INTRODUCING A NEW XML-BASED PROTOCOL FOR SIP USER-AGENT SERVICES COLLABORATION: INTEGRATION WITH IP-PHONE AND PC

Wajdi Elleuch
Université de Sherbrooke
Wajdi.Elleuch@USherbrooke.ca

Alain C. Houle
Université de Sherbrooke
Alain.Houle@USherbrooke.ca

Samuel Guénette
M5T
sam@m5t.com

Abstract

Some IP-phones are offering only a simple user interface with plain DTMF digits and optional function buttons, but without any graphical display. This kind of limited user interface makes cumbersome the usage of advanced IP-telephony services in many scenarios. Enhancing IP-telephony services with collaborative devices can both resolve the capabilities limitations of IP-phones and also encourage user location mobility. The introduced solution consists of using a third party device, such as a PC on an IP network, to make a connection with the IP-phone where the connection is realised with a specific protocol based on the Session Initiation Protocol (SIP); this protocol benefits from the Subscribe/Notify mechanism defined by the IETF. By exchanging Notify requests and by using suitable information encoding, the two devices are able to collaborate and can evolve together to carry out advanced IP-telephony services. On the scope of this work, some collaborative services such as the caller-id service were analysed, developed and implemented.

Keywords: *IP-Telephony, SIP, XML-based protocol, Subscribe/Notify mechanism, SIP user agent services, service collaboration.*

1. Introduction

The take-up of voice over IP (VoIP) and convergence of voice and data networks has brought new opportunities in both mass-market and business communications. The procurement of this new technology is often made on the promise of cost savings: e.g. the management (including human resources) of a single network infrastructure against two separate networks and 'free' telephone calls. Indeed, as competition in the telephony industry grows from both traditional and new (IP) providers, it is imperative to provide clear technological leadership and product differentiation — particularly in the corporate market.

The current cost of IP-phone devices is often seen as one of the major issues for any company considering the deployment of IP telephony. Actually, a great number of IP-phones are available on the market where some of them are well equipped with advanced hardware and software capabilities (large LCD, hands-free, camera, etc.). The cost of these machines remains relatively expensive with the effect of slowing down their expansion for the mass-market.

To offer less expensive equipment to the mass-market, many manufacturers introduced basic IP-phone with limited hardware/software capabilities. Unfortunately, this type of devices causes a limitation for both the improvement of the actual IP telephony services and for the development of new ones. As a basic example, consider a user who wishes to see the identification of the caller before answering the incoming call with an IP-phone that has no display capability.

A solution to this limitation is possible by "connecting" this basic IP-phone to a PC: it becomes now possible to display the caller's identification information. As a logical improvement of this service, it will be possible to create a full collaboration between a PC and an IP-phone to handle the incoming calls. This collaboration includes the ability to answer the call using either the IP-phone or the PC and to notify each device about the sequence of events.

To realize full collaboration of devices' services, a new protocol has to be designed to insure proper information exchange between two to many devices: e.g. an IP-phone and a PC. In the present case, the PC can be considered as third party device extension of the IP-phone that offers new possibilities for the development of services that rely on graphical user interfaces. All the PC capabilities can then be used to offer various collaborative IP-telephony services to the user.

This article is composed of 3 main sections. In section 2, we begin by presenting how collaboration can be initiated and by justifying the principal technological choices. In section 3, we present the principles under the protocol design and its implementation. In section 4, we explore the caller-id service in details. Finally, we conclude this article by suggesting some future work.

2. Enabling collaboration

2.1. Initial use case

The system is composed of 3 entities: one generic SIP User-Agent, one IP-Phone and one PC. The IP-phone plays the role of a centric element that divides the system in two sides, as shown on figure 1. On one side, the standard SIP protocol ensures the establishment, modification and termination of IP-telephony calls. On the other side, collaboration of the PC and

the IP-phone is ensured by a collaboration protocol that guarantees the exchange of asynchronous events notifications. With this collaboration, the proposed system can offer various services for provisioning UAs, exchanging call-history, listening to the voice mailbox, etc. In addition, the execution of a service on a UA can be requested by its "collaborator": e.g. the PC instructs the IP-phone to make a call and to play a ring tone when the call is completed.

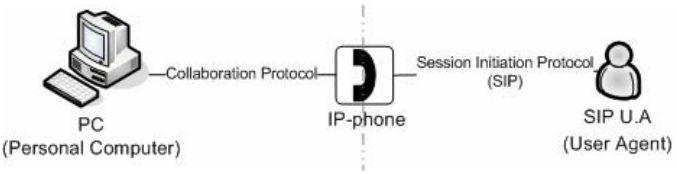

Figure 1: System overview

2.2. Communication model

The collaboration protocol must be session-oriented with the ability to provide sequenced data delivery between peers. A session is long-lived and persistent over protocol transaction operations. This allows each peer to make changes to the session state for the lifetime of the communication. As an example, authentication information specific to a session remains in effect until the session is terminated.

Many technologies can be used to implement this collaboration protocol. Some distributed object technologies, such as CORBA, JAVA/RMI or COM/DCOM, are mainly based on the client/server communication model [1] [2]. In this model, services are defined on the server side and the client can use them by sending requests to the server. The client can't be automatically notified when change occurs on the server side. This model inducts a "master-slave" relation between peers that reduces the collaboration ability of the communication model. Otherwise, the peer-to-peer model [3] can guarantee a symmetric communication where all peers are equal i.e. peers can both request and offer services. They act as both client and server rather than having a role only as client or only as server. This communication model is more adapted for collaboration between user agent.

2.3. Exchanged data description

Some services supported by the collaboration protocol need description of the data with deep hierarchical organisation. As example, suppose that we would like to send the description of a set of calls that includes received calls, placed calls and missed calls. For each call category, we might need to include a set of information about the called person identification, the caller identification, the start time of the call, the stop time, etc. When choosing to handle these data as objects, many methods have to be introduced to manage each information hierarchical level. This solution can cause complication and extra overhead for developers. A simple solution is to keep data in its textual state. Coding exchanged data in readable human format can facilitate the debugging process.

XML is the lingua franca of interchange, providing a flexible but fully specified encoding mechanism for hierarchical content [4]. Therefore, XML can be a data coding solution between the PC and the IP-phone.

Some remote procedure call (RPC) systems based on XML-RPC [5] or SOAP-XML[6] coding format can be introduced to support the coding data format between the PC and the IP-phone. The limited set of the XML-RPC name-space and the very exhaustive and expensive XML coding requirements of SOAP opened up the opportunity to define our adapted XML coding name-space.

2.4. Transport of data

There are at least two choices to convey data between the IP-phone and the PC. While HTTP[7,8] is popular on Web client-server communication architectures, introducing it on the peer-to-peer communication world can cause some complexity for session management and to give each peer the privilege to provide both request and response abilities [9]. On the other hand, the UA located inside an IP-phone is already enable with the SIP protocol and then offers a SIP software component. Therefore, selecting SIP as the base of the collaboration protocol is natural and represents an easily supported solution which is possible because of the flexibility that SIP offers through its different extensions mechanisms.

3. Designing the *UAs Collaboration Protocol*

3.1. Protocol session establishment

Our goal is to provide a SIP-specific framework for event notification that is simple and efficient for simple features while being flexible enough to provide powerful services. The creation of SIP sessions can be fulfilled by both INVITE[10] or SUBSCRIBE[11] methods. Some SIP methods such as INFO [12], OPTION[10], MESSAGE[13] or NOTIFY[11] can be used within the established SIP session to convey textual information between peers.

Recent IETF RFCs show that there is a strong complementarily between the SUBSCRIBE and NOTIFY SIP methods. The SUBSCRIBE/NOTIFY mechanism, initially introduced by IETF RFC 3265 (SIP – Specific Event Notification), provides an extensible framework by which SIP nodes can request notification from remote nodes indicating that given events have occurred [11]. Used in the presence [14] and Instant Messaging systems, the SUBSCRIBE/NOTIFY mechanism can be generalised to other communication systems that require state events synchronisation. The SUBSCRIBE message establishes a "dialog" with the service agent. A dialog is defined in RFC 3261 [10] and it represents the SIP state between a pair of entities to facilitate peer-to-peer message exchanges. Described in RFC 3265 [11], the NOTIFY message contains bodies (payload) that describe operation related to the subscribed service.

The subscription state can be controlled by introducing the watcher information event template-package for SIP [15,16]. The subscriber can then receive notification about its subscription state (pending, active, waiting, terminated, etc.). The watcher information can be applied on every event package that requires subscription. By following the same principle, it is possible to create a new template-package to report call state for each package that includes call state information. The finite state machine (FSM) framework included in a SIP user agent can then be reported and shared between multiple SIP user agents.

Our system uses separate event package names for different services. The SUBSCRIBE/NOTIFY mechanism is used to ensure the session management between peers and the notifications operations exchange within a service oriented context.

3.2. Protocol layers

Four layers defined the proposed protocol as shown on figure 2: The session layer that provides a communication path between peers using the SUBSCRIBE/NOTIFY mechanism; the event package layer that specifies the concerned service; the operation layer that specifies the event that can be either a request or a response; the parameters layer that contains the specified data for each event.

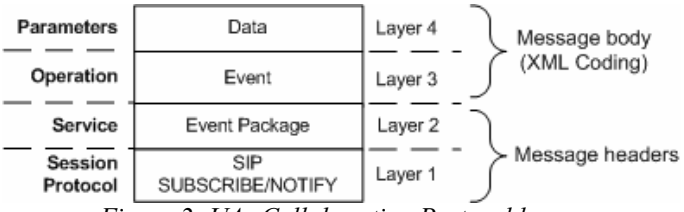

Figure 2: UAs Collaboration Protocol layers

3.3. Protocol messaging mechanism

Before using a service, the user agent (IP-phone or PC) has to subscribe to the event package that corresponds to the intended service. If the subscription is accepted, the user agent will be authorized to receive notifications about subscribed event package state changes. Figure 3 shows an example of the needed message exchange for the execution of the caller-id service.

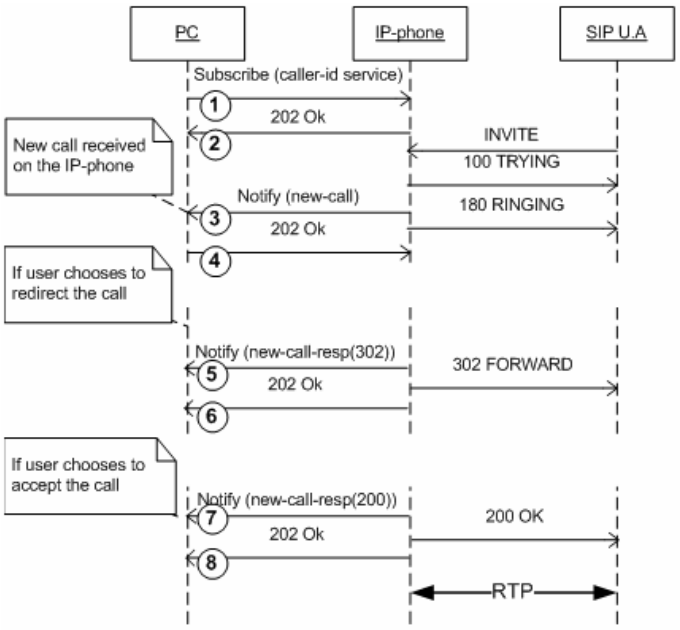

Figure 3: Example of sequence diagram for caller-id service

3.4. Implementation overview

Based on the SIP stack, the SIP user agent (PC or IP-phone) can be mounted by integrating some components to implement the user interface : the SIP stack offers mainly the methods to handle SIP packets; the SIP engine manages sessions, plays the middleware function between logical running applications and SIP stack by translating SIP messages to application messages and vice versa; the logical running application maintains the SIP machine state, gets user action events from user interface component, sends user events to the SIP engine, manages the user interface and refreshes it when receiving new application events.

The implementation of the proposed protocol requires an additional component. Since event handling policy is defined and managed on the logical running application component, we introduced a new engine beneath this component layer. Therefore, on the same level as the SIP engine a new engine named “Collaboration SIP-based engine” is added for handling the collaboration protocol. These two engines use different instantiations of the SIP Stack objects. Nevertheless, they communicate simultaneously with almost the same instantiations of the logical running application objects. Figure 4 shows an overview of the main implemented components and their interactions.

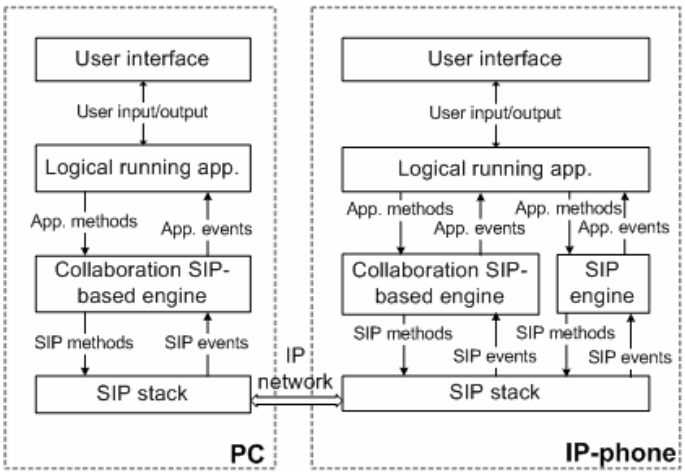

Figure 4: Overview of the main implemented components

4. Example of service

The caller-id service represents a good example of an IP-telephony service executed in collaboration between two devices. When receiving an INVITE, the IP-phone has to notify subscribed users on the caller-id event package. The NOTIFY payload message contains XML coding (see message 3 on figure 3). It includes the name of the event package, i.e. caller-id, the name of the operation i.e. new-call and the description of the received call:

```
<?xml version="1.0"?>
  <caller-id xmlns="..." version="..." state="...">
     <operation-name>new-call</operation-name>
     <call-description>
          <from>SIP_ua_uri</from>
          <to>IP-phone_uri</to>
          <call-id>Call-id </call-id>
          <cseq>CSeq</cseq>
          <contact>Contact</contact>
     </call-description>
  </caller-id>
```

5. Conclusion

The proposed protocol benefits from the existent technologies i.e. SIP and XML. It is subdivided in layers to permit flexibility and adaptability, making possible new advanced collaborative services between UAs devices such as IP-phone and PC. Additionally, enhancing existent services on the PC side can favor user mobility. On the other hand, the increasing popularity of new handheld devices, such as PDAs and IP-enabled cellular phones (dual-mode), can let us think about deployment of the protocol to various third party devices, other than just PCs.

References

[1] D. Box. Essential COM. Addison-Wesley Publishing Company, 1998.

[2] W. Emmerich. Engineering Distributed Objects. John Wiley & Sons, Apr. 2000.

[3] M. Gupta, P. Judge, M. Ammar, "A reputation system for peer-to-peer networks," *International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 144-152, 2004.

[4] R. Enns, Ed., "NETCONF Configuration protocol", draft-ietf-netconf-prot-11, February 2006.

[5] Mark Allman, "AN EVALUATION OF XML-RPC," *ACM SIGMETRICS Performance Evaluation Review,* vol. 30, pp. 2-11, March, 2003

[6] World Wide Web Consortium, "Simple object access protocol (soap) 1.1" URL : http://www.w3.org/TR/SOAP/.

[7] T. Berners-Lee, R. Fielding, and H. Nielsen. Hypertext Transfer Protocol - HTTP/1.0, May 1996. RFC 1945.

[8] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1, Jan. 1997. RFC 2068.

[9] S. Berger, H. Schulzrinne, S. Sidiroglou, X. Wu "Ubiquitous Computing Using SIP," *International Workshop on Network and Operating System Support for Digital Audio and Video*, pp 82-89, 2003

[10] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[11] Roach, A.B., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[12] S. Donovan, "The SIP INFO Method", RFC 2976, October 2000.

[13] RFC 3426 - Session Initiation Protocol (SIP) Extension for Instant Messaging ftp://ftp.rfc-editor.org/in-notes/rfc3428.txt

[14] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, July 2004.

[15] Rosenberg, J., "A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)", RFC 3857, August 2004.

[16] Rosenberg, J., "An Extensible Markup Language (XML) Based Format for Watcher Information", RFC 3858, August 2004.