Internet Engineering Task Force (IETF)
Request for Comments: 5897
Category: Informational
ISSN: 2070-1721

J. Rosenberg
jdrosen.net

June 2010

Identification of Communications Services in the Session Initiation Protocol (SIP)

Abstract

This document considers the problem of service identification in the Session Initiation Protocol (SIP). Service identification is the process of determining the user-level use case that is driving the signaling being utilized by the user agent (UA). This document discusses the uses of service identification, and outlines several architectural principles behind the process. It identifies perils when service identification is not done properly -- including fraud, interoperability failures, and stifling of innovation. It then outlines a set of recommended practices for service identification.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5897.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [RFC3261] defines mechanisms for initiating and managing communications sessions between agents. SIP allows for a broad array of session types between agents. It can manage audio sessions, ranging from low-bitrate voice-only up to multi-channel high-fidelity music. It can manage video sessions, ranging from small, "talking-head" style video chat, up to high-definition multipoint video conferencing and ranging from low-bandwidth user-generated content, up to high-definition movie and TV content. SIP endpoints can be anything -- adaptors that convert an old analog telephone to Voice over IP (VoIP), dedicated hardphones, fancy hardphones with rich displays and user entry capabilities, softphones on a PC, buddy-list and presence applications on a PC, dedicated videoconferencing peripherals, and speakerphones.

This breadth of applicability is SIP's greatest asset, but it also introduces numerous challenges. One of these is that, when an endpoint generates a SIP INVITE for a session, or receives one, that session can potentially be within the context of any number of different use cases and endpoint types. For example, a SIP INVITE with a single audio stream could represent a Push-To-Talk session between mobile devices, a VoIP session between softphones, or audio-based access to stored content on a server.

Each of these different use cases represents a different service. The service is the user-visible use case that is driving the behavior of the user agents and servers in the SIP network.

The differing services possible with SIP have driven implementors and system designers to seek techniques for service identification. Service identification is the process of determining and/or signaling the specific use case that is driving the signaling being generated by a user agent. At first glance, this seems harmless and easy enough. It is tempting to define a new header, "Service-ID", for example, and have a user agent populate it with any number of well-known tokens that define what the service is. It could then be consumed for any number of purposes. A token placed into the signaling for this purpose is called a service identifier.

Service identification and service identifiers, when used properly, can be beneficial. However, when done improperly, service identification can lead to fraud, systemic interoperability failures, and a complete stifling of the innovation that SIP was meant to achieve. The purpose of this document is to describe service identification in more detail and describe how these problems arise.

Section 2 begins by defining a service and the service identification problem. Section 3 gives some concrete examples of services and why they can be challenging to identify. Section 4 explores the ways in which a service identification can be utilized within a network. Next, Section 5 discusses the key architectural principles of service identification. Section 6 describes what declarative service invocation is, and how it can lead to fraud, interoperability failures, and stifling of service innovation.

Consequently, this document concludes that declarative service identification -- the process by which a user agent inserts a moniker into a message that defines the desired service, separate from explicit and well-defined protocol mechanisms -- is harmful.

Instead of performing declarative service identification, this document recommends derived service identification, and gives several recommendations around it in Section 7:

1. The identity of a service should always be derived from the explicit signaling in the protocol messages and other contextual information, and never indicated by the user through a separate identifier placed into the message.

2. The process of service identification based on signaling messages must be designed to SIP's negotiative expressiveness, and therefore handle heterogeneity and not assume a fixed set of use cases.

3. Presence can help in providing URIs that can be utilized to connect to specific services, thereby creating explicit indications in the signaling that can be used to derive a service identity.

4. Service identities placed into signaling messages for the purposes of caching the service identity are strictly for intra-domain usage.

5. Device dispatch should be based on feature tags that map to well-defined SIP extensions and capabilities. Service dispatch should not be based on abstract service identifiers.

2. Services and Service Identification

The problem of identifying services within SIP is not a new one. The problem has been considered extensively in the context of presence. In particular, the presence data model for SIP [RFC4479] defines the concept of a service as one of the core notions that presence describes. Services are described in Section 3.3 of RFC 4479.

Essentially, the service is the user-visible use case that is driving the behavior of the user agents and servers in the SIP network. Being user-visible means that there is a difference in user experience between two services that are different. That user experience can be part of the call, or outside of the call. Within a call, the user experience can be based on different media types (an audio call vs. a video chat), different content within a particular media type (stored content, such as a movie or TV session), different devices (a wireless device for "telephony" vs. a PC application for "voice chat"), different user interfaces (a buddy-list view of voice on a PC application vs. a software emulation of a hardphone), different communities that can be accessed (voice chat with other users that have the same voice chat client vs. voice communications with any endpoint on the Public Switched Telephone Network (PSTN)), or different applications that are invoked by the user (manually selecting a Push-To-Talk application from a wireless phone vs. a telephony application). Outside of a call, the difference in user experience can be a billing one (cheaper for one service than another), a notification feature for one and not another (for example, an IM that gets sent whenever a user makes a call), and so on.

In some cases, there is very little difference in the underlying technology that will support two different services, and in other cases, there are big differences. However, for the purposes of this discussion, the key definition is that two services are distinct when there is a perceived difference by the user in the two services.

This leads naturally to the desire to perform service identification. Service identification is defined as the process of:

1. determining the underlying service that is driving a particular signaling exchange,

2. associating that service with a service identifier, and

3. attaching that moniker to a signaling message (typically a SIP INVITE).

Once service identification is performed, the service identifier can then be used for various purposes within the network. Service identification can be done in the endpoints, in which case the UA would insert the moniker directly into the signaling message based on its awareness of the service. Or, it can be done within a server in the network (such as a proxy), based on inspection of the SIP message, or based on hints placed into the message by the user.

When service identification is performed entirely by inspecting the signaling, this is called derived service identification. When it is done based on knowledge possessed only by the invoking user agent, it is called declarative service identification. Declarative service identification can only be done in user agents, by definition.

3. Example Services

It is very useful to consider several example services, especially ones that appear difficult to differentiate from each other. In cases where it is hard to differentiate, service identification -- and in particular, declarative service identification -- appears highly attractive (and indeed, required).

3.1. IPTV vs. Multimedia

IP Television (IPTV) is the usage of IP networks to access traditional television content, such as movies and shows. SIP can be utilized to establish a session to a media server in a network, which then serves up multimedia content and streams it as an audio and video stream towards the client. Whether SIP is ideal for IPTV is, in itself, a good question. However, such a discussion is outside the scope of this document.

Consider multimedia conferencing. The user accesses a voice and video conference at a conference server. The user might join in listen-only mode, in which case the user receives audio and video streams, but does not send.

These two services -- IPTV and listen-only multimedia conferencing -- clearly appear as different services. They have different user experiences and applications. A user is unlikely to ever be confused about whether a session is IPTV or listen-only multimedia conferencing. Indeed, they are likely to have different software applications or endpoints for the two services.

However, these two services look remarkably alike based on the signaling. Both utilize audio and video. Both could utilize the same codecs. Both are unidirectional streams (from a server in the network to the client). Thus, it would appear on the surface that there is no way to differentiate them, based on inspection of the signaling alone.

3.2. Gaming vs. Voice Chat

Consider an interactive game, played between two users from their mobile devices. The game involves the users sending each other game moves, using a messaging channel, in addition to voice. In another service, users have a voice and IM chat conversation using a buddy-list application on their PC.

In both services, there are two media streams -- audio and messaging. The audio uses the same codecs. Both use the Message Session Relay Protocol (MSRP) [RFC4975]. In both cases, the caller would send an INVITE to the Address of Record (AOR) of the target user. However, these represent fairly different services, in terms of user experience.

3.3. Gaming vs. Voice Chat #2

Consider a variation on the example in Section 3.2. In this variation, two users are playing an interactive game between their phones. However, the game itself is set up and controlled using a proprietary mechanism -- not using SIP at all. However, the client application allows the user to chat with their opponent. The chat session is a simple voice session set up between the players.

Compare this with a basic telephone call between the two users. Both involve a single audio session. Both use the same codecs. They appear to be identical. However, different user experiences are needed. For example, we desire traditional telephony features (such as call forwarding and call screening) to be applied in the telephone service, but not in the gaming chat service.

3.4. Configuration vs. Pager Messaging

The SIP MESSAGE method [RFC3428] provides a way to send one-shot messages to a particular AOR. This specification is primarily aimed at Short Message Service (SMS)-style messaging, commonly found in wireless phones. Receipt of a MESSAGE request would cause the messaging application on a phone to launch, allowing the user to browse the message history and respond.

However, a MESSAGE request is sometimes used for the delivery of content to a device for other purposes. For example, some providers use it to deliver configuration updates, such as new phone settings or parameters, or to indicate that a new version of firmware is available. Though not designed for this purpose, the MESSAGE method gets used since, in existing wireless networks, SMS is used for this purpose, and the MESSAGE request is the SIP equivalent of SMS.

Consequently, the MESSAGE request sent to a phone can be for two different services. One would require invocation of a messaging app, whereas the other would be consumed by the software in the phone, without any user interaction at all.

4. Using Service Identification

It is important to understand what the service identity would be utilized for, if known. This section discusses the primary uses. These are application invocation in user agents and the network, Quality of Service authorization, service authorization, accounting and billing, service negotiation, and device dispatch.

4.1. Application Invocation in the User Agent

In some of the examples above, there were multiple software applications executing on the host. One common way of achieving this is to utilize a common SIP user agent implementation that listens for requests on a single port. When an incoming INVITE or MESSAGE arrives, it must be delivered to the appropriate application software. When each service is bound to a distinct software application, it would seem that the service identity is needed to dispatch the message to the appropriate piece of software. This is shown in Figure 1.

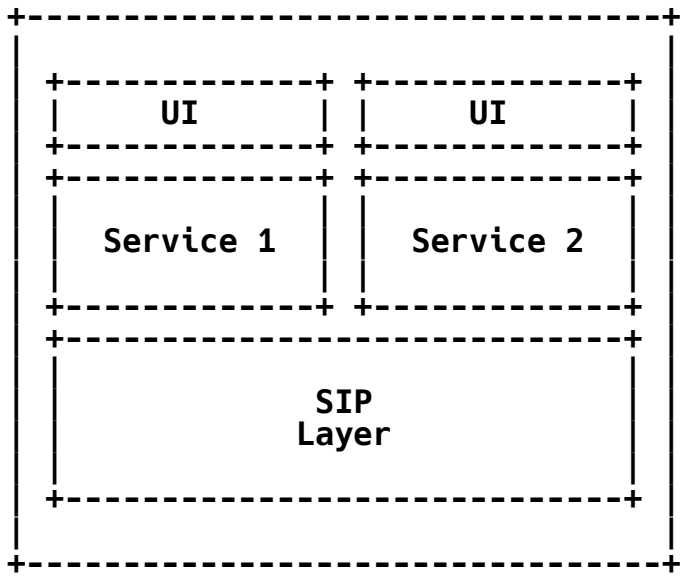

Figure 1

The role of the SIP layer is to parse incoming messages, handle the SIP state machinery for transactions and dialogs, and then dispatch requests to the appropriate service. This software architecture is analogous to the way web servers frequently work. An HTTP server listens on port 80 for requests, and based on the HTTP Request-URI, dispatches the request to a number of disparate applications. The same is happening here. For the example services in Section 3.2, an incoming INVITE for the gaming service would be delivered to the gaming application software. An incoming INVITE for the voice chat service would be delivered to the voice chat application software. The example in Section 3.3 is similar. For the examples in Section 3.4, a MESSAGE request for user-to-user messaging would be delivered to the messaging or SMS app, and a MESSAGE request containing configuration data would be delivered to a configuration update application.

Unlike the web, however, in all three use cases, the user initiating communications has (or appears to have -- more below) only a single identifier for the recipient -- their AOR. Consequently, the SIP Request-URI cannot be used for dispatching, as it is identical in all three cases.

4.2. Application Invocation in the Network

Another usage of a service identifier would be to cause servers in the SIP network to provide additional processing, based on the service. For example, an INVITE issued by a user agent for IPTV would pass through a server that does some kind of content rights management, authorizing whether the user is allowed to access that content. On the other hand, an INVITE issued by a user for multimedia conferencing would pass through a server providing "traditional" telephony features, such as outbound call screening and call recording. It would make no sense for the INVITE associated with IPTV to have outbound call screening and call recording applied, and it would make no sense for the multimedia conferencing INVITE to be processed by the content rights management server. Indeed, in these cases, it's not just an efficiency issue (invoking servers when not needed), but rather, truly incorrect behavior can occur. For example, if an outbound call screening application is set to block outbound calls to everything except for the phone numbers of friends and family, an IPTV request that gets processed by such a server would be blocked (as it's not targeted to the AOR of a friend or family member). This would block a user's attempt to access IPTV services, when that was not the goal at all.

Similarly, a MESSAGE request as described in Section 3.4 might need to pass through a message server for filtering when it is associated with chat, but not when it is associated with a configuration update.

Consider a filter that gets applied to MESSAGE requests, and that filter runs in a server in the network. The filter operation prevents user Joe from sending messages to user Bob that contain the words "stock" or "purchase", due to some regulations that disallow Joe and Bob from discussing stock trading. However, a MESSAGE for configuration purposes might contain an XML document that uses the token "stock" as some kind of attribute. This configuration update would be discarded by the filtering server, when it should not have been.

4.3. Network Quality-of-Service Authorization

The IP network can provide differing levels of Quality of Service (QoS) to IP packets. This service can include guaranteed throughput, latency, or loss characteristics. Typically, the user agent will make some kind of QoS request, either using explicit signaling protocols (such as the Resource ReSerVation Protocol (RSVP) [RFC2205]) or through marking of a Diffserv value in packets. The network will need to make a policy decision based on whether or not these QoS treatments are authorized. One common authorization policy is to check if the user has invoked a service using SIP that they are authorized to invoke, and that this service requires the level of QoS treatment the user has requested.

For example, consider IPTV and multimedia conferencing as described in Section 3.1. IPTV is a non-real-time service. Consequently, media traffic for IPTV would be authorized for bandwidth guarantees, but not for latency or loss guarantees. On the other hand, multimedia conferencing is in real time. Its traffic would require bandwidth, loss, and latency guarantees from the network.

Consequently, if a user should make an RSVP reservation for a media stream, and ask for latency guarantees for that stream, the network would choose to be able to authorize it if the service was multimedia conferencing, but not if it was IPTV. This would require the server performing the QoS authorization to know the service associated with the INVITE that set up the session.

4.4. Service Authorization

Frequently, a network administrator will want to authorize whether a user is allowed to invoke a particular service. Not all users will be authorized to use all services that are provided. For example, a user may not be authorized to access IPTV services, whereas they are authorized to utilize multimedia processing. A user might not be able to utilize a multiplayer gaming service, whereas they are authorized to utilize voice chat services.

Consequently, when an INVITE arrives at a server in the network, the server will need to determine what the requested service is, so that the server can make an authorization decision.

4.5. Accounting and Billing

Service authorization and accounting/billing go hand in hand. One of the primary reasons for authorizing that a user can utilize a service is that they are being billed differently based on the type of service. Consequently, one of the goals of a service identity is to be able to include it in accounting records, so that the appropriate billing model can be applied.

For example, in the case of IPTV, a service provider can bill based on the content (US $5 per movie, perhaps), whereas for multimedia conferencing, they can bill by the minute. This requires the accounting streams to indicate which service was invoked for the particular session.

4.6. Negotiation of Service

In some cases, when the caller initiates a session, they don't actually know which service will be utilized. Rather, they might choose to offer up all of the services they have available to the called party, and then let the called party decide, or let the system make a decision based on overlapping service capabilities.

As an example, a user can do both the game and the voice chat service described in Section 3.2. The user initiates a session to a target AOR, but the devices used by the target can only support voice chat. The called device returns, in its call acceptance, an indication that only voice chat can be used. Consequently, voice chat gets utilized for the session.

4.7. Dispatch to Devices

When a user has multiple devices, each with varying capabilities in terms of service, it is useful to dispatch an incoming request to the right device based on whether the device can support the service that has been requested.

For example, if a user initiates a gaming session with voice chat, and the target user has two devices -- one that can support the gaming service, and another that cannot -- the INVITE should be dispatched to the device that supports the gaming session.

5. Key Principles of Service Identification

In this section, we describe several key principles of service identification:

1. Services are a by-product of signaling
2. Identical signaling produces identical services
3. Declarative service identification is an example of "Do What I Mean" (DWIM)
4. Declarative service identifiers are redundant
5. URIs are a key mechanism for producing differentiated signaling

5.1. Services Are a By-Product of Signaling

Declarative service identification -- the addition of a service identifier by clients in order to inform other entities of what the service is -- is a very compelling solution to solving the use cases described above. It provides a clear way for each of the use cases to be differentiated. On the other hand, derived service identification appears "hard", since the signaling appears to be the same for these different services.

Declarative service identification misses a key point, which cannot be stressed enough, and which represents the core architectural principle to be understood here:

> A service is the byproduct of the signaling and the context around it (the user profile, time of day, and so on) -- the effects of the signaling message once it is launched into the network. The service identity is therefore always derivable from the signaling and its context without additional identifiers. In other words, derived service identification is always possible when signaling is being properly handled.

When a user sends an INVITE request to the network and targets that request at an IPTV server, and includes the Session Description Protocol (SDP) for audio and video streaming, the *result* of sending such an INVITE is that an IPTV session occurs. The entire purpose of the INVITE is to establish such a session, and therefore, invoke the service. Thus, a service is not something that is different from the rest of the signaling message. A service is what the user gets after the network and other user agents have processed a signaling message.

It may seem that delayed offers (SIP INVITE requests that lack SDP) make it impossible to perform derived service identification. After all, in some of the cases above, the differentiation was done using the SDP in the request. What if it's not there? The answer is simple -- if it's not there, and the SDP is being offered by the called party, you cannot in fact know the service at the time of the INVITE. That's the whole point of delayed offer -- to give the called party the chance to offer up what it wants for the session. In cases where service identification is needed at request time, delayed offer cannot be used.

5.2. Identical Signaling Produces Identical Services

This principle is a natural conclusion of the previous assertion. If a service is the byproduct of signaling, how can a user have different experiences and different services when the signaling message is the same? They cannot.

But how can that be? From the examples in Section 3, it would seem that there are services that are different, but have identical signaling. If we hold true to the assertion, there is in fact only one logical conclusion:

> If two services are different, but their signaling appears to be the same, it is because one or more of the following is true:
>
> 1. there is in fact something different that has been overlooked
>
> 2. something has been implied from the signaling, when in fact it should have been signaled explicitly
>
> 3. the signaling mechanism should be changed so that there is, in fact, something that is different

To illustrate this, let us take each of the example services in Section 3 and investigate whether there is, or should be, something different in the signaling in each case.

IPTV vs. Multimedia Conferencing: The two services described in Section 3.1 appear to have identical signaling. They both involve audio and video streams, both of which are unidirectional. Both might utilize the same codecs. However, there is another important difference in the signaling -- the target URI. In the case of IPTV, the request is targeted at a media server or to a particular piece of content to be viewed. In the case of multimedia conferencing, the target is a conference server. The administrator of the domain can therefore examine the Request-URI

and figure out whether it is targeted for a conference server or a content server, and use that to derive the service associated with the request.

Gaming vs. Voice Chat: Though both sessions involve MSRP and voice, and both are targeted to the same AOR of the called user, there is a difference. The MSRP messages for the gaming session carry content that is game specific, whereas the MSRP messages for the voice chat are just regular text, meant for rendering to a user. Thus, the MSRP session in the SDP will indicate the specific content type that MSRP is carrying, and this type will differ in both cases. Even if the game moves look like text, since they are being consumed by an automata, there is an underlying schema that dictates their content, and therefore, this schema represents the actual content type that should be signaled.

Gaming vs. Voice Chat #2: In this case, both sessions involve only voice, and both are targeted at the same AOR. Indeed, there truly is nothing different -- if indeed the signaling works this way. However, there is an alternative mechanism for performing the signaling. For the gaming session, the proprietary protocol can be used to exchange a URI that can be used to identify the voice chat function on the phone that is associated with the game (for example, a Globally Routable User Agent URI (GRUU) can be used [RFC5627]). Indeed, the gaming chat is not targeting the USER -- it's targeting the gaming instance on the phone. Thus, if a special GRUU is used for the gaming chat, this makes the signaling different between these two services.

Configuration vs. Pager Messaging: Just as in the case of gaming vs. voice chat, the content type of the messages differentiates the service that occurs as a consequence of the messages.

5.3. Do What I Say, Not What I Mean

"Do What I Mean", abbreviated as DWIM, is a concept in computer science. It is sometimes used to describe a function that tries to intelligently guess at what the user intended. It is in contrast to "Do What I Say", or DWIS, which describes a function that behaves concretely based on the inputs provided. Systems built on the DWIM concept can have unexpected behaviors, because they are driven by unstated rules.

Declarative service identification is an example of DWIM. The service identifier has no well-defined impact on the state machinery or protocols in the system; it has various side effects based on an assumption of what is meant by the service identifier. Derived service identification, on the other hand, is an expression of the

principle of DWIS -- the behavior of the system is based entirely on the specifics of the protocol and are well defined by the protocol specification. The service identifier is just a shorthand for summarizing things that are well defined by signaling.

As a litmus test to differentiate the two cases, consider the following question. If a request contained a service identifier, and that request were processed by a domain that didn't understand the concept of service identifiers at all, would the request be rejected if that service were not supported, or would it complete but do the wrong thing? If it is the latter case, it's DWIM. If it's the former, it's DWIS.

5.4. Declarative Service Identifiers Are Redundant

Because a declarative service identifier is, by definition, inside of the signaling message, and because the signaling itself completely defines the behavior of the service, another natural conclusion is that a declarative service identifier is redundant with the signaling itself. It says nothing that could not or should not otherwise be derived from examination of the signaling.

5.5. URIs Are Key for Differentiated Signaling

In the IPTV example and in the second gaming example, it was ultimately the Request-URI that was (or should be) different between the two services. This is important. In many cases where services appear the same, it is because the resource that is being targeted is not, in fact, the user. Rather, it is a resource that is linked with the user. This resource might be an instance of a software application on the particular device of a user, or a resource in the network that acts on behalf of the user.

The Request-URI is an infinitely large namespace for identifying these resources. It is an ideal mechanism for providing differentiation when there would otherwise be none.

Returning again to the example in Section 3.3, we can see that it does make more sense to target the gaming chat session at a software instance on the user's phone, rather than at the user themselves. The gaming chat session should really only go to the phone on which the user is playing the game. The software instance does indeed live only on that phone, whereas the user themselves can be contacted in many ways. We don't want telephony features invoked for the gaming chat session, because those features only make sense when someone is trying to communicate with the USER. When someone is trying to

communicate with a software instance that acts on behalf of the user, a different set of rules apply, since the target of the request is completely different.

6. Perils of Declarative Service Identification

Based on these principles, several perils of declarative service identification can be described. They are:

1. Declarative service identification can be used for fraud

2. Declarative service identification can hurt interoperability

3. Declarative service identification can stifle service innovation

6.1. Fraud

Declarative service identification can lead to fraud. If a provider uses the service identifier for billing and accounting purposes, or for authorization purposes, it opens an avenue for attack. The user can construct the signaling message so that its actual effect (which is the service the user will receive), is what the user desires, but the user places a service identifier into the request (which is what is used for billing and authorization) that identifies a cheaper service, or one that the user is not authorized to receive. In such a case, the user will receive service, and not be billed properly for it.

If, however, the domain administrator derived the service identifier from the signaling itself (derived service identification), the user cannot lie. If they did lie, they wouldn't get the desired service.

Consider the example of IPTV vs. multimedia conferencing. If multimedia conferencing is cheaper, the user could send an INVITE for an IPTV session, but include a service identifier that indicates multimedia conferencing. The user gets the service associated with IPTV, but at the cost of multimedia conferencing.

This same principle shows up in other places -- for example, in the identification of an emergency services call [ECRIT-FRAMEWORK]. It is desirable to give emergency services calls special treatment, such as being free and authorized even when the user cannot otherwise make calls, and to give them priority. If emergency calls were indicated through something other than the target of the call being an emergency services URN [RFC5031], it would open an avenue for fraud. The user could place any desired URI in the request-URI, and indicate separately, through a declarative identifier, that the call is an emergency services call. This would then get special treatment but

of course would get routed to the target URI. The only way to prevent this fraud is to consider an emergency call as any call whose target is an emergency services URN. Thus, the service identification here is based on the target of the request. When the target is an emergency services URN, the request can get special treatment. The user cannot lie, since there is no way to separately indicate that this is an emergency call, besides targeting it to an emergency URN.

6.2. Systematic Interoperability Failures

How can declarative service identification cause loss of interoperability? When an identifier is used to drive functionality -- such as dispatch on the phones, in the network, or QoS authorization -- it means that the wrong thing can happen when this field is not set properly. Consider a user in domain 1, calling a user in domain 2. Domain 1 provides the user with a service they call "voice chat", which utilizes voice and IM for real-time conversation, driven off of a buddy-list application on a PC. Domain 2 provides their users with a service they call "text telephony", which is a voice service on a wireless device that also allows the user to send text messages. Consider the case where domain 1 and domain 2 both have their user agents insert a service identifier into the request, and then use that to perform QoS authorization, accounting, and invocation of applications in the network and in the device. The user in domain 1 calls the user in domain 2, and inserts the identifier "Voice Chat" into the INVITE. When this arrives at the server in domain 2, the service identifier is unknown. Consequently, the request does not get the proper QoS treatment, even if the call itself will succeed.

If, on the other hand, derived service identification were used, the service identifier could be removed by domain 2, and then recomputed based on the signaling to match its own notion of services. In this case, domain 2 could derive the "text telephony" identifier, and the request completes successfully.

Declarative service identification, used between domains, causes interoperability failures unless all interconnected domains agree on exactly the same set of services and how to name them. Of course, lack of service identifiers does not guarantee service interoperability. However, SIP was built with rich tools for negotiation of capabilities at a finely granular level. One user agent can make a call using audio and video, but if the receiving UA only supports audio, SIP allows both sides to negotiate down to the lowest common denominator. Thus, communication is still provided. As another example, if one agent initiates a Push-To-Talk session (which is audio with a companion floor control mechanism), and the

other side only did regular audio, SIP would be able to negotiate back down to a regular voice call. As another example, if a calling user agent is running a high-definition video conferencing endpoint, and the called user agent supports just a regular video endpoint, the codecs themselves can negotiate downward to a lower rate, picture size, and so on. Thus, interoperability is achieved. Interestingly, the final "service" may no longer be well characterized by the service identifier that would have been placed in the original INVITE. For example, in this case, if the original INVITE from the caller had contained the service identifier "hi-fi video", but the video gets negotiated down to a lower rate and picture size, the service identifier is no longer really appropriate. That is why services need to be derived by signaling -- because the signaling itself provides negotiation and interoperability between different domains.

This illustrates another key aspect of the interoperability problem. Declarative service identification will result in inconsistencies between its service identifiers and the results of any SIP negotiation that might otherwise be applied in the session.

When a service identifier becomes something that both proxies and the user agent need to understand in order to properly treat a request (which is the case for declarative service identification), it becomes equivalent to including a token in the Proxy-Require and Require header fields of every single SIP request. The very reason that [RFC4485] frowns upon usage of Require and certainly Proxy-Require is the huge impact on interoperability it causes. It is for this same reason that declarative service identification needs to be avoided.

6.3. Stifling of Service Innovation

The probability that any two service providers end up with the same set of services, and give those services the same names, becomes smaller and smaller as the number of providers grow. Indeed, it would almost certainly require a centralized authority to identify what the services are, how they work, and what they are named. This, in turn, leads to a requirement for complete homogeneity in order to facilitate interconnection. Two providers cannot usefully interconnect unless they agree on the set of services they are offering to their customers and each do the same thing. This is because each provider has become dependent on inclusion of the proper service identifier in the request, in order for the overall treatment of the request to proceed correctly. This is, in a very real sense, anathema to the entire notion of SIP, which is built on the idea that heterogeneous domains can interconnect and still get interoperability.

Declarative service identification leads to a requirement for homogeneity in service definitions across providers that interconnect, ruining the very service heterogeneity that SIP was meant to bring.

Indeed, Metcalfe's Law says that the value of a network grows with the square of the number of participants. As a consequence of this, once a bunch of large domains did get together, agree on a set of services, and then agree on a set of well-known identifiers for those services, it would force other providers to also deploy the same services, in order to obtain the value that interconnection brings. This, in turn, will stifle innovation, and quickly force the set of services in SIP to become fixed and never expand beyond the ones initially agreed upon. This, too, is anathema to the very framework on which SIP is built, and defeats much of the purpose of why providers have chosen to deploy SIP in their own networks.

Consider the following example. Several providers get together and standardize on a bunch of service identifiers. One of these uses audio and video (say, "multimedia conversation"). This service is successful and is widely utilized. Endpoints look for this identifier to dispatch calls to the right software applications, and the network looks for it to invoke features, perform accounting, and provide QoS. A new provider gets the idea for a new service (say, "avatar-enhanced multimedia conversation"). In this service, there is audio and video, but there is a third stream, which renders an avatar. A caller can press buttons on their phone, to cause the avatar on the other person's device to show emotion, make noise, and so on. This is similar to the way emoticons are used today in IM. This service is enabled by adding a third media stream (and consequently, a third m-line) to the SDP.

Normally, this service would be backwards-compatible with a regular audio-video endpoint, which would just reject the third media stream. However, because a large network has been deployed that is expecting to see the token, "multimedia conversation" and its associated audio+video service, it is nearly impossible for the new provider to roll out this new service. If they did, it would fail completely, or partially fail, when their users call users in other provider domains.

7. Recommendations

From these principles, several recommendations can be made.

7.1. Use Derived Service Identification

Derived service identification -- where an identifier for a service is obtained by inspection of the signaling and of other contextual data (such as subscriber profile) -- is reasonable, and when done properly, does not lead to the perils described above. However, declarative service identification -- where user agents indicate what the service is, separate from the rest of the signaling -- leads to the perils described above.

If it appears that the signaling currently defined in standards is not sufficient to identify the service, it may be due to lack of sufficient signaling to convey what is needed, or may be because request URIs should be used for differentiation and they are not being used. By applying the litmus tests described in Section 5.3, network designers can determine whether or not the system is attempting to perform declarative service identification.

7.2. Design for SIP's Negotiative Expressiveness

One of SIP's key strengths is its ability to negotiate a common view of a session between participants. This means that the service that is ultimately received can vary wildly, depending on the types of endpoints in the call and their capabilities. Indeed, this fact becomes even more evident when calls are set up between domains.

As such, when performing derived service identification, domains should be aware that sessions may arrive from different networks and different endpoints. Consequently, the service identification algorithm must be complete -- meaning it computes the best answer for any possible signaling message that might be received and any session that might be set up.

In a homogeneous environment, the process of service identification is easy. The service provider will know the set of services they are providing, and based on the specific call flows for each specific service, can construct rules to differentiate one service from another. However, when different providers interconnect, or when different endpoints are introduced, assumptions about what services are used, and how they are signaled, no longer apply. To provide the best user experience possible, a provider doing service identification needs to perform a "best-match" operation, such that

any legal SIP signaling -- not just the specific call flows running within their own network amongst a limited set of endpoints -- is mapped to the appropriate service.

7.3. Presence

Presence can help a great deal with providing unique URIs for different services. When a user wishes to contact another user, and knows only the AOR for the target (which is usually the case), the user can fetch the presence document for the target. That document, in turn, can contain numerous service URIs for contacting the target with different services. Those URIs can then be used in the Request-URI for differentiation. When possible, this is the best solution to the problem.

7.4. Intra-Domain

Service identifiers themselves are not bad; derived service identification allows each domain to cache the results of the service identification process for usage by another network element within the same domain. However, service identifiers are fundamentally useful within a particular domain, and any such header must be stripped at a network boundary. Consequently, the process of service identification and their associated service identifiers is always an intra-domain operation.

7.5. Device Dispatch

Device dispatch should be done following the principles of [RFC3841], using implicit preferences based on the signaling. For example, [RFC5688] defines a new UA capability that can be used to dispatch requests based on different types of application media streams.

However, it is a mistake to try and use a service identifier as a UA capability. Consider a service called "multimedia telephony", which adds video to the existing PSTN experience. A user has two devices, one of which is used for multimedia telephony and the other strictly for a voice-assisted game. It is tempting to have the telephony device include a UA capability [RFC3840] called "multimedia telephony" in its registration. A calling multimedia telephony device can then include the Accept-Contact header field [RFC3841] containing this feature tag. The proxy serving the called party, applying the basic algorithms of [RFC3841], will correctly route the call to the terminating device.

However, if the calling party is not within the same domain, and the calling domain does not know about or use this feature tag, there will be no Accept-Contact header field, even if the calling party was

using a service that is a good match for "multimedia telephony". In such a case, the call may be delivered to both devices, but it will yield a poorer user experience. That's because device dispatch was done using declarative service identification.

The best way to avoid this problem is to use feature tags that can be matched to well-defined signaling features -- media types, required SIP extensions, and so on. In particular, the golden rule is that the granularity of feature tags must be equivalent to the granularity of individual features that can be signaled in SIP.

8. Security Considerations

Oftentimes, the service associated with a request is utilized for purposes such as authorization, accounting, and billing. When service identification is not done properly, the possibility of unauthorized service use and network fraud is introduced. It is for this reason, discussed extensively in Section 6.1, that the usage of declarative service identifiers inserted by a UA is not recommended.

9. Acknowledgements

This document is based on discussions with Paul Kyzivat and Andrew Allen, who contributed significantly to the ideas here. Much of the content in this document is a result of discussions amongst participants in the SIPPING mailing list, including Dean Willis, Tom Taylor, Eric Burger, Dale Worley, Christer Holmberg, and John Elwell, amongst many others. Thanks to Spencer Dawkins, Tolga Asveren, Mahesh Anjanappa, and Claudio Allochio for reviews of this document.

10. Informative References

[RFC3261] 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.

[RFC4479] Rosenberg, J., "A Data Model for Presence", RFC 4479, July 2006.

[RFC4485] Rosenberg, J. and H. Schulzrinne, "Guidelines for Authors of Extensions to the Session Initiation Protocol (SIP)", RFC 4485, May 2006.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[RFC5031] Schulzrinne, H., "A Uniform Resource Name (URN) for Emergency and Other Well-Known Services", RFC 5031, January 2008.

[ECRIT-FRAMEWORK] Rosen, B., Schulzrinne, H., Polk, J., and A. Newton, "Framework for Emergency Calling using Internet Multimedia", Work in Progress, July 2009.

[RFC5627] Rosenberg, J., "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627, October 2009.

[RFC5688] Rosenberg, J., "A Session Initiation Protocol (SIP) Media Feature Tag for MIME Application Subtypes", RFC 5688, January 2010.

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[RFC3841] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", RFC 3841, August 2004.

[RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

Author's Address

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
USA

EMail: jdrosen@jdrosen.net
URI: http://www.jdrosen.net