the environment for deploying our solution is that of an IMS service provider. First we introduce the SHIP concept and analyze the functionality required by the (mobile) terminal as well as on the server. Section 2.2 explains the integration of SHIP into an IMS overlay network.

2.1 General Architecture

The basic concept of SHIP is to combine and manage HTTP sessions with SIP. A SIP INVITE message is used to negotiate a TCP channel for HTTP connections. To achieve this, the media capabilities of a User Agent to handle HTTP requests are extended. A detailed explanation of the session setup is given in section 3. Of course it is still possible to establish additional media sessions (e.g. voice and video) within the same SIP dialog.

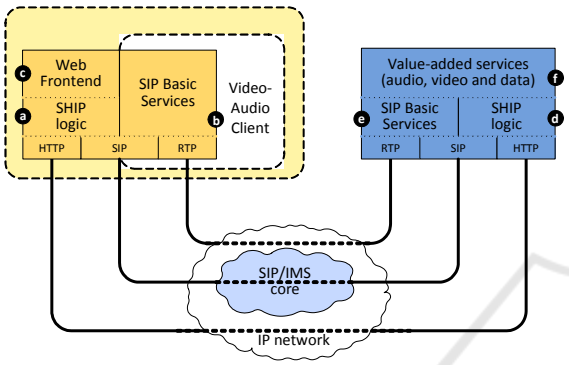

Figure 1: SHIP architecture.

Figure 1 gives an overview of the involved functions. It illustrates the SHIP terminal on the left hand side and a SHIP server on the right hand side. Both components are connected via a mobile IP network.

A traditional User Agent for voice and audio calls, called "SIP basic services" is depicted in

Figure 1ⓑ. For these services a client uses both a SIP and a RTP stack. The SHIP terminal reuses the existing functionalities while adding additional capabilities. The SHIP logic ⓐ uses SIP to establish a dialog that negotiates a TCP channel for HTTP traffic. The processing logic establishes, manages and terminates the SIP dialog for the SHIP session. It also ensures that each HTTP request is associated with a secret session key. The SHIP logic is the only component, which is aware of the 1:1 relation between a SIP dialog and a SHIP session. The Web front-end ⓒ (e.g. a web browser or html rendering engine) uses the SHIP logic to send and receive HTTP messages and to display the HTTP part of the service. It is possible to access services which offer audio, video and data in any combination with only one user agent. This approach makes the SHIP architecture flexible for service developers and can be used to run the user interface at the user terminal, whereas service logic is being processed at the server.

At the server side HTTP and RTP traffic as well as SIP signaling are terminated. The RTP media is being processed by the SIP basic services ⓔ component and the HTTP traffic by the SHIP logic ⓓ component. SIP signaling is terminated in one or both components, depending on the requested media type. Since the SIP signaling and the HTTP traffic of the client terminates at the server, the SHIP logic can correlate both sessions with a unique secret session key negotiated via SIP. The SHIP logic assigns the key to each incoming valid HTTP request and forwards the request to the application. The latter can be a value-added service ⓕ as depicted in Figure 1, or a proxy which forwards the request with the user identification to a 3rd party application in a trusted domain.

A SHIP server consists of two main components the SIP Session Manager (SSM) and an HTTP proxy.

The SSM is the endpoint for SIP messages sent by a client to establish a SHIP session. It stores a unique session ID for each SHIP session. Subsequent HTTP requests from the client's terminal containing a valid session ID are forwarded by the HTTP proxy. The proxy optionally replaces the SHIP session ID with the *"P-Asserted-Identity"* (Jennings et al., 2002) header value to assure the identity of the requester to subsequent AS.

2.2 Integration of SHIP in IMS

This section shows a possible integration of SHIP into a mobile provider's IMS. Figure 2 depicts the provider's IMS core, the SHIP server and IMS service enablers, like presence, location or group management. The enablers provide beneficial IMS features, which can be easily used and combined by SHIP services without the need of deploying enabler specific software on the user terminals. SHIP clients and 3rd party providers can connect to the SHIP server. The IMS core is the point of contact for any SIP based communication from and to the end-user. SIP requests, addressing a SHIP service, received at the IMS Call Session Control Functions (CSCFs) 0 are routed to the SHIP server, by executing Initial Filter Criteria (IFC) or by resolving the provided SIP URI. The connection from the SHIP terminal to the IMS core and the SHIP server is provided by the PLMN using HSDPA, WLAN, etc. 3rd party content and service provider are connected via Internet.

However, it is possible to build more sophisticated services using a SHIP terminal as shown in section 5. Similar to enabler services, 3rd party data and applications could be used to create

valuable services.

The existence and correctness of the *"P-Asserted-Identity"* header provided by the IMS overlay network is essential for the functionality and authentication, because the involved components may authenticate the subscriber by using this header. To realize correctness, all involved SIP entities and HTTP entities which receive or forward SIP or HTTP requests, have to guarantee that this header is not manipulated and therefore they have to be in a domain of trust. In the case of IMS this could be the home network of the subscriber or just a trusted 3rd party service provider.

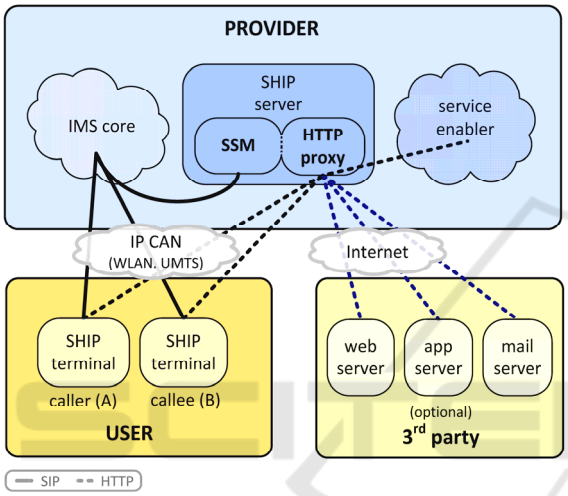

Figure 2: SHIP in IMS.

3 MANAGING THE HTTP SESSION

SIP uses the Session Description Protocol (SDP) (Handley, Jacobson, 1998) to negotiate the parameters of a session like protocol, media type, IP addresses and port numbers. Media types in SDP are: video, audio, application and data, with UDP as transport protocol. Currently, it is not possible to specify HTTP interactions with SDP since HTTP relies on TCP as transport protocol. In (Gamma et al., 1995) the authors specified an extension to SDP to enable the transport of fax data using SIP. For that purpose an extension is defined that expresses TCP data as a media type.

According to (Yon, Camarillo, 2005) in which a TCP connection for transmitting a fax is announced via SDP (see figure 3) we propose a similar extension for HTTP sessions. If the calling party offers the SDP of Figure 3 and the called party answers with the SDP of Figure 4, then the called party would start a TCP connection form port 9 (IP address 192.0.2.1) to port 54111 (IP address 102.0.2.2).

```
m=image 9 TCP t38
c=IN IP4 192.0.2.1
a=setup:active
a=connection:new
```

Figure 3: TCP client part.

```
m=image 54111 TCP t38
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:new
```

Figure 4: TCP server part.

To handle a HTTP connection we propose the configuration depicted in Figure 5 and Figure 6. In our case media type m changes from image to application and protocol from t38 to http. Furthermore, we reuse the encryption key parameter k of the SDP to store a session key.

```
m=application 54111 TCP http
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:new
k=clear:key123
```

Figure 5: HTTP server part.

```
m=application 9 TCP http
c=IN IP4 192.0.2.1
a=setup:active
a=connection:new
```

Figure 6: HTTP client part.

For a User Agent, which is able to handle HTTP request and response messages, the associated SDP message includes the server and client part (Figure 7) to establish bi-directional HTTP data transport.

```
m=application 54111 TCP http
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:new
k=clear:key123
m=application 9 TCP http
c=IN IP4 192.0.2.1
a=setup:active
a=connection:new
```

Figure 7: HTTP bi-directional.

The encryption of the HTTP stream should be handled by SSL/TLS (Lennox, 2006). The SDP also transports a session key, to be added as custom header to each HTTP request. It maps the incoming HTTP requests to their corresponding SIP session. This mapping mechanism authenticates the HTTP request originator. An IP address and port inspection could also identify the user.

Figure 8 shows the message flow of a SHIP session. The calling party acts as HTTP client and

SIP User Agent, the called party as HTTP server and SIP User Agent. In this scenario the called party assigns a session key *"key123"* to the SIP Call-ID *"call1"*. The called party sends the session key in the 200 OK SIP response to the calling party. The HTTP server maps all incoming HTTP requests with session key *"key123"* to call-ID *"call1"*.

The validity of a HTTP session is limited by the validity of the corresponding SIP session. If the SIP session is closed or no longer valid, the HTTP servers will not accept any request with key *"key123"*. Both parties are able to start additional video and audio streams using re-INVITE. It is also possible to add an HTTP stream to a video and audio stream, so that the multimedia session can cover audio, video and data (application).

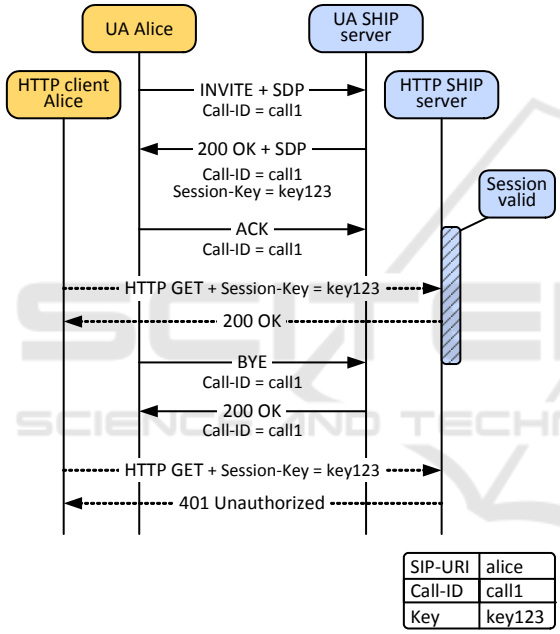

Figure 8: Authenticated HTTP dialog within a SIP session.

4 SYSTEM IMPLEMENTATION

Our implementation is based on the IMS Communicator (http://imscommunicator.berlios.de/) and the SIP communication server from Oracle. Our extended version of the IMS Communicator performs the IMS registration, and the SIP session establishment. From the received SDP data, it takes the IP address, port and session key information and forwards it to the local running HTTP proxy, which realizes the SHIP logic in Figure 1. The local web browser is configured to communicate via this HTTP proxy. At the remote side the SIP server handles both SIP and an HTTP. SIP functions are implemented as SIP Servlets (Java Specification, 2008), the HTTP part is implemented using HTTP Servlet technology. To achieve SHIP functionality the SIP Servlet as well as the HTTP Servlet share a common session object provided by the SIP server.

4.1 Session Setup

When the user accesses a webpage (e.g. personalized access to subscribed services), the browser sends the request to the local HTTP proxy. If a SHIP session already exists, the proxy adds the corresponding session key to the HTTP request and forwards the request to the SIP communication server. If no SHIP session is running, the proxy triggers the IMS Communicator to establish the SIP dialog of the new SHIP session.

For this purpose a SIP INVITE message is sent to a predefined SIP service URL including SDP data depicted in Figure 7. Initial Filter Criteria (IFC) evaluate the corresponding SIP server. The *"P-Asserted-Identity"* header assures the application server of the callers' identity. The incoming SIP INVITE message triggers a SIP Servlet at the application server which generates the SHIP session key. This ID is stored for later usage by HTTP Servlets. A SIP 200 OK response message containing session key as well as HTTP server IP and port number is sent in reply to the INVITE. When the IMS Communicator receives the 200 OK it forwards the relevant SDP parameters to the HTTP proxy module.

To setup the HTTP part of the SHIP session, the HTTP proxy takes the IP address and port number of the communication server and prepares a connection for later HTTP requests. Now, the SHIP session is valid and HTTP requests can flow in both directions

Incoming HTTP requests at the SHIP server trigger an HTTP Servlet which inspects the authentication header as well as the originating IP address and port. It compares the session key to the previously stored value. In case of matching session key values, the session key header is replaced by the P-Asserted-Identity header of the linked SHIP session. Next, the server forwards the request to its final destination. If there is no matching key, the Servlet rejects the HTTP request with an HTTP 401 error response.

Since the application server correlates all incoming traffic with a certain session key to the identity of the called party and no further authentication is used to access value-added services, it might become a target for "man in the middle" attacks. If one eavesdrops the HTTP (or

SIP) traffic he can find out the session key and use it for service requests on behalf of the original called party. Therefore, it is required to use TLS (Dierkse, Allen, 1999) to secure the HTTP transport and IPSec (Kent, Atkinson, 1998) to secure SIP.

Since the proposed mechanism is applied for each TCP connection, the HTTP performance deteriorates, since each request has to be processed sequentially. Most web browsers start several TCP connections to decrease the latency due to the processing of a webpage composed of several resources. To improve the performance, the proxy module of the IMS Communicator has been modified so that several TCP connections are established and are also accepted by the HTTP protocol stack within the SIP application server, without considering the TCP port announced earlier in the SDP of the IMS Communicator.

5 APPLICATION EXAMPLE

This section discusses a typical SHIP based application example. The Multimedia Call Center presented is a rich call scenario where communication takes place via voice and visual components derived from an associated HTML connection. Thus, we combine two different media types - data and voice - to one session, referred to as "rich call" (optional video).

In call centers there are often different experts for different concerns or questions. To forward calls to the right expert, Interactive Voice Response (IVR) is used. The system informs the caller, which key to press or which keyword to speak in order to proceed. The IVR system detects the selections and finally forwards the call to the corresponding expert. This approach reduces the number of call agents, who only categorize and forward calls. But there are also drawbacks. After listening to a long list of options people might not be able to recall their best choice, or cannot decide which option to choose. Some sorts of dialogues cannot be implemented using IVR systems at all, as there are too many options confusing the customer just by listening to them. To reduce this problem most IVR systems query the caller several times with small questions. But in general, complex, voice-only, IVR dialogues are hard to follow for humans. We believe that serving customer requests can be simplified, if graphical user interfaces are combined with voice communication.

Callers are guided through the menus by visual input forms optionally in combination with voice instructions like "please assign a category to your question, if possible". Indeed this approach does not require any voice recognition, because the choices made by the caller are transmitted as data to the call center. Therefore, this type of rich call applications, can realize much more complex dialogues as today's IVRs. Processing the Personal Identification Number (PIN) or determining the right agent are just two possible examples.

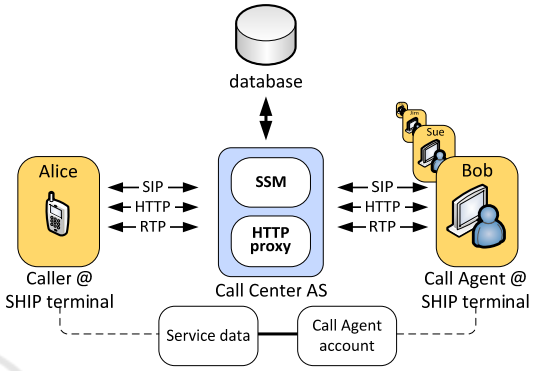

Figure 9: Call Center with SHIP architecture.

Figure 9 shows the architecture of a call center using SHIP. There is a group of call agents with partly different skills. The Call Center AS (SHIP server) terminates SIP and HTTP signaling as well voice data. It might access a database, where account data are stored. For each SHIP caller the AS stores the service data (account data, selected question category, etc.), already entered by the caller. The call agent has access to these service data, as soon as he accepts the call.

Figure 10 shows the call flow for this scenario. Alice sends a SIP INVITE message (1) addressed to the call center's SIP URI. The IMS core routes the request to the application server. It is worth to mention, that the request includes two different media types inside the SDP part, one for *audio* and one for *data* (HTTP). If both media types are present, the AS detects the client's *SHIP capabilities*. The AS agrees to the client's connection parameters (2 & 3) and starts a session. Both, audio and data terminate at the AS. If the SDP data media type were missing - indicating that the client cannot handle HTTP - a voice-only call would be established.

Since both client and server are SHIP enabled, the client sends an HTTP request (4) associated to the previously established SHIP session to the server. The server responds with an HTML form (5) where Alice can enter her account data, and specify her concerns. In traditional IVR systems these data

are transferred via voice. The major benefit of our solution is that the system assists Alice utilizing text and audio to fill up the form properly. The client sends Alice's service data to the AS (6 & 7) which generates a ticket for this request and assigns this request to the correct group of call center agents. The way the AS assigns the call to a certain agent, is out of scope of this document.

Before the AS sends an INVITE message (8) to an agent (Bob), it correlates the call from Alice with the new one. Note that the correlation includes all service data of Alice which become accessible for Bob from now on. From the SIP and RTP perspective, the AS acts as a Back-2-Back User Agent (B2BUA), as defined in (J. Rosenberg et al., 2002) Regarding HTTP the AS acts as an HTTP server, handling both parties within one joint session. This call center system requires that all call agents use SHIP enabled clients. Thus the call can be established with voice and HTTP media (9 & 10). Bob's client is aware of the AS' SHIP capabilities, because it received the corresponding SDP data media type within the INVITE request (8). As a result Bob's call agent sends an HTTP request (11). The AS responds with Alice's service data (12) (e.g. an HTML form with all account data and the recent bills).

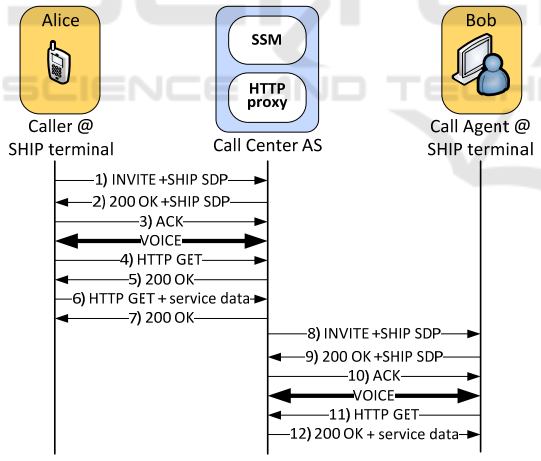

Figure 10: Call flow in a SHIP enabled Call Center.

During this session, the associated HTTP connection supports the call center agent by displaying certain aspects of Alice's bill on her mobile's display. The following Figure 11 shows how Bob uses the data channel to display billing data at Alice's SHIP client. For this reason he updates the session at the AS with a new HTTP request (1 & 2) (e.g. "show bill"). Alice's client has to update its screen using AJAX or SHIP based

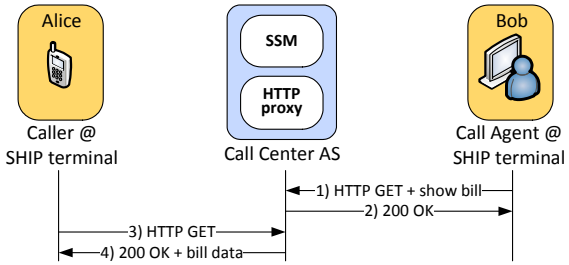

Figure 11: End 2 End data communication.

HTTP push) (3 & 4).

With this mechanism it is possible to push images and websites to the other party or to send data from the caller to the call agent.Parts of the functionality described above could also be implemented as a web application guided by voice (transferred over HTTP), in connection with a third party voice call. But our approach reuses the IMS infrastructure to achieve QoS, and to validate the identity of the users.

6 CONCLUDING REMARKS

In this paper we show a new way of approaching application development in IMS. The basic idea is to blend the SIP and HTTP protocols in order to achieve authenticated and authorized web based communication, while keeping the advantages of IMS (charging, QoS, roaming, usage of service enablers, etc.). Our first prototype implementation could proof the concept. For a large scale deployment, we will discuss the required changes for SDP standard within the respective standardization groups.

Instead of executing a monolithic stand-alone application on the mobile terminal we propose to distribute the software between client and server following the SaaS model. In our application architecture GUI components are executed on the client side, whereas functional components, data processing and enabler orchestration reside on the application server. Interactions between terminal and server are entirely controlled by SIP. Bi-directional HTTP traffic to transport GUI and user events is managed by a SIP session. We reduced the protocols required to be supported by an IMS client to SIP (incl. RTP) and HTTP. There is no need for deploying enabler or application specific code on the mobile terminal.

We discussed an application example demonstrating the capabilities of our approach, which is currently being implemented by the

BACCARDI project (www.ftw.at). This project investigates future opportunities and enhancements of IMS and was initiated by the Telecommunication Research Center Vienna (ftw.) and its industrial partners.

In the future we will compare the possibilities of introducing a HTTP push mechanism controlled by either SIP Subscribe/Notify messages or a Call-Id related SIP MESSAGE message. In both cases the client is informed about changes of previously downloaded content (occurring during the SIP session). Furthermore, we are looking at integrating our solution into currently available mobile widget engines such as the Symbian Web runtime (WRT) .

The SHIP concept is not only a simplification of IMS protocol handling but opens new opportunities to service providers, service developers, network operators and users. Service providers can offer their Internet services to the community of IMS subscriber and reuse the IMS infrastructure for charging, QoS and authentication. Service developers can implement the services, as they did for the Internet community, without any special knowledge about deploying telco services. The network operators might profit from this approach, because they can sell more services, like those which are deployed for the Internet. Finally there is no need for the user to install and update service specific software.

ACKNOWLEDGEMENTS

This work has been supported by the Austrian Government and the City of Vienna within the competence center program COMET. We thank our colleagues in the BACCARDI project, especially Joachim Fabini and Rudolf Pailer for the fruitful discussions and contributions.

REFERENCES

3GPP TS23.228, 2009. *IP Multimedia Subsystem (IMS); Stage 2*

J. Rosenberg et al., 2002. *SIP: Session Initiation Protocol*, RFC 3261

J. Rosenberg, 2007. *The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)*, RFC4825

B. Campbell, R. Mahy, C. Jennings, 2007. *The Message Session Relay Protocol (MSRP)*, RFC 4975

R. Fielding et al., 1999. *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. 1995. *Design Patterns*, Addison-Wesley.

Bennett, K., Layzell, P, Budgen, D., Bereton, P., Macauley, L. A., Munro, M., 2000. *Service-Based Software: The Future for Flexible Software*, Asia-Pacific Software Engineering Conference, 5-8 December 2000.

3GPP TS33.220, 2008. *Generic Authentication Architecture (GAA); Generic bootstrapping architecture*

M. Sher, T. Magedanz, 2006. *Secure Access to IP Multimedia Services Using Generic Bootstrapping Architecture (GBA) for 3G & Beyond Mobile Networks*, Q2Swinet

Java Specification, 2006. *IMS Services API JSR-281*, online: http://jcp.org/en/jsr/detail?id=281

C. Jennings, J. Peterson, M. Watson, 2002. *Private Extensions to Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks*, RFC 3325

M. Handley, V. Jacobson, 1998. *SDP: Session Description Protocol*, RFC 2327

D. Yon, G. Camarillo, 2005. *TCP-Based Media Transport in the Session Description Protocol*, RFC 4145

J. Lennox, 2006. *Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)*, RFC4572

Java Specification, 2008. *Java SIP Servlet Specification v1.1*, JSR-289 online: http://jcp.org/en/jsr/detail?id=289

T. Dierkse, C. Allen, 1999. *The TLS protocol Version 1.0, RFC 2246*

S. Kent, R. Atkinson, 1998. *Security Architecture for the Internet Protocol*, RFC 2401