Advanced Services Architectures for Internet Telephony: A Critical Overview

Roch H. Glitho, Ericsson Research Canada

Abstract

Advanced services are differentiating factors and crucial to service providers' survival and success. Examples are credit card calling, call forwarding, and toll-free calling. In classical telephony's early days, their implementation was embedded in switching software, and this hindered fast deployment. A more modern architecture known as the intelligent network (IN) was born in the 1980s, allowing implementation in separate nodes, resulting in faster deployment of new services. Two tracks are emerging for Internet telephony: one from the ITU-T and the other from the IETF. As far as advanced services are concerned, the ITU-T track offers a rather archaic architecture, reminiscent of the early days of classical telephony. On the other hand, the IETF architecture, although more modern, does have a few pitfalls. There is plenty of room for improvement to both. This article scrutinizes the ITU-T and IETF advanced services architectures for Internet telephony. Salient features are reviewed and weaknesses pinpointed. Although these architectures are constantly evolving, alternatives may emerge. In the conclusion of this article we provide a very brief discussion of two potential alternatives: IN-based architectures and mobile-agent-based architectures.

Internet telephony, defined as real-time voice or multimedia communications over packet-switched networks (PSNs), is far from a novelty. It dates back to the early days of the Internet. The Advanced Research Projects Agency's (ARPA's) Network Secure Communications project implemented an infrastructure for local and transnet real-time voice communications as early as December 1973. The key goal of Network Voice Protocol (NVP), the heart of the implementation, was to demonstrate the feasibility of secure, high-quality, low-bandwidth, real-time two-party phone calls over PSNs [1].

Two sets of standards are emerging today for Internet telephony, the first from the International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) and the second from the Internet Engineering Task Force (IETF). Recommendation H.323 [2] is the principal ITU-T standard for Internet telephony. It is an umbrella standard which refers to many other standards. It was first released in 1996, then subsequently in 1998 and 1999. As it is customary in the industry, we use the term "H.323" to refer to the set of ITU-T standards for Internet telephony, including Recommendation H.323 itself.

The Session Initiation Protocol (SIP) [3] is the principal IETF proposed standard for Internet telephony. It allows the establishment, modification, and termination of multimedia calls. SIP relies on a host of Internet protocols including the Real-Time Protocol (RTP) [4] for data transport. The same applies to H.323, as shown by Fig. 1, which depicts a simplified Internet telephony protocol stack. In this article we also use the term "SIP" to designate the set of IETF specifications for Internet telephony.

While large-scale deployment of Internet telephony systems offering high-quality voice calls still remains a challenge, classical (i.e., circuit-switched) telephony offers today, in addition to high-quality voice calls, a host of related services. In this article we call these *advanced services* or sometimes just *services* to differentiate them from the basic services, meaning those that allow call establishment and termination.

Some examples of advanced services are free phone, split charging, credit card calling, and toll-free calling. Nowadays in classical telephony, these services are created and managed using the intelligent network (IN) architecture. For an overview of IN architecture, we refer the reader to the book by I. Faynberg *et al.* [5].

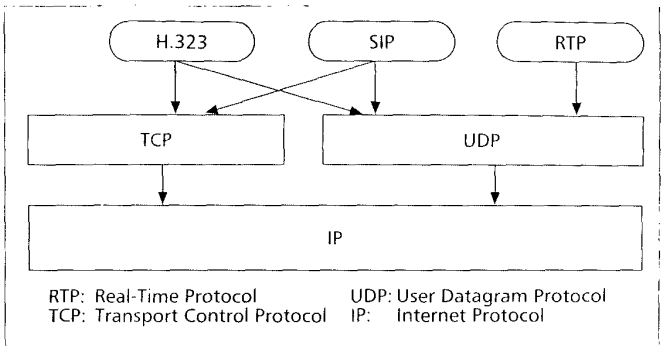
From the end users' perspective, the only (and of course key) advantage Internet telephony has over classical telephony today is the pricing scheme. However, current price models of both classical telephony and Internet telephony are rather awkward. They will evolve and there may not be any significant difference in the future. In order to survive and thrive, Internet telephony will need to offer not only the same high quality for voice calls, but also a set of advanced services which beats (or at least is on par with) what classical telephony offers. Sound architectures are needed for the creation and management of these services.

This article provides a critical overview of H.323 and SIP advanced service architectures. H.323 offers a rather archaic

0890-8044/00/$10.00 © 2000 IEEE

advanced service architecture which follows a "service-by-service" approach reminiscent of classical telephony's early days. The SIP service architecture is more generic and relies to a large extent on more modern technologies. However, it has a few pitfalls. We are indeed still far from the comprehensive architectures needed for creating and managing advanced services in Internet telephony.

We assume the reader is familiar with the basics of Internet telephony. Tutorials have been published on the topic in the recent past [6–9]. Readers with no prior knowledge can consult them. The article starts by deriving a consistent set of architectural requirements. H.323 is the first to undergo our scrutiny. After that, SIP is dealt with. The last section gives a summary and concludes with prospects for the future. Research will certainly burgeon.



■ Figure 1. *A simplified Internet telephony protocol stack.*

Architectural Requirements for Advanced Services in Internet Telephony

Advanced services have life cycles. The concept of life cycle permits the refinement of the activities related to the creation and management of advanced services. The concept of service life cycle has been extensively studied in the context of the Telecommunication Information Network Architecture Consortium (TINA-C) [10]. The overall goal of TINA-C is to define and validate an open architecture for the telecommunication infrastructure, including both network and service infrastructures.

A sound advanced service architecture for Internet telephony must offer a flexible and powerful set of concepts, principles, and rules to support the life cycle. The chief goal of this section is to derive a set of architectural requirements for advanced services in Internet telephony.

The concept of life cycle and related terminology are introduced first because the requirements are based on them. The actual requirements are articulated last. This section is TINA-C flavored. However, we adapt the concepts to Internet telephony whenever appropriate.

The Service Life Cycle

The TINA-C service architecture [11] identifies four phases in the life cycle: construction, deployment, utilization, and withdrawal.

Construction — Service construction allows refinement of the activities related to what is traditionally called *service creation* in telecommunications. It includes service requirements specification, service logic design, and service logic testing. In the rest of this article we stick to the traditional terminology and refer to service construction as service creation.

Deployment and Withdrawal — Service deployment encompasses service planning, installation, and activation (at the network level). In classical telephony, services need to be activated at the network level prior to their activation for specific end users. Service withdrawal encompasses deactivation at the network level and eventual removal from the network. In Internet telephony, services are applications running on either end users' phones or servers residing at the edges of the network. In this case, activation and deactivation are done at the edges of the network, generally at the server level.

Creation phase •Requirements specification •Service logic design •Service logic testing
Deployment •Planning •Installation •Activation
Utilization •Activation/deactivation for specific users •Execution-related signaling
Withdrawal

■ Table 1. *Advanced service life cycle (Internet telephony).*

Utilization — Users need to subscribe to most services in order to use them. These services can be activated or deactivated for them. Activation and deactivation for specific users are part of service utilization. Service utilization also includes service execution. A specific aspect of service execution in Internet telephony worth mentioning is here termed *service-execution-related signaling*.

Service-execution-related signaling comprises the messages exchanged by Internet telephony entities as part of service execution, and also the rules and procedures that govern the exchange. Let us illustrate this with a concrete example: call diversion. Call diversion is a service applied during call establishment. It provides redirection of an incoming call to another destination point. In the context of Internet telephony, it is an application that resides on either end-user phones or network servers. In this example we assume it resides on a network server.

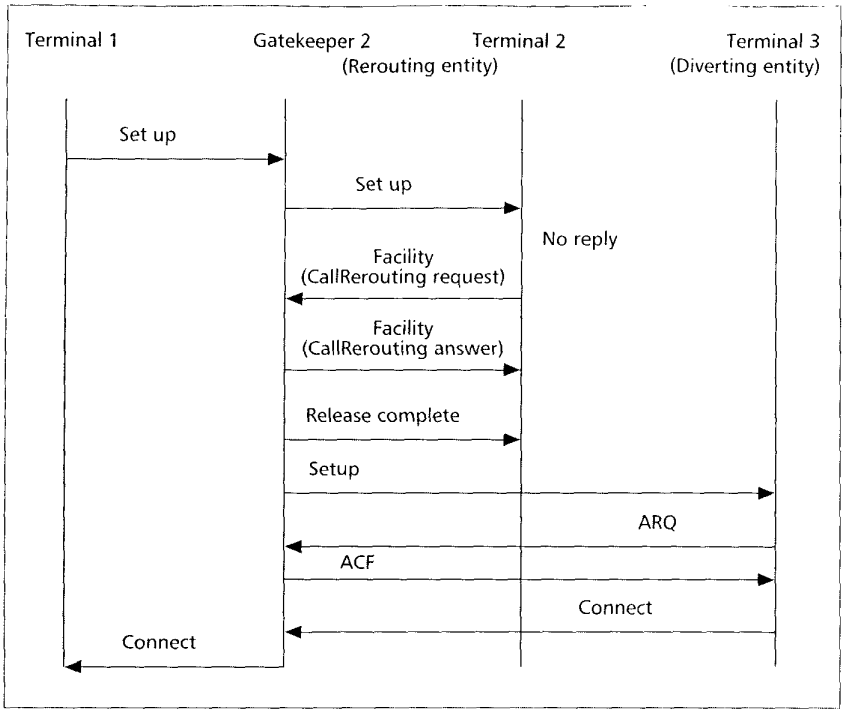
Messages that may be generated by this application in the context of service-execution-related signaling include:

- A message to an application residing on the caller's phone to ask the application to call the number to which the initial call should be diverted. In this case, the caller could be asked by the application whether or not he wants the call to be forwarded to the given number.
- A message to an application residing on another server to ask the server to call the number to which the call should be diverted. In this case, the diversion is usually done transparently to the caller, although the application on the server may send a message to an application on the end-user phone in order to get the caller's view before carrying out the diversion.
- A message to an application residing on the caller's phone in order to inform the caller that his call has been diverted. It should be noted that this is not needed when the caller's opinion was asked before the actual diversion was done.

Service deployment, utilization (except service execution), and withdrawal refine the activities related to what is traditionally known in telecommunications as service management. Table 1 gives a summary. It lists the typical activities encompassed by each phase of the cycle.

Architectural Requirements

The requirements below are rooted in TINA-C. They do target an ideal architecture. However, we find them



■ Figure 2. *An H.323 call forward scenario with the gatekeeper acting as the rerouting entity.*

quite reasonable and believe they are a good starting point for exploring Internet telephony advanced service architectures.

- Support for a wide range of services: The architecture should not be tied to specific services or groups of services. It is of paramount importance that it supports both the old and the new. The old encompasses the IN-based types of services offered by today's classical telephony, while the new refers to new services. Examples of new services are hybrid services. They blend telephony with other Internet technologies (e.g., e-mail log of unsuccessful calls, click to dial). The architecture should also span all phases of the life cycle.
- Rapid service creation and deployment: Rapid service creation is key to success in the marketplace. It is an important way to beat the competition. It should be possible to rapidly specify, design, test, and install both the IN-based and new services mentioned above.
- Tailored services: It should be possible to customize services in order to satisfy the requirements of different users or groups of users. It should be possible for both the user (or group of users) and the service provider to do this customization.
- Independent evolution of services and network infrastructure: Services should not be tied to specific network technology. This means the advanced service architecture should be tied to neither H.323 nor SIP. In concrete terms, it should be possible to migrate from an H.323 network infrastructure to an SIP one (or vice versa) without having to build a new advanced service infrastructure.
- Support for multiplayer environments: There are many parties in the telephony environment: equipment suppliers, network operators, service providers, and end users. The architecture should be open and allow the participation of all these parties. As an example, besides equipment suppliers, service providers and eventually end users should be able to create services.
- Service manageability: It should be easy to deploy, activate, deactivate, and withdraw services.
- Universal access: Users should be able to access services independent of location. They should also be able to access services using any terminal, provided the terminal has the capabilities required by the services.
- Interwork with other advanced service architectures: This implies interworking with not only other Internet telephony advanced service architectures, but also classical telephony advanced service architectures (e.g., IN). While interworking with classical telephony advanced service architectures may be a must for existing service providers who already own a classical telephony advanced service infrastructure, it is not a stringent requirement for new entrants.

Advanced Service Architecture a la H.323

Advanced services in the H.323 world are limited to supplementary services. Supplementary services represent only a fraction of the services offered today by classical telephony. The H.323 advanced service architecture draws quite heavily on the integrated services digital network (ISDN) advanced services architecture [12]. Here, we successively introduce the architecture, illustrate it, and scrutinize it in light of the requirements derived in the previous section.

The Architecture

The architecture focuses on service utilization, especially service-execution-related signaling. The chief concept is the supplementary service control (SS-C) entity. These entities reside within H.323 entities and exchange messages for the support of supplementary services. The messages include service-execution-related signaling messages and messages for activation/deactivation at the user level.

The generic architecture is described in Recommendation H.450.1 [13], while architectures for specific services are described in separate Recommendations. Only six services have been covered so far:

- Call transfer [14]
- Call diversion [15]
- Call hold [16]
- Call park and pickup [17]
- Call waiting [18]
- Message waiting indication [19]

The principles and rules of the generic architecture are described in the next paragraphs. They are illustrated by the concrete example of call diversion in the next subsection.

SS-C entities are defined on a service basis. In other words, every supplementary service has it own set of control entities. The same applies to the messages exchanged by the entities. The messages are either call-independent or call-dependent. When they are call-dependent, the call signaling channel is used for their transportation. When they are not, a call-independent signaling connection is established between the peer SS-C entities for transportation.

Messages for service activation/deactivation at the user level are examples of call-independent messages. Service-execution-related signaling messages are call-dependent. Actions to be taken upon the receipt of these messages are succinctly

described in the specifications. Both normal and exceptional procedures are specified.

An Illustration

There are many flavors of call diversion. Three are described in H.450.3 [15]. In the first, calls to a given subscriber are always diverted provided the service is active; in the second, they are diverted if the line is busy; and in the third, they are diverted if there is no reply after a given time. The first flavor is supplementary service call forward unconditional (SS-CFU), the second supplementary service call forward busy (SS-CFB), and the last supplementary service call forward no reply (SS-CFNR).

The call diversion SS-C entities are the activating, deactivating, diverting (or served), interrogating, last diverting, original diverting (or served), and rerouting endpoints. Some of these terms are self-explanatory, while others are not. The diverting (or served) endpoint is the endpoint where diversion is invoked. Last diverting and original diverting endpoints are specific to calls with successive diversions.

The last diverting point is the diverting (or served) endpoint at any specific stage of a call with successive diversions. The original diverting (or served) endpoint is the diverting (or served) endpoint at the very first diversion. The rerouting entity is the entity that establishes the call to the endpoint to which the call should be rerouted. Interrogating endpoints get from served endpoints information including the activation status of the service and, if activated, the number to which calls are diverted.

Figure 2 depicts a concrete scenario involving a gatekeeper. The gatekeeper is an H.323 entity, and the functionality it provides includes address translation. It can also act as a control point on the network. The following assumptions are made for the scenario:

- Gatekeeper routed call signaling is used.
- SSF-CFNR has been activated for user 2.
- Terminal 2 is the diverting endpoint.
- Terminal 3 is the endpoint to which the call is diverted.
- Gatekeeper 2 is the rerouting entity.

Terminal 2 detects, using unspecified mechanisms, that SSF-CFNR should be executed. It sends an execution-related signaling message to ask the rerouting entity (i.e., gatekeeper 2) to rereroute the call to terminal 3. The gatekeeper acknowledges, using an execution-related signaling message, then releases the connection to terminal 2. After that, the gatekeeper reroutes the call to terminal 3.

Terminal 3 asks admission of the gatekeeper before accepting the call. When admission is granted, it sends a connect message to the gatekeeper, and the gatekeeper forwards it to terminal 1.

The Evaluation

The H.323 advanced service architecture focuses on supplementary services and is reminiscent of the early days of telephony, the pre-IN days. Its chief weakness is the service-by-service approach which offers a very thin generic specification and a rather thick specific specification for each and every service. A grand total of six services have been specified so far. Two more are now being specified:

- .Name identification [20]
- Call completion [21]

The standardization process is lengthy, and the range of services supported at any given time can only be limited. Utilization is the only phase addressed by the architecture, activation/deactivation at the user level and execution-related signaling being part of the H.450.x Recommendation series. Although service creation is not addressed, it may be argued that tools such as Call Processing Language (CPL) [22, 23] currently being specified by IETF could be used.

1. Support for a wide range of services	No
2. Rapid service creation and deployment	No
3. Tailored services	No
4. Independent evolution of services and network infrastructure	Yes (to a large extent)
5. Support fort multi-player environment	No
6. Easy Service manageability	No
7. Universal access	No
8. Interwork with other service architectures	No

■ Table 2. *H.323 advanced service architecture requirements.*

The service creation tools under specification in IETF are discussed later in this article in the context of SIP. It is important to note that all the tools except CPL are geared toward SIP. Although CPL is generic in nature, we think it is fair to state at the present stage that its applicability to H.323 remains to be proven, despite what is claimed in the specifications. To the best of our knowledge there is, for instance, no mechanism currently being specified for uploading CPL scripts to H.323 gatekeepers.

Service logic is developed by equipment suppliers and implicitly embedded in switching software. Neither creation nor development can be rapid. Many nodes will need to be updated for deployment. Furthermore, there is no provision for service customization.

The architecture is to a large extent independent of the network infrastructure. Dedicated messages and procedures are defined in the H.450.x series of Recommendations for the sole control of advanced services. Players other than equipment suppliers have very little room in the H.323 advanced service realm, service logic being implicitly embedded in switching software, as previously stated. Furthermore, no mechanism has been specified to allow other players (e.g., service providers, end users) to create services.

Service manageability will depend on the equipment supplier. While it may be easy in homogeneous and univendor environments, it may become a nightmare in heterogeneous and multivendor environments. Universal access is not addressed. No framework exists today for interworking with other advanced service architectures. Table 2 gives a summary. It shows that of eight requirements, only one is met.

Advanced Service Architecture a la SIP

Advanced (voice-related) services are called standard telephony services in the SIP world. The architecture draws quite heavily on existing Internet tools. It focuses on service creation and service-execution-related signaling. Service creation is discussed first, service-execution-related signaling second. We end the section by evaluating the architecture with the same criteria we used in our evaluation of H.323 advanced service architecture.

Service Creation

Three tools are being specified by IETF for service creation: CPL (already mentioned), SIP common gateway interface (CGI) [24], and SIP Servlet application programming interface

(API) [25]. While the primary target of the first tool is the end user, the other two are aimed at the experienced developer.

In this article we focus on CPL and SIP CGI and discuss them briefly. For an in-depth tutorial we refer the reader to the paper by J. Rosenberg *et al.* [26]. There are significant differences between SIP CGI and SIP Servlet API, although they both target experienced developers. A paper presented at ICIN 2000 [27] gives a brief overview of these differences. We refer the interested reader to it.

In SIP, callers can express preferences about request handling in servers. This can somewhat be assimilated to service creation. We say few words on the topic after having introduced SIP CGI and CPL.

SIP CGI — SIP CGI targets experienced and trusted developers. These developers are supposed to know what they are doing, and there are very few restrictions on the resources available to the service logic they design. This logic can use all information available on the messages received by the server. Furthermore, this same logic can exert a great level of control over the behavior of the server.

SIP CGI is based on HTTP CGI [28], a tool customarily used in the Web for creating dynamic content. We assume the reader is familiar with HTTP CGI. The most important differences between SIP CGI and HTTP CGI are:

- Unlike HTTP CGI, the output of a SIP CGI script is not necessarily the response to be sent by the server. It can be a set of instructions to the server, including instructions for proxying and even generating messages.
- A SIP CGI script is persistent while an HTTP CGI script is not. This stems from the difference mentioned in the previous item. The SIP CGI script does need to process the responses to the requests it has instructed the server to proxy or generate.

The processing of the responses to the messages a SIP CGI script has instructed a server to send can lead the script to send a new set of instructions to the server for proxying or generating another set of messages. A SIP CGI script therefore exerts a great level of control over which messages are sent by a server and when they are sent.

A SIP CGI script also exerts a great level of control over the content of the messages it instructs a server to proxy or generate. By default, the server will follow the rules in the SIP RFC [3]. However, the script can instruct the server to add, delete, or replace specific headers in the message. This applies even to messages that are proxied.

CPL — CPL scripts are generated by end users and uploaded on servers. The use of the REGISTER message has been advocated for transporting these scripts [29]. Since end users are untrusted parties and do not necessarily know what they are doing, a set of stringent constraints have been put on CPL in terms of verifiability, completion, and safety of execution.

CPL represents services as decision graphs. Individual nodes describe either actions to be performed or choices to be made. Nodes may or may not have outputs. When they do, servers follow one of the outputs as a result of the action performed or choice made. Otherwise, servers exit the script.

CPL is based on the Extended Markup Language (XML) [30] and has been designed as a non-Turing-complete language. It provides no method for writing loop and functions. It does not provide ability to access/run external programs. It offers four broad classes of primitives: switch nodes, location nodes, signaling actions, and nonsignaling actions.

Switch nodes are used for matching attributes of the call request or items, independent of the call request, against a list of conditions. Location nodes specify the locations subsequent signaling actions should contact. The most powerful signaling action is proxy. It prompts the server to forward the call to the locations previously specified by a location node primitive. Nonsignaling actions allow user notifications and event logging.

Header Fields — Services can be created using header fields. The set of services that can be created this way is currently limited to services which allow callers to express preferences about how calls should be handled. These header fields are extensions to the core SIP RFC. Two have been specified so far in an Internet draft [31]:

- Reject-Contact
- Accept-Contact

The first header allows the caller to restrict the locations to which his/her call should be forwarded. The second allows the caller to prioritize the location(s) to which the call should be forwarded. In both cases, parameters permit the description of the location not only in terms of URIs but also in terms of characteristics (home, work, fixed phone, etc.).

Although Request-Disposition is specified in the same Internet draft as Reject-Contact and Accept-Contact, we do not consider it a header field for creating services. The use of proxy or redirect servers should be transparent to end users. It is not clear at the present stage how the possibility for expressing preference for proxy or server as allowed by Request-Disposition can be used to build end-user-oriented advanced services.

Service-Execution-Related Signaling

Service-execution-related signaling relies on a toolkit. The kit consists of two types of tools: SIP request methods and header fields. Service-execution-related signaling can be handled in SIP for a very wide range of services. This is done by using the request methods, header fields, or a combination.

The request methods are:

- INVITE
- BYE
- OPTIONS

The header fields used are either integral to the core SIP RFC [3] or specified as extensions to the core RFC. Contact is part of the core specification, while the header fields below have been specified as extensions in the call control services draft [32]:

- Also
- Requested-By
- Replaces

INVITE allows adding new parties to a call, while BYE allows dropping them. OPTIONS does not trigger any specific action in the server. However, the server executes the actions specified by the header field. The semantics of the four header fields used for service-execution-related signaling is elaborated on in the next paragraphs.

Contact is a general header that provides the list of URIs where the user can be reached for further communications. In the context of service-execution-related signaling, it can be used for services such as call diversion. Let us assume that A sends an INVITE to B. The call diversion application residing on B can detect, using unspecified mechanisms, that the call should be forwarded to C. It then generates a response to the application residing on A, with the Contact header containing the URI where C can be reached. A will then send an INVITE directly to C.

Also lists the other participants in a call. It can appear in both requests and responses. Let us illustrate its use by call transfer. The following assumptions are made:

- A call has been established between A and B.
- A decides to transfer the call to C.

A just needs to send a BYE to B with an Also field containing the URI where C can be reached. B will then send an INVITE to C and terminate the call with A.

An INVITE such as the one sent by B to C in the illustration above is known as triggered INVITE in the sense that it has been done on a request from A. The Requested-By header is then used to indicate this. It will contain, in this specific case, the URI where A can be reached.

Replace is used in INVITE only. It contains a list of call legs; upon its receipt, the call legs listed should be abandoned. The receiver should consider that they were never established, and only the new leg established with the INVITE is valid.

1. Support for a wide range of services	Yes
2. Rapid service creation and deployment	Yes
3. Tailored services	No
4. Independent evolution of services and network infrastructure	No
5. Support for a multiplayer environment	Yes
6. Easy service manageability	No
7. Universal access	No
8. Interwork with other service architectures	No

■ Table 3. *SIP advanced service architecture requirements.*

The Evaluation

SIP advanced service architecture relies on modern Internet technologies and offers a great deal of flexibility. The services supported go far beyond the IN types of services offered by today's classical telephony. In today's classical telephony callers cannot even express preferences about how their calls should be handled. SIP advanced service architecture allows this. Furthermore, SIP CGI allows the development of a myriad of new services that blend telephony with Internet applications (e.g., Web, e-mail).

Deployment, activation/deactivation for a specific user, and withdrawal are not explicitly addressed by the architecture. Relevant Internet tools could be used, however, since the architecture relies heavily on Internet technologies. Service creation and deployment may not be as fast as in other Internet services due to telephony specifics, but will certainly be faster than what we are used to in classical telephony.

Although the service creation tools have great potential, it is worth stressing that they are still immature. All SIP service creation specifications, including SIP CGI and CPL, are Internet drafts. There are still many open issues. SIP CGI and CPL scripts cannot yet run on separate and dedicated servers. Feature interaction has not yet been addressed. SIP CGI scripts cannot yet elegantly provide asynchronous directives to servers.

There is no provision for service tailoring. CPL may be used by end users to tailor generic services developed using SIP CGI. This will, however, require interactions between SIP CGI scripts and CPL scripts, and as far as we know the topic has not yet been addressed. The service architecture is tightly coupled to the network infrastructure. They cannot evolve separately. Service-execution-related signaling messages are either plain basic service messages or basic service messages enhanced with dedicated header fields.

There is plenty room for players other than equipment suppliers in the SIP advanced service realm. Service providers and specialized software houses can create services using SIP CGI. Although end users should be allowed in this realm thanks to CPL, we think CPL is still too complex for average end users. It should be stated, though, that this complexity can be hidden to end users by implementing the usage scenarios described in the draft "Call Processing Framework and Requirements" [23].

It will not be easy to manage services. Service scripts are collocated with either network servers or end-user phones. There will be many servers and several end-user phones in any real network. Service logic will then be scattered over them. The situation will certainly improve when the issue of running SIP CGI and CPL scripts on separate and dedicated servers is solved. Manageability will become easier.

As previously stated, service logic resides on either network servers or end-user phones. In the latter case, users cannot have access to their service when they use a different phone in the same domain. In the former case, while they may still be able to access their services when they move within the domain, they cannot access them when they move to a foreign domain. Generally speaking, there is no support for universal access. There is also no general framework for interworking with other advanced services architectures. Table 3 gives a summary. It shows that out of the eight requirements, only three are met.

Summary and Conclusion

In this article we introduce and evaluate the ITU-T H.323 and IETF SIP specifications for advanced services architectures. The evaluation is based on a consistent set of TINA-C flavored requirements. These requirements include support for a wide range of services; rapid creation and deployment of services; service tailoring; independence between service and network infrastructure; support for multiplayer environment; easy manageability of services; universal access to services; and interworking with other services infrastructures.

H.323 advanced service architecture draws heavily on ISDN service architecture and relies on a set of entities which reside in the H.323 nodes and exchange messages for the control of advanced services. It addresses a subset of what is offered by today's classical telephony. It has a major deficiency, the service-by-service approach.

Service creation is not addressed. There is no room for service tailoring, no support for a multiplayer environment, and no support for universal access. Services are not easily managed. There is no standard framework for interworking with other service architectures. It is important to note that network infrastructure and service architecture can evolve separately. There is no tight coupling.

SIP advanced service architecture is very open and draws heavily on Internet technologies. It offers three main tools for service creation: SIP CGI and SIP Servlet API for trusted parties, and CPL for end users. Furthermore, header fields can be used to create services, although the set of services that can be created this way is rather limited for the time being. SIP CGI opens the door to the creation of new services that can blend telephony with the Web and e-mail. The architecture also offers a very flexible framework for service-execution-related signaling.

Service creation and deployment are rapid, and there is support for multiplayer environments. The architecture does lack maturity, however; all the specifications are still at the draft stage. There is support for neither service tailoring nor universal access. Network infrastructure and service architecture cannot evolve separately. Furthermore, services cannot be managed easily, and there is no standard framework for interworking with other advanced service architectures.

Different schemes can be used to associate weights to the

different criteria on which we have based the evaluations made in this article. Furthermore, other criteria could be used, although we consider our criteria quite consistent and reasonable. Anyhow, at the present stage we believe that SIP advanced service architecture will certainly rank higher than its H.323 counterpart for most criteria and weighting schemes. However, there is plenty room for improvement in both.

Predicting the future is a perilous exercise. One, can, however safely forecast that both the old and the new will play a role in the realm of H.323 and SIP advanced services architectures. Researchers will be haunted by the past and solicited by the future. At least two ghosts will emerge from the past: IN and TINA-C. On the novelty side, mobile agents and programmable switches are likely to get on stage.

IN has a large installed base. Business pragmatism may make IN-based advanced service architectures for Internet telephony a reality. The feasibility of these architectures is being studied in the context of the 3rd Generation Partnership Project (3GPP) [33]. An Internet draft was also recently published [34]. For the time being, emphasis is put on reusing the IN infrastructure in its present form. This leads to H.323 gatekeepers and SIP proxy servers mimicking telecommunications switches.

Research in IN-based advanced service architectures will, however, evolve, and evolving IN for meeting Internet telephony needs will soon be on the agenda. IN physical architecture relies on ASN.1-based protocols with Signaling System No. 7 (SS7) transportation. In the context of Internet telephony, text-based protocols with Internet Protocol (IP) transport may be more appropriate. In IN, services are triggered solely by switches residing in the network, because the phones are dumb. In the context of Internet telephony where phones are more intelligent, triggering from end terminals could be contemplated.

Research in mobile-agent-based advanced service architectures for Internet telephony is timely as well, although for a quite different reason. These architectures are currently being investigated for advanced services in classical telephony as evidenced by MARINE, a European Advanced Communications Technologies and Services (ACTS) project on a mobile agent environment for broadband IN services. Interesting articles have also been published on the topic in the recent past [35, 36].

It does make sense to investigate whether mobile-agent-based architectures could be of any help in the context of Internet telephony. They may help in tackling many of the open issues. Universal access, for example, is not supported by either H.323 or SIP. Mobile agents could provide an elegant and network-infrastructure-independent solution to the issue. The same goes for service tailoring.

What are the prospects for advanced service architectures in the Internet telephony realm? Today's ITU-T and IETF architectures are constantly evolving. Alternatives including IN-based and mobile-agent-based architectures may emerge. An interesting issue is the viability of these alternatives. This viability will certainly depend to a large extent on the potential for meeting the consistent set of requirements derived in this article.

Acknowledgments

Many thanks to Prof. Gerald "Chip" Maguire, Royal Institute of Technology Stockholm, Sweden, for his comments on earlier drafts of this article.

References

[1] D. Cohen, "Specifications for the Network Voice Protocol (NVP)," RFC 741, Nov. 1977.
[2] ITU-T Rec. H.323, "Packet Based Multimedia Communications Systems," Geneva, Switzerland, Sept. 1999.
[3] D. Handley *et al.*, "SIP: Session Initiation Protocol," RFC 2543, Mar. 1999.
[4] H. Schulzrinne *et al.*, "RTP: A Transport Protocol for Real Time Application," RFC 1889, Jan. 1996.
[5] I. Faynberg *et al.*, *The Intelligent Networks Standards: Their Applications to Services*, McGraw-Hill, 1997
[6] G. Thom, "H.323: The Multimedia Communications Standard for Local Area Networks," *IEEE Commun. Mag.*, Dec. 1996, vol. 34, no. 12, pp. 52–56.
[7] J. Toga and J. Ott, "ITU-T Standardization Activities for Interactive Multimedia Communications on Packet Networks: H.323 and Related Recommendations," *Computer Networks*, vol. 31, 1999, pp. 205–23.
[8] H. Schulzrinne and J. Rosenberg, "The Session Initiation Protocol: Providing Advanced Services Across the Internet," *Bell Labs Tech. J.*, Oct.–Dec. 1998, vol. 3, pp. 144–60.
[9] H. Schulzrinne and J. Rosenberg, "The IETF Internet Telephony Architecture and Protocols," *IEEE Network*, May/June 1999, pp. 18–23.
[10] TINA-C, http://www.tina-c.com
[11] TINA-C, "Draft Service Architecture, Version 5.0," June 1997.
[12] ITU-T Rec. Q.932, Generic Procedures for the Control of ISDN Supplementary Services, 1993.
[13] ITU-T Rec. H.450.1, "Generic Functional Protocol for the Support of Supplementary Services in H.323," 1998.
[14] ITU-T Rec. H.450.2, "Call Transfer Supplementary Service for H.323," 1998.
[15] ITU-T Rec. H.450.3, "Call Diversion Supplementary Service for H.323," 1998.
[16] ITU-T Rec. H.450.4, "Call Hold Supplementary Service for H.323," 1999.
[17] ITU-T Rec. H.450.5, "Call Park and Pick Up Supplementary Service for H.323," 1999.
[18] ITU-T Rec. H.450.6, "Call Waiting Supplementary Service for H.323," 1999.
[19] ITU-T Rec. H.450.7, "Message Waiting Indication Supplementary Service for H.323," 1999.
[20] ITU-T Draft Rec. H.450.8, "Name Identification Supplementary Service for H.323," 2000.
[21] ITU-T Draft Rec. H.450.9, "Call Completion Supplementary Service for H.323," 2000.
[22] J. Lennox and H. Schulzrinne, "CPL: A Language for User Control of Internet Telephony Services," Internet draft, Mar. 1999, work in progress.
[23] J. Lennox and H. Schulzrinne, "Call Processing Language Framework and Requirements," Internet draft, Jan. 2000, work in progress.
[24] J. Lennox, J. Rosenberg, and H. Schulzrinne, "Common Gateway Interface for SIP," Internet draft, Oct. 1999, work in progress.
[25] A. Kristensen *et al.*, "The SIP Servlet API," Internet draft, Sept. 1999, work in progress.
[26] J. Rosenberg, J. Lennox, and H. Schulzrinne, "Programming Internet Telephony Services," *IEEE Network*, May/June 1999, vol. 13, no. 3, pp. 42–49.
[27] W. Van Leekwijck and D. Brouns, "Siplets: Java based Service Programming for IP Telephony," *ICIN 2000 Proc.*, pp. 22–27.
[28] K. Caar and D. Robinson, "The WWW Common Gateway Interface Version 1.1," Internet draft, Nov. 1998, work in progress.
[29] J. Lennox and H. Schulzrinne, "Transporting User Control Information in SIP REGISTER Payloads," Internet draft, Mar. 1999, work in progress.
[30] I. T. Bray, J. Paoli and C.-M. Sperbeg-McQueen, "Extensible Markup Language (XML) 1.0," W3C REC-xml-19980210, Feb. 1998.
[31] H. Schulzrinne and J. Rosenberg, "SIP Caller Preferences and Callee Capabilities," Internet draft, Feb. 1999, work in progress.
[32] H. Schulzrinne and J. Rosenberg, "SIP Call Control Services," Internet draft, June 1999, work in progress.
[33] 3rd Generation Partnership Project (3GPP), "Technical Specification Group Core Network; Feasibility Technical Report — CAMEL Control of VoIP Services," 3G TR 21.978 v. 2.1.0, Jan. 2000.
[34] V. Gurbani, "Accessing IN Services from SIP Networks," Internet draft, June 1999, work in progress.
[35] M. Breugst and T. Magedanz, Mobile Agents - Enabling Technology for Active Networks Implementation, *IEEE Network*, May/June 1998, vol. 12, no. 3, pp. 53–60.
[36] L. Hagen *et al.*, "Impacts of Mobile Agents Technology on Mobile Communication System Evolution," *IEEE Pers. Commun.*, Aug. 1998, vol. 5, no. 4, pp. 56–69.

Biography

ROCH H. GLITHO [SM] (roch.glitho@lmc.ericsson.se) received M.Sc. degrees in business economics (University of Grenoble, France), pure mathematics (University of Geneva, Switzerland), and computer science (University of Geneva, Switzerland). He is currently pursuing a Ph.D. in computer communications at the Royal Institute of Technology, Stockholm, Sweden, while working as a principal engineer at Ericsson Research, Montreal, Canada. His research interests include Internet telephony, service engineering, network management, and signaling. He joined Ericsson Telecom in Stockholm in 1990, after having worked five years for a computer manufacturer in Oslo, Norway, then moved to Ericsson Research in Montreal, Canada, in 1993. He is Editor-In-Chief of *IEEE Communications Surveys & Tutorials* (http://www.comsoc.org/pubs/surveys/), and serves as technical editor for *IEEE Communications Magazine* and the *Journal of Network and Systems Management* (JNSM). He holds many patents and has published several articles in referred journals.