

Introduction

Telecommunications networks are being radically transformed in many ways. One of the most important transformations is from being closed systems that are owned, operated, and programmed by a few companies, to being open systems that allow, in principle, anyone to develop new programs and offer new services to users.

This transformation represents a major development in the evolution of networks. If fully or even substantially completed, it will be a key to realizing the tremendous potential of telecommunications and the Internet today. This ongoing transformation is the context and motivation for this book.

This book is about *call control*, by which we mean the process of creating, manipulating, and terminating communications sessions. As you can imagine, call control is a central technical concern in communications networks. More specifically, however, this book is about call control *Application Program Interfaces (APIs)*. For the moment, consider an API to be a set of functions offered by a network that allow a programmer to write programs that provide new, useful, innovative, and (hopefully) lucrative services to users.

Why do we care about APIs? The ability to offer new services rapidly and efficiently is crucially important to service providers, such as large telephone companies and Internet Service Providers (ISPs). Since the 1990s, the revenue that service providers have obtained from voice calls, measured in terms of Average Revenue Per Minute (ARPM) of use, has continued to decline steadily. This decline is particularly true for long-distance and wireless service, largely as a result of deregulation and increased competition. The ability to make two-person voice telephone calls, or what is called *Plain Old Telephone Service (POTS)*, is rapidly becoming a commodity, at least in the developed world. The key to continued revenue growth and profitability clearly lies in advanced services for which users are willing to pay a premium and that service providers can use to differentiate themselves from competitors.

As far as the ability to program new services goes, the API defines the capabilities and limitations of the network. If the network functionality is not available to programmers via the API, then generally only the vendors of the individual elements

(switches, network databases, etc.) have access to that functionality and can create new services using those functions.

In this book, we focus on *open, standard* APIs for call control—that is, APIs that are specified, published, readily available, and accepted within the industry. The intent of the open, standard APIs we discuss in this book is that they provide benefits not only to service providers, but to all parties (see Fig. 1–1). Thus, a company that develops application software benefits because it can then enter the market for telecommunications applications, which without an open API was generally restricted to relatively few companies. If the API is also a standard, the application software vendor can sell the same application to a large number of service providers. Similarly, a manufacturer of network equipment (e.g., switches) that implements an open standard API can sell the same product to multiple service providers, rather than developing a customized product for each.

Of course, all three parties (service providers, application vendors, and network equipment manufacturers) also face increased competition as a result of open standard APIs. For example, the application vendor must compete with other similar companies based on cost, speed of delivery, performance, and so on, and the same holds for service providers and network equipment manufacturers. However, the hope is that this increased competition lowers prices and spurs innovation, resulting in consumer benefits and, ultimately, benefitting the industry as a whole.

For this reason, the definition of a clear, precise, functionally rich, and yet easily implementable API, particularly for the central function of call control, is such an important task, and is the subject of this book.

Finally, this book is about APIs for *converged networks*. Converged networks consist of—and can take advantage of—multiple underlying technologies, such as packet networks (like the Internet), circuit-switched networks (like the telephone network), and wireless networks. The promise of converged networks lies not in

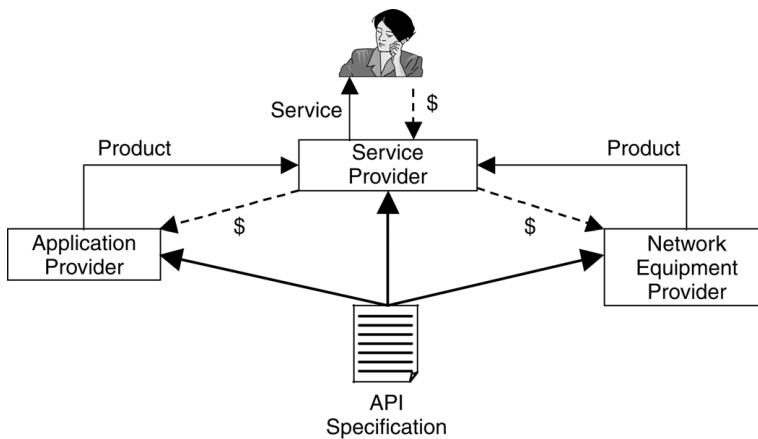


Figure 1–1. A simplified view of the parties using an API.

their ability to interconnect different networking technologies per se, but that with the help of open, standard APIs, they can provide a wide range of connectivity options and innovative and advanced services to end users.

In this chapter, we will provide a brief introduction to call control, APIs, and converged networks, and discuss why they are important and where they are used. In later chapters, we will discuss each topic in more detail.

1.1 SESSIONS AND CALL CONTROL

If call control is about creating, manipulating, and terminating sessions, perhaps we should call it “session control”. But in this instance (for once), we bow to tradition and use the term call control, although we will use both call control and session control interchangeably.

So what is a session anyway? At a basic level, a *session* is an unbroken exchange of information between two or more entities, called *parties*. Typically, parties are human users, but could also be machines. Some examples of sessions are an Internet text chat, a telephone call, and a videoconference.

In general, the exchange of information in a session can be one- or two-way. More precisely, the session may be *simplex* (there is only one source of information during the entire session), *half-duplex* (there is only one source at any given time, but different sources at different times), or *full-duplex* (two sources at the same time). There can be periods of silence during the session, when all parties are quiet (perhaps thinking), but we do not consider the session to be broken, because every party is listening and some party can start talking at any time. Thus, we assume that sessions are explicitly created, exist for some period of time, and are then involuntarily or voluntarily disconnected (or torn down), after which no party can communicate with the others unless it uses or creates another session.

Why do we care about controlling sessions—that is, why do we care about call control? For simple calls, users actually think or care little about call control, and that is how it should be. Ideally, a user should be able to think, for example, “I want to talk to Mom”, and a call should be automatically set up. The call should last as long as both parties want to talk, and then should disconnect automatically. Of course, in real life, users are willing to suffer such inconveniences as dialing phone numbers, entering Internet addresses, or at least saying “Call Mom” into a voice-activated phone.

But someone (an engineer, typically), needs to figure out what happens after the user initiates a call; how to actually set up the call, including ringing the telephone or sending a message to the communications device Mom is using, maintaining the call during the conversation, and then disconnecting it. It turns out that these operations can be quite complicated, even for a simple voice call between two parties.

Further, users want richer and more complex services related to their communications sessions. Users want to place not only two-party voice calls, but also

conference calls with multiple parties, multiple communications sessions of different types at the same time, calls that involve the exchange of data in addition to simple voice communication, and specialized calls that are specifically related to other activities, such as speaking conveniently to a customer service representative while shopping online or paying bills.

But that's not all. Increasingly in this age of information overload, users want services that provide control and flexibility in their communications. For example, they want to

- Use different devices at different times.
- Communicate wirelessly or while mobile.
- Make special efforts to avoid calls (especially from certain people, like telemarketers).
- Make special efforts to reach certain other people, even if they are busy.
- Have machines initiate or receive calls for them.
- Forward calls for their convenience or to save money.
- Prepay for calls or have other people pay for them.
- Know who is calling.
- Avoid calls from people who do not want them to know who is calling.

And the list goes on.

In fact, users even want control over this control. For example, during a video-conference call users may want the ability to decide who the chairperson is and who is paying for the call (or who can disconnect the call unilaterally). Or they may want to pass the chairperson role from one person to another.

The more we rely on technology for communication, the richer the services and the more flexibility and control we want technology to provide. Implementing such services typically involves many different aspects. One such aspect is *user interaction*, which deals with defining the way the user interacts with the system. [For example, does the user have to enter special information, such as a personal identification number (PIN)? Does the system have to play special instructions or tones to the user?] Another important aspect is *charging*—how the system measures and calculates how much money the user has to pay for the service. However, it is often the case, as in the examples above, that the core of the service is defined by the way the communication session is set up, manipulated (e.g., forwarded, transferred, or merged with other sessions), and terminated. In other words, the core of many rich, flexible services that users want lies in call control, making it a central concern when defining new services.

In addition, call control is typically among the more complicated aspects of defining a new service. Sometimes the same service can be implemented in different ways by using different call control features offered by the network, and choosing the right approach involves interesting trade-offs that the service designer must evaluate.

1.2 PROGRAMMABILITY AND APIS

For most of its existence, the Public Switched Telecommunications Network (PSTN) has been a monolithic entity that offered essentially only one service: the ability to make two-party voice calls, that is, POTS.

It turns out that providing the rich, flexible services that users want is hard to do, especially in a system as large and complex as the telephone system. Introducing a new service, even one that is not very different from existing services, often takes a significant investment of money, time, and effort. Implementing a new telephone service can take at least 18 months, and often much longer. The reason is not the complexity of call control, *per se*. The key reason is that the PSTN as a whole is inherently inflexible, and not easily programmable. As we discuss later, this is the case even with innovations like the *Advanced Intelligent Network (AIN)* architecture, of which initial versions were introduced into the PSTN in the 1980s.

The key to introducing new services in a network lies in whether or not it is easily programmable. The first element of programmability lies in the interface that the network provides programmers, that is, the API. The term API has been defined as “a formalized set of software calls and routines that can be referenced by an application program in order to access supporting network services” [TelGloss 2000].

Some readers will recognize that *sockets*, in either their UNIX or Windows flavors, offer an API to networks, allowing logical communication pipes to be set up between machines over a physical medium, and controlling the transfer of messages back and forth [Stevens 1998]. While similar in some sense, the APIs we discuss in this book operate at a higher layer of abstraction and offer richer facilities to the programmer. For example, they allow a programmer to issue commands to the network elements such as:

- “Set up a session between Ali and Beth”.
- “Forward all calls to Beth’s home number to her cell phone”.
- “Make sure that all calls to George get top priority handling”.

In general, the aim of call control APIs is to enable the kind of session manipulations that we described in Section 1.1.

When discussing APIs, it is important to separate the API specification from the API implementation (see Fig. 1–2). The specification describes *what* the network must do, but not *how*. The programmer writes an application that uses or invokes the API. The API specification may, for example, describe the commands the application is allowed to issue, the effect of those commands, and the information that the network provides to the application. It is up to the implementation of the API to convert the specification into software that executes on one or more network elements and actually realizes the functionality of the specification, that is, convert the “*what*” of the specification into “*how*”. For example, the implementation software causes messages to be sent, user data to be transferred, and so on. When we refer to an

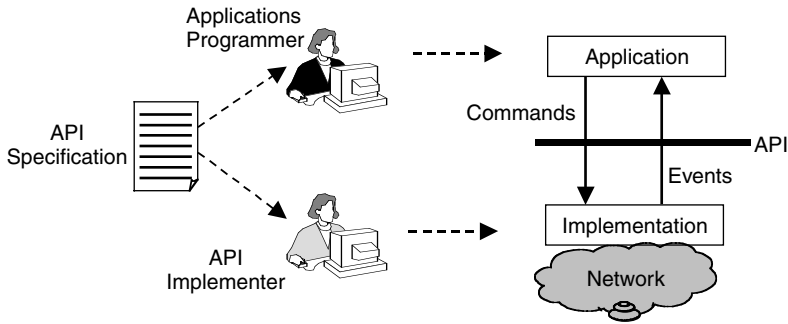


Figure 1–2. API specification and implementation.

API in this book, we mean the API specification. When referring to the API implementation, we say so specifically (unless clear from context).

Because the application logically issues commands to the implementation, and not vice versa, an API is inherently *asymmetric*. Thus, we refer to the application as being “above” the API, and the implementation as being “below” the API. This asymmetry extends to how the API is (or, in our opinion, should be) written. The API specification is written to serve the needs of applications, not the API implementation, although the capabilities of the network obviously define the scope of the API content. The primary user of the API specification is the application programmer, not the implementation programmer.

One thing worth mentioning here is the difference between APIs and protocols. The two are often confused, and some believe that APIs are not necessary, or that protocols can do as good a job of providing network programmability. An API is a “horizontal” separation between different software layers, while a protocol is a “vertical” separation between communicating entities, typically different machines communicating over a network. Unlike APIs, protocols, by their very nature, are intended for interoperability rather than programmability, and cannot really be used directly for programmability. We will discuss this issue in more detail in Chapter 2.

1.3 HOW THIS BOOK IS ORGANIZED

This book assumes that the reader has a basic understanding of fundamental Internet operation, such as IP, HTTP, and related protocols. We also assume some familiarity with object-oriented programming and design. Some knowledge of Java and XML is helpful, particularly in Chapters 5, 6 and 9, which deal with JTAPI, JAIN and Web Services.

The rest of this book proceeds as follows. Chapter 2 provides an historical perspective on programmability in telecommunications networks, beginning with a brief history of the PSTN and ending with the definition of converged networks.

It also provides background material, such as the difference between APIs and protocols. Chapter 2 describes two running examples we use throughout the book to illustrate how the APIs can be used to implement services. Note that several chapters also describe additional examples in detail.

Chapter 3 then discusses the basic features of APIs, particularly call control APIs, and the fundamental choices that the designers of the APIs discussed in the book have made. We use the example of setting up a simple voice call to illustrate this. The basic design choices are helpful to understand and keep in mind as one reads the individual API descriptions, where sometimes they can be obscured by the details of syntax and logic.

The next four chapters (Chapters 4–7) focus on core technical descriptions of what we view as the four main approaches being used for programming telecommunications networks: those embodied in AIN, JTAPI, JAIN, and Parlay. Each chapter begins with a brief historical perspective, followed by a description of the main components of the API. Although the main focus remains on call control functions, brief descriptions of other parts of the overall API, for example, user interaction or charging, are given where appropriate. Each chapter then provides details of how the running example services can be implemented using the API.

In Chapter 8 we take a step back to extend the discussion of Chapter 3 on the design of APIs, but this time with the benefit of having seen the innards of JTAPI, JAIN, and Parlay call control. This chapter briefly discusses some subtle but important issues in API design that ultimately impact the application programmer.

An important emerging area for programming networks is that of integration with the Web, and particularly the use of XML-based languages and methodologies for developing new services. Chapter 9 briefly discusses four efforts in this direction, namely, PINT, SPIRITS, SCML, and Parlay X Web Services. We anticipate that these approaches will become increasingly important in future.

Finally, Chapter 10 makes a critical appraisal of the APIs described in this book, briefly pointing out some of their limitations and how they could be developed further. It then takes a speculative look at the future of APIs: Are they simply an industry fad, or will they remain important in future?

1.4 RELEVANT INDUSTRY FORA

In most of this book, we will discuss the details of APIs that have been standardized. As background, we will briefly describe the standards bodies and industry fora that have been created for these specifications. Rather than list all the standards bodies active in communications, we will focus on those that have bearing on the APIs discussed in this book.

The International Telecommunications Union (ITU) is an agency of the United Nations that sets global standards for radio and phone communication. The Telecommunications Sector (ITU-T) sets standards for telephone communication. Membership is open to governments, corporations, and other groups,

and decision-making is by voting. The Intelligent Network (IN) architecture discussed in Chapter 4 is defined in ITU-T.

The European Telecommunications Standards Institute (ETSI) produces the telecommunications standards for the European Union. ETSI originally defined an extension to the IN for mobile networks, called Customized Application of Mobile Enhanced Logic (CAMEL), as well as a European variation of ITU's IN. This is discussed in Chapter 4.

The Telecommunication Industry Association (TIA) develops U.S. standards for telecommunications. It is officially accredited by the American National Standards Institute (ANSI). The U.S. extension to the Advanced Intelligent Network (AIN) for mobile networks, called *Wireless Intelligent Network (WIN)* is standardized by TIA. This is discussed in Chapter 4.

The 3rd Generation Partnership Project (3GPP) is a formal collaboration agreement between a number of other telecommunications standards bodies, with the aim of producing standards for a 3rd Generation (3G) mobile system based on an evolution of the 2nd generation Global System for Mobile communication (GSM) core networks and the radio access technologies that they support. 3GPP now furthers the definition of protocols such as those in its multimedia domain IP Multimedia Subsystem (IMS). This is discussed in Chapter 4.

The Enterprise Computer Telephony Forum (ECTF) is a standards-setting body for the computer telephony industry. The Java Telephone API (JTAPI) discussed in Chapter 5 was started as an industry-consortium activity, led by four companies. Now ECTF is the lead organization developing JTAPI under the umbrella of Sun Microsystems' Java specifications process.

The JAIN forum is an industry forum that develops API specifications under the terms of Sun Microsystems' Java specification process. The JCC and JCAT APIs discussed in Chapter 6 are defined by JAIN.

The Parlay Group is an industry consortium set up to define open, multivendor APIs for network services. The Parlay call control APIs discussed in Chapter 7 are defined by Parlay, 3GPP, 3GPP2 and ETSI.

The Internet Engineering Task Force (IETF) is a body that defines standards for the Internet. It is "the protocol engineering and development arm of the Internet", [IETF] and is open to any interested individual. There is no official membership, and decision-making is by consensus. The PINT and SPIRITS standards are defined and exist within the context of the IETF. These are discussed in Chapter 9.

The World Wide Web Consortium (W3C) "develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential" [W3C]. In particular, it defines XML and its derivatives (see Chapter 9).

1.5 CONCLUDING REMARKS

A radical change is underway in telecommunications networks: The move from closed systems under the control of a few companies to more open systems that allow new and innovative services and applications to be programmed by others.

The programmability of a telecommunications network is defined by means of its API. This book discusses the specification and use of open, standard APIs for call control, which is defined as the process of creating, manipulating, and terminating communications sessions, and is a central aspect of telecommunication applications.

The definition of open, standard APIs potentially offers benefits to service providers, because they can tap into the creativity and effort of programmers all over the world to develop innovative and lucrative new applications. In turn, they provide an incentive for programmers to write such applications, because the applications can run on any network that supports the API, increasing the potential market. And finally, they expand the market available to a network equipment manufacturer that builds products conforming to the standard. However, open standard APIs also increase competition, hopefully eventually benefiting the consumer.

While APIs define *what* commands an application can issue to the network, they do not define *how* those commands are carried out. In particular, software and protocols “below the API” are required for actually carrying out the commands that are allowed by the API.

Several standard bodies and industry fora have developed the APIs described in this book.

In later chapters, we will describe the details of individual APIs, giving code and examples of how they can be used, as well as principles of API design.

1.6 TO EXPLORE FURTHER

To our knowledge, there are almost no books on APIs for converged networks, and none that describe the design issues involved in these APIs. The rationale for programming telecommunications networks is presented eloquently in [Lazar 1997]. A broad introduction to protocols and APIs underlying converged networks is provided in [Mueller 2002] and [Zuidweg 2002]. The JAIN APIs, in particular, are described in [Jepsen 2001]. The operation and specifications of industry fora such as IETF and W3C is described on their respective Web sites, as well as in numerous articles and books.

