An Introduction to IMS

Symsource Ltd.

Published by the Symbian Developer Network

Version: 1.1 – January 2008

1 INTRODUCTION 2

2 WHAT IS IMS? 2

3 WHAT DOES IMS DO? 2

3.1 User and service addressing 3
3.2 User identification and authentication 3
3.3 Variable quality of service 3
3.4 Differential charging 4
3.5 Roaming 4
3.6 Interworking 4

4 WHY SHOULD A SMARTPHONE DEVELOPER BE INTERESTED? 4

5 WHAT SPECIFICATIONS ARE INCLUDED IN IMS? 5

6 SIP 5

6.1 Identifiers 5
6.2 Requests and responses 6
6.3 Dialogs 6
6.4 A typical call 6
6.5 Registration 8
6.6 Other request methods 9

7 SDP 9

7.1 Protocol outline 9
7.2 Usage in SIP 10

8 WHAT DOES SYMBIAN OS OFFER YOU? 11

9 FURTHER READING 12

10 BIOGRAPHY 12

1 Introduction

This is the first of a pair of white papers, intended as an introduction to IMS from a smartphone developer's perspective. They will not contain a deep technical explanation of the network side of IMS, instead focusing on the interaction between end-user devices and the rest of IMS.

This white paper gives an overview of IMS and two of its most important protocols, SIP and SDP. The second part will cover some of the practicalities of development on Symbian OS smartphones using SIP and SDP.

2 What is IMS?

IMS is a connectivity architecture intended to enable the streaming of multimedia content to and from end-users over mobile and fixed line networks. IMS stands for IP Multimedia Subsystem, and it's the "IP" part of the name that is most significant: the idea behind IMS is that it will provide a one-size-fits-all framework for the everything-over-IP/IP-over-everything future.

Its origins lie in the 3rd Generation Partnership Project (3GPP), a standards body set up to develop and maintain technical specifications for 3G and beyond, including what eventually took the form of what we now know as UMTS. IMS formed part of the fifth release of the 3GPP specifications, issued in the first quarter of 2002, and updated versions were included in the sixth and seventh releases.

Since then IMS has also been adopted by 3GPP2 (an entirely different body to 3GPP, concerned with the 'other' 3G, CDMA2000) and TISPAN (the European standardisation body for fixed network and internet convergence), giving it a good claim to universality.

3 What does IMS do?

One of the difficulties in explaining IMS is that it is primarily an enabling technology. It doesn't directly implement complete, useful end-user applications, instead offering a standard layer for multimedia applications to run on (see Figure 1). These multimedia applications might include:

- VoIP or voice over IP. Several VoIP services are already available on WiFi enabled devices (e.g. Truphone and Vyke), but when IMS is rolled-out an operator's 'ordinary' voice calls will be sent over IP too.
- Presence. This is the catch-all term for user-visible status information, like that shown in instant messaging buddy lists.
- Instant messaging.
- Push-To-Talk (PTT). Essentially, using a phone as an advanced walkie-talkie.
- Streaming video/internet television.

Of course, examples of these applications already exist in the marketplace, but part of the aim of IMS is to provide a better, unified platform for them, and to make it easier to develop new mobile internet applications. To this end, IMS addresses some common problems encountered when trying to develop applications that communicate between mobile devices over IP, as detailed in the following sections.

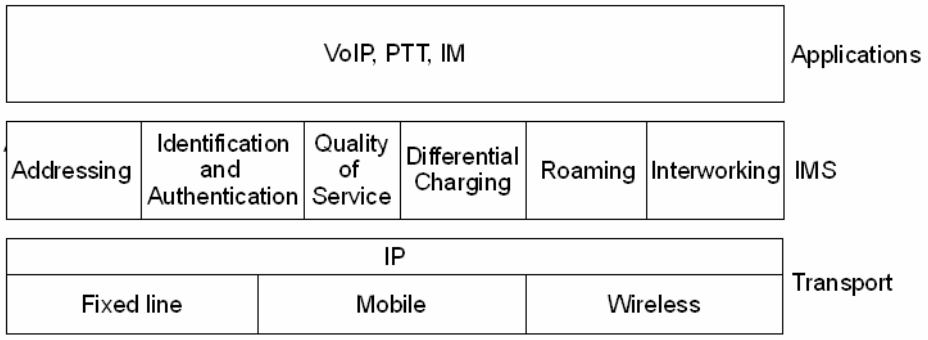

Figure 1 IMS provides a platform for multimedia applications

3.1 User and service addressing

The traditional way of addressing something on the internet is to use an IP address. This works well if the service or user you are after is tied to a particular location, but less so if it is not. For example, consider using VoIP on mobile phones: to transmit the call to the user you need their IP address, but the IP address of a device is dynamically allocated and liable to change as the device moves between cells or WiFi hotspots.

As a further example, imagine a cross-platform IM client and a user that wants to use IM from their home and work PCs, as well as from a mobile phone. Each of these devices will have a different IP address, and the end user is likely to switch between them regularly. How does a third party know which address to use in order to contact our user?

A simple solution to this problem is to use some sort of registration system, whereby users periodically contact a central registry with their location and IP address. Then, when one user wishes to contact another, the system uses the registry to look up the location or address of each party. Currently, this is usually implemented on a per-service basis, or is dependent on the deployment of Mobile IP proxies throughout the internet. IMS offers the promise of a single registry per network operator, or rather a group of registries that can be accessed in a standardised way with a single point of contact, and a standard way for the registries to communicate with each other, both intra- and inter-network.

3.2 User identification and authentication

Not only must we be able to find our users, but we also need a way of identifying and authenticating them, both to control access to our service and possibly to charge them for using the service. Again, IMS aims to provide a standardised way of doing this. As a result of the tight integration of IMS with operator networks, the system enables public identifiers to be tied to subscriber identities using the SIM card embedded in the user's mobile phone, thus making user impersonation much harder than it would otherwise be.

3.3 Variable quality of service

When streaming multimedia content, a guaranteed minimum level of service can be important. For example, delivering real time broadcast quality television over IP is currently a difficult proposition, thanks to the usual 'best effort delivery' nature of the internet, in which everything gets an equal share of resources. IMS promises to change this by making it possible for the network to

differentiate between different types of sessions at the session initiation stage, and then reserving network resources for the session at the decided-upon routing points.

3.4 Differential charging

With the inevitable migration from circuit-switching to packet-switching based services, network operators can continue to differentiate their value-added services by deploying IMS infrastructure. IMS allows them control over the content streamed over sessions, and the ability to differentially price this content, by examining data packets as they traverse the network and requiring that multimedia sessions be set up in a network-controllable fashion.

3.5 Roaming

Given the geographically restricted ranges of operators, the ability to transparently access services on your home network while roaming is important. IMS provides a full specification for handling inter-network communication and routing of data.

3.6 Interworking

Finally, IMS can support seamless interworking between WLAN and 3G systems. This means, for example, that an end-user would be able to make a call via their own wireless broadband connection from their phone as though they were connected directly to the cellular network.

4 Why should a smartphone developer be interested?

When the standard was first published, IMS was widely heralded as the future of mobile telephony, with most operators stating their intention to adopt the specification. Inevitably, such a large and complex change in the network infrastructure has required a long period of development and testing and, as of late 2007, several operators are running trials, but none has rolled out a full public system. The reasons for this are twofold.

Firstly, IMS is an enormous specification, meaning that producing a full, network-side implementation is a difficult task. Some of its fine detail is still in flux, partially as a result of problems arising from unforeseen complexity in implementation.

Secondly, existing 3G infrastructure has been sufficiently technically robust to satisfy the needs of today's mobile multimedia services, which have been taken up more slowly by consumers than perhaps was expected. This means that the need for functionality such as differential content charging over IP is less pressing than it once appeared.

However, it is still worth paying attention to IMS now. The basic proposition of IMS remains very compelling from an operator's point of view and is therefore a likely candidate for at least partial implementation. Therefore a basic understanding of IMS will probably be needed to write software that accesses network services in the not-too-distant future.

Even if IMS doesn't become as all-pervasive as the 3GPP would like it to be, it is very likely that its key technologies will remain important and useful. For example, SIP offers a standard way of addressing users in a context-independent way, and currently underpins many existing standards-based VoIP implementations.

5 What specifications are included in IMS?

The full IMS specification is enormous, running to several thousand pages, and encompassing a huge range of different protocols and interfaces. A typical explanatory diagram of the architecture includes a large number of opaque acronyms and a complex set of interconnections (see the wikimedia commons diagram[1] for an example of this).

However, from a smartphone developer's perspective, most parts of the IMS specification have no direct bearing, dealing with the precise details of inter- and intra-network communication. In the following sections we will describe the basics of the protocols that are arguably most important to the smartphone developer: SIP and SDP. Together these provide IMS with the information it needs to do the sophisticated routing, charging and service quality manipulation described above. SIP and SDP are also useful protocols in their own right, giving an application the power to negotiate its own, non-IMS, sessions across ordinary networks.

6 SIP

The Session Initiation Protocol (SIP) is the most important part of IMS, providing the glue that makes the entire system possible. As the name suggests, it is primarily a protocol for initiating and controlling multimedia sessions. It has several extra features, some of which are discussed below, but, critically, under normal operation, the one thing it does not do is carry the content of the session. Instead, this content is streamed over protocols, and to destinations, that are negotiated using SIP.

As protocol formats go, SIP is fairly straightforward, being text based, human readable, and following standard internet conventions. It's also a peer-to-peer protocol, in that a typical user agent will switch between client and server modes as the situation demands. For example, when SIP is being used as the initiation protocol for voice calls, the same agent on a phone will be used for both incoming and outgoing calls: incoming corresponding to SIP user agent server (UAS) mode, and outgoing, SIP user agent client (UAC) mode.

SIP uses both TCP and UDP as its transport layers, the choice between the two being made on a stage-by-stage basis as a SIP message is passed between user agents and proxies. SIP includes its own reliability mechanisms, which makes the lighter-weight UDP the usual choice, but for technical networking reasons (including message size limits, and firewall and NAT traversal) TCP is sometimes preferred.

6.1 Identifiers

For sessions to be initiated between participants, there needs to be some way of identifying the participants. SIP chooses to identify all potential participants with a public URI that conforms to the generic URI rules (specified in RFC 2396). This serves a similar purpose to an email address and follows a similar form. Typical SIP URIs would look something like this:

```
sip:alice@alpha.com
sips:bob@beta.com
```

The part before the colon specifies the protocol (usually "sip" or "sips", the latter guaranteeing a secure session, but "tel" is also supported on IMS for interworking with circuit switched networks). For further details see section 19 of RFC 3261.

[1] http://commons.wikimedia.org/wiki/Image:Ims_overview.png

6.2 Requests and responses

Much like HTTP, the basic units of interaction in SIP are the client request and the server response, with a single transaction being composed of a request and one or more responses.

A request from a client consists of a single line containing the method name, SIP URI and version, followed by a collection of header fields on their own lines, and a body whose content type and length were described in the headers. For example, a basic INVITE request might look like this:

```
INVITE sip:bob@beta.com SIP/2.0
Via: SIP/2.0/UDP atreides.alpha.com;branch=z9hG4bK78a94
Max-Forwards: 70
To: Bob <sip:bob@beta.com>
From: Alice <sip:alice@alpha.com>
Call-ID: 014098ad897b0809e@atreides.alpha.com
CSeq: 1123 INVITE
Contact: <sip:alice@atreides.alpha.com>
Content-Type: application/text
Content-Length: 23
Some example body text
```

A server response consists of a single line detailing the SIP version and the response code, followed by headers and a body, as in the request. The response codes follow roughly the same pattern as those for HTTP: 2xx for acceptance, 4xx for error, etc.

```
SIP/2.0 404 Not found
Via: SIP/2.0/UDP atreides.alpha.com;branch=z9hG4bK78a94
To: Bob <sip:bob@beta.com>
From: Alice <sip:alice@alpha.com>
Call-ID: 014098ad897b0809e@atreides.alpha.com
Cseq: 1123 INVITE
Content-Type: application/text
Content-Length: 13
Bob not found
```

6.3 Dialogs

At a higher level, a dialog is a relationship that exists between two parties in an ongoing SIP session. A dialog is created when the first non-failure response to a request of a certain type (such as a call invitation) is generated, the assumption being that a non-failure response indicates that there will be further interaction in the immediate future. The requests and responses between the two parties that then follow as part of the session are said to be in-dialog. This distinction between in-dialog and out-of-dialog communication is important: the protocol makes several assumptions about the state held by participants when handling in-dialog communication that it does not make for out-of-dialog communication.

For example, the SIP transactions in the typical call outlined in the next section collectively form a dialog. Once the ringing response has been generated, Bob and Alice's user agents must maintain state concerning each other's location, the routing and sequencing of the requests and responses, whether communication should be done over secure channels, and the identity of the dialog itself.

6.4 A typical call

The most common SIP activity is that of creating and then terminating a media session, usually referred to as a call. This is so fundamental to understanding the way that SIP works in practice that it's worth going through an illustrative example of the transactions that make up a typical call dialog.

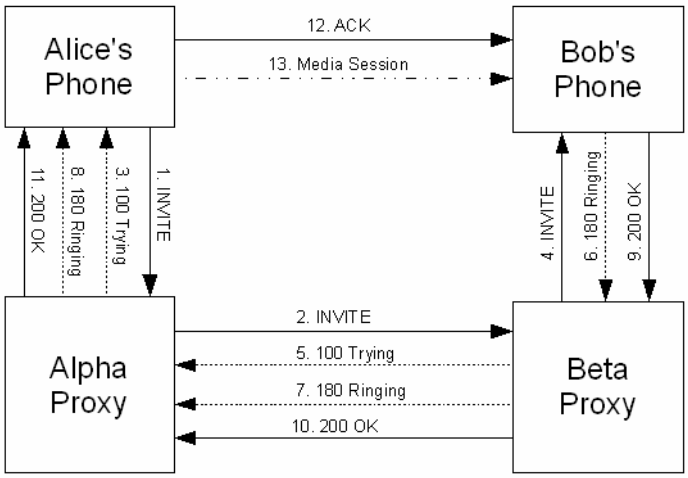

Figure 2: Typical SIP call

So, imagine that we have two subscribers to a VoIP service, called Alice and Bob. Alice is on network Alpha, Bob on network Beta, and Alice wishes to make a call to Bob. How does the call proceed?

To initiate the call, Alice first sends out an INVITE request to `sip:Bob@beta.com`. This will look very like the one in the example request above but the example text body is replaced by an SDP capabilities packet describing the type of call required and how to start it.

Now, Alice's phone is set up to use the central `alpha.com` server as a proxy, so the INVITE request gets sent there first (stage 1 in Figure 2). On receiving the request, the `alpha.com` proxy server recognises that it isn't responsible for `beta.com`, so it does a DNS look up for the `beta.com` SIP server and then forwards the request to the resulting IP address (2). Once it has done this it returns a `100 Trying` response to Alice's phone to indicate reception of the request, and that it's trying to contact the final destination (3).

When the `beta.com` proxy receives the request, it repeats the process, forwarding the request to Bob's phone (4) and sending a `100 Trying` response back to the `alpha.com` proxy (5).

If Bob's phone supports the capabilities specified in Alice's request, it starts ringing. It then sends back a `180 Ringing` response to the `beta.com` proxy (6), which sends it back to the `alpha.com` proxy (7), which in turn sends it back to Alice's phone (8).

When Bob picks up the call, a `200 OK` response gets propagated to Alice (9,10,11). This `200 OK` response contains Bob's phone's actual IP address, which means that Alice's phone can then send a ACK request directly to Bob's phone to confirm that she has received the `200 OK` and accepts the session (12), and then start the actual voice call over the protocol of choice (usually RTP/RTCP) (13).

If Bob's phone doesn't receive this ACK within a certain period of time, it will re-transmit the `200 OK` again at intervals, until it receives the ACK. This ensures a higher degree of reliability: once the INVITE/200/ACK cycle has been completed Alice knows that Bob has received the INVITE, and that he is happy with it, and Bob knows that Alice knows this, which means that it is safe for the call to go ahead. If it weren't for the ACK, Bob could potentially be left waiting forever for Alice to begin the media session.

If either of the parties changes address or media capabilities mid-call, they can re-issue the INVITE with the new details, keeping the call identifying headers the same so that the other party can distinguish it from a fresh invite. The same negotiation process then ensues. Note that either party

can modify a call in this way: there is no restriction that the same party remain the UAC throughout the session, only for each transaction.

When one of the parties wishes to terminate the call, they send a BYE request to the other party, who then replies with a `200 OK`. Again, either party can issue the request.

Contacting a service, such as a video over IP provider, instead of a person, is done in a similar way. The details of the messages may differ but the structure remains the same.

There are several significant details that have been omitted above for the sake of clarity. These include: distinguishing between session types; how Bob's phone decides to accept the session; how the server responses get propagated backwards; and what happens if Bob fails to pick up. The first two of these are covered in the section on SDP, for the rest see the SIP RFC 3261

6.5 Registration

One omission from the typical call overview is sufficiently important to deal with here: how do the network proxies know where Alice and Bob are? This is an interesting question for the smartphone developer in general, as phone IP addresses are frequently reallocated. It's even more important for IMS development, with its promises of transparent roaming and interworking.

SIP provides a straightforward mechanism for achieving this, known as registration. This requires an additional logical server, known as the registrar, which maps the public SIP URI of the user (often referred to as the Address Of Record or AOR) to their physical address (for simplicity and efficiency, this is commonly co-located on the same physical server as the proxy). To keep the registrar updated with its location, the SIP user agent periodically pings the server with REGISTER requests containing its current address, to which the server responds with an appropriate response (usually a 2xx or 4xx). Each REGISTER request must contain an expiration header, which gives the validity of the location in seconds.

Using the REGISTER request, it is also possible to specify different physical locations for different media directed to the same user. For example, Alice might want all incoming IM to be directed to her laptop, while all incoming voice messages should go to her mobile phone.

IMS uses the basic framework of SIP registration, but adds a shared secret challenge and response authentication mechanism to it. On the user's side, the secret is held on the SIM card equivalent, as part of the IP Multimedia Services Identity Module (ISIM); on the IMS side it's held by the Home Subscriber Server (HSS).

The main practical effect of this is that to register a user two REGISTER requests have to be sent instead of one. The first request is much the same as that on an ordinary SIP system, except for the addition of the private identity information obtained from the ISIM. On reception of this request, the IMS registrar passes this private identity to the HSS, which uses it to look up the secret, and from the secret generate a hash using a random nonce and a specific algorithm. It then returns the hash, nonce and algorithm specification to the registrar, which in turn includes the nonce and algorithm as part of a `401 Unauthorized` response to the user agent.

The UE then passes the nonce and algorithm specification to the ISIM so that it can generate an equivalent hash. This hash is then included in the second REGISTER request to the registrar. If the ISIM generated hash matches the HSS generated one, the registrar finally registers the user and returns a `200 OK` response to signal that registration is complete.

To ease the burden this additional complexity causes when dealing with the registration of multiple AORs to a single address, IMS also offers a mechanism known as implicit registration. This allows several AORs to be grouped together by the registrar, so that when one is registered the others are automatically registered at the same time. By extension, normal, non-implicit registration on an IMS network is often called explicit registration.

6.6 Other request methods

In addition to the INVITE/ACK/BYE and REGISTER methods outlined above, SIP also supports the following methods

- **CANCEL**: used to cancel INVITEs that have not completed.
- **OPTIONS**: used to query another user agent or proxy server as to its capabilities, without initiating an actual call.

The following extension methods are also part of the current SIP standards and in some cases are mandatory when using SIP over IMS.

- **MESSAGE**: used to send a short message to another SIP user agent, outside a call dialog. This is often used to supply basic instant message functionality over SIP, and forms part of the SIMPLE open messaging specification. However, it is not limited to IM, and can be used for sending any short packets of data.
- **SUBSCRIBE**, **PUBLISH** and **NOTIFY**: these collectively make up a publish and subscribe system. A user agent can SUBSCRIBE to a particular event; it will then receive a NOTIFY request whenever that event state changes. Conversely, PUBLISH requests will cause an event state to change.
- **UPDATE**: this allows a user agent to modify call settings after an initial INVITE has been sent, but before the transaction has completed. It does the same job as a re-INVITE at a time when a re-invitation would not make sense.
- **PRACK**: acknowledgement request for in-dialog 1xx provisional responses. Introduced to ensure the reliability of 1xx responses, which is required for IMS's compatibility with some fixed-line telephony protocols. Its use is controlled by the user agent server, which can specify that the client uses it by including a 'Require: 100rel' header in its 1xx responses.
- **REFER**: indicates to the recipient that it should contact a third party. Used for call transfer, amongst other things.
- **INFO**: used to update application specific information once a call has been established.

7 SDP

The Session Description Protocol gives a standardised way to define the media capabilities of a proposed session, and as such usually forms the payload of SIP INVITE, OPTIONS, UPDATE and ACK requests and responses.

7.1 Protocol outline

SDP is a simple, compact, text-based protocol that uses key=value fields separated by CRLF pairs. Unusually, the ordering of the fields is fixed and the keys restricted to a limited set of single, lower-case characters; this is to make parsing simpler and increase the likelihood that transmission errors lead to easily rejectable malformed packets. A typical packet might look something like this:

```
v=0
o=alice 2890844526 2890842807 IN IP4 192.1.0.1
s=phone call with Bob
e=alice@alpha.com (Alice)
c=IN IP4 192.1.0.1/127
t=2873397496 2873404696
m=audio 49170 RTP/AVP 0
```

Three different types of information are carried in SDP packets: the session description, one or more time descriptions and zero or more media descriptions.

The first part of a packet consists of the session description. This must contain the SDP protocol version number (the 'v' field in the example) and the originator, identifier ('o') and name of the session ('s'). It can also optionally include several other fields, including the email address ('e') and phone number of the originator, bandwidth requirements and encryption keys. In addition, it should also contain some connection information (i.e. an IP address), unless all the media descriptions contain their own connection data.

The timing descriptions appear midway through the session description. These consist of a field giving the start and end of the session's availability in seconds since 1900 ('t'), and an optional field giving repeat information.

The final part of a packet is made up of zero or more media descriptions, one for each type of media supported by the session. Each of these must include a field containing the type of media, the port the media uses, the protocol used, and the format of the particular protocol ('m'). The media field in the example specifies the RTP/AVP protocol for audio, running on port 49170. They can also contain titles, bandwidth information and encryption keys, and must contain connection information if no session wide connection information is provided.

7.2 Usage in SIP

As outlined above, a typical user agent client to server INVITE request will contain SDP information in its body. This is passed through the proxy servers to its final destination, potentially undergoing some modification as it propagates. Once the request has reached the user agent server, it examines the SDP information, and decides what action to take.

If the UAS wants to accept the call, and has the requested media capabilities, it sends back the SDP information as part of its `180 Ringing` response, having edited the port numbers and connection information to map on to its own ports and IP address. The call then proceeds as detailed in the example above.

If the UAS can theoretically accept the call (i.e. it is not engaged or otherwise out of action), but doesn't support one or more of the media types in question, it can instead attempt to negotiate the capabilities.

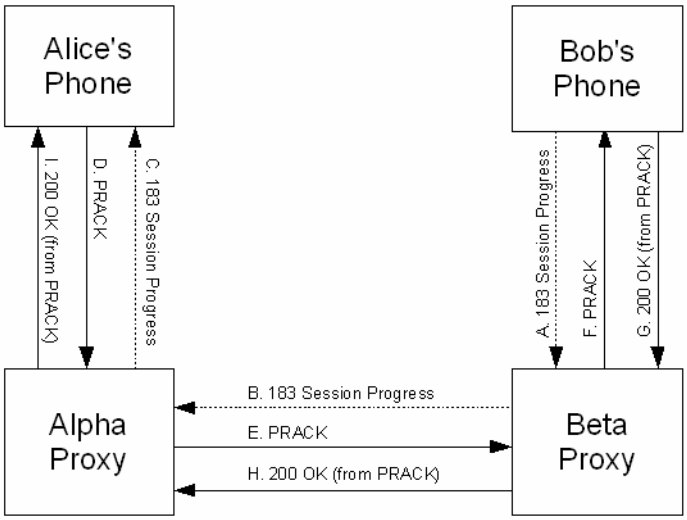

Figure 3 Extra SIP requests and responses when negotiating capabilities

It does this by sending a `183 Session Progress` response (A, B and C in Figure 3) containing the same SDP information as the `180 Ringing` response, only with the port numbers of the unsupported media types set to zero. It may seem simpler to drop the relevant media descriptions altogether; the SDP protocol forbids this by insisting that the same number of media descriptions be included in requests and responses. Again, this is to make parsing and detection of transmission errors easier.

On receiving the `183 Session Progress` the UAC can then re-offer a new set of capabilities in the next packet. If the session is acknowledging partial responses, this new offer can be made in the body of the PRACK (D, E and F), the idea being to piggyback on all available requests to avoid wasting bandwidth by sending dedicated requests unnecessarily. Otherwise, the UAC can use an UPDATE request.

If the UAS accepts these new capabilities, it makes the usual `200 OK` response (G, H and I) to the PRACK or UPDATE and then sends back the `180 Ringing` response. If not, the UAC can make yet another offer, and the cycle repeats itself.

If the call is being made on an IMS system, the proxies can use the contents of the transmitted SDP information to determine charging rates for the session, reserve bandwidth, or even disallow the session entirely.

8 What does Symbian OS offer you?

From version 9.2 onwards, Symbian OS comes with a full SIP stack, plus an SDP handling API.

The API to the SIP stack is comprehensive, containing the following features:

- Full support for initiating outgoing calls.
- An ECOM plug-in architecture for selectively diverting incoming, out-of-dialog and dialog-initiating requests to different applications.
- Dialog state maintenance.
- Full header manipulation tools, with specialised classes for each header type.
- HTTP Digest security handling.
- Management of multiple SIP profiles.

The SDP API is more limited, but still useful, supplying a set of classes for generating and parsing SDP information. Each SDP field is represented by a single class, which both parses and generates SDP formatted input and output, and allows manipulation of each of the components of a field. In addition, there is a container document class to hold all of these fields in the correct order.

Together, the SIP and SDP APIs comprise over 90 classes. Despite this apparent complexity, and the apparent simplicity of the basic SIP protocols, it is almost always worth using the Symbian stack rather than rolling your own.

The SIP protocols are more complex than they initially appear (remember that the example call outlined above is as simple as they get) and the Symbian stack abstracts away a fair amount of the hard work involved in dealing with this complexity.

Further, using the Symbian stack allows your application to run alongside any other applications using SIP on the device without conflict, via its plug-in client resolver architecture. This part of the stack handles all incoming out-of-dialog and dialog-initiating SIP requests, using the headers and SDP body of the request to determine which application it should be forwarded to.

While running multiple SIP applications on one device may not seem like a major issue now, as SIP-controlled communication increases in importance (as it almost certainly will, whether or not

IMS takes off) there will be an increasing number of applications attempting to handle the same incoming packets, often on the same port number.

It's also possible that operators might demand that conforming phones contain only one SIP stack, in which case you will have no choice but to use the Symbian stack when developing applications to be shipped with an operator-supplied phone.

In the second paper, we will go further in to the Symbian SIP and SDP APIs, explaining how to use them in practice.

9 Further reading

The first port of call for SIP and SDP information should be the IETF website <http://tools.ietf.org/html/>. This contains all the relevant RFCs, most of which are clear, unambiguous and easy to understand. The basic SDP (4566) and SIP (3261) RFCs are the places to start, but there have been several SIP extensions that are also worth consulting (3265, 3853, 4310 and 4916).

The 3GPP site <http://www.3gpp.org/specs/releases-contents.htm> includes all the current and historical IMS specifications. Navigating through it can be difficult, though, and the specifications lack the clarity of the IETF equivalents.

There have been two comprehensive books published on IMS: *The IMS* by Poikselka et al (ISBN: 0-470-01906-9) and *The 3G IP Multimedia Subsystem* by Camarillo and Garcia-Martin (ISBN: 0-470-01818-7).

A good high level (if slightly outdated) overview of IMS can be found in the Nokia IMS white paper: <http://www.forum.nokia.com/info/sw.nokia.com/id/fa12d593-31b1-444c-bbc9-02f57cd8d4ea/IMS_white_paper.pdf.html>.

The Tech-invite <http://www.tech-invite.com/> website contains an enormous amount of SIP-related information.

10 Biography

Founded in 2004, Symsource (www.symsource.com) has developed a reputation for reliably delivering complex and innovative solutions in the wireless broadband arena, particularly in innovative SIP-applications. Working in collaboration with our customers, we ensure their business needs are addressed in the most efficient manner, mindful of the total cost of ownership of the solution.

Symsource has built a library of mobile utilities that can be deployed rapidly into our clients' internet-enabled value-added services, meaning that our connected solutions always have a head start. This library encompasses server-side and client-side components built by a team with the foremost expertise in mobile development. For examples of the type of application enabled by Symsource's technology, see www.motxt.com.