

Chapter 1

Introduction

Then a minstrel and loremaster stood up and named all the names of the Lords of the Mark in their order. . . . And when Théoden was named, Éomer drained the cup. Then Éowen bade those that served to fill the cups, and all there assembled rose and drank to the new king, crying: 'Hail, Éomer, King of the Mark!' "

"The Return of the King"
J. R. R. Tolkien.

A protocol is a set of rules which have to be followed in the course of some activity. Originally, the term was used solely of human activities, especially those of a somewhat formal kind, such as the state funeral for King Théoden described in the quotation at the start of this chapter. The *chef de protocole* for a Head of State sets formal rules for how activities take place according to the niceties of diplomatic practice. But protocols must also be followed in less elevated spheres, such as games of all kinds, the way in which conversations are conducted, and in fact all activities which are governed by custom and convention. If the protocol is not followed, the activity will not be successful.

In this book we shall consider *communication protocols*, and in particular those which regulate communication between computers. The characteristics of protocols mentioned above are equally evident in this case: A set of *formal rules* governs the exchange of information, and the communication activity *fails* if the protocol is not correctly followed.

1.1 What is a Protocol?

In the general sense, communication between computers takes place by the exchange of data – information encoded in some way which depends on the system concerned. We can consider this exchange as taking place in discrete steps, which we shall call *elementary communications*, in each of which a *message* is transferred. Again depending on the system, a message may be a single electronic signal, or a

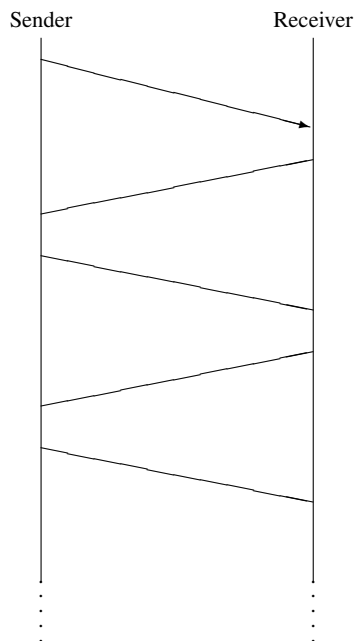


Fig. 1.1 Exchange of messages in a simple protocol.

A common definition of a communication protocol [133] is thus that it is a set of rules for the order in which messages of particular types are exchanged. With our definition of message type, this also implies a set of rules for the encoding of the various types of message.

For each of the N parties to an N -peer communication, the protocol defines a *language*, whose sentences are the legal sequences of messages received by that party, and whose alphabet of symbols is the set of all possible messages. A machine to obey the rules of the protocol must thus essentially be a recogniser for the protocol language. For simple protocols, this is a useful abstraction, as the language is *regular* or at most *context-free*, and standard compiler techniques [124] can be used to implement the machine as a finite state machine or push-down automaton respectively.

A trivial example of a simple protocol described in this way is given below. It is a type of *stop-and-wait protocol*. The sender requests the receiver to indicate when it is ready to receive data, and waits for this indication. On receipt of the indication, the sender sends the data and waits for an acknowledgment. The exchange of messages is as shown in Figure 1.1. The languages to be recognised by the sender and receiver

respectively (and of course to be generated by the receiver and sender respectively) are defined by the BNF:

$$\begin{aligned} \text{sender} &::= \text{readyindication acknowledge sender} \\ \text{receiver} &::= \text{requesttoaccept data} \quad \text{receiver} \end{aligned}$$

Each party must generate and recognise sentences of a regular language. This is a simple task for a finite state machine.

Unfortunately, there are some important objections to this language-oriented view of a protocol. The first is a practical objection: Simple languages generally do not correspond to protocols which can tolerate faults, such as missing or duplicated messages. Protocols which are fault-tolerant often require the use of state machines with enormous numbers of states, or they may define *context-dependent* languages.

A more radical objection is that classical analysis of the protocol language from a formal language point of view traditionally concerns itself with the problems of constructing a suitable recogniser, determining the internal states of the recogniser, and so on. This does not help us to *analyse* or *check* many of the properties which we may require the protocol to have, such as the properties of fault-tolerance mentioned above. To be able to investigate this we need analytical tools which can describe the parallel operation of all the parties which use the protocol to regulate their communication.

1.2 Protocols as Processes

A radically different way of looking at things has therefore gained prominence within recent years. This involves considering the protocol as being defined, not so much by the internal states of the protocol machine, but rather by the *observable external behaviour* of a *process*. The external behaviour is defined as the set of all possible *traces* – sequences of elementary communications in which the process takes part. The work of Hoare, Milner and others ([64], [128], [19], [94]) has shown how the behaviour of combinations of processes can be deduced from the behaviours of the individual component processes through the use of a calculus or algebra, and how it is possible to prove properties of processes starting from certain axioms about the behaviours of their component parts. Chapter 2 gives a short introduction to the method used in this book and the required notation.

This type of approach makes it possible to specify and analyse complex protocols. In particular, the rules for the composition of processes make it possible to analyse protocols which rely on the use of other protocols in some layered manner, as is commonly the case in communication systems. A well-known example of this is seen in the OSI standard architecture for communication systems [133]. Some simple illustrations of the approach and an introduction to the OSI Reference

Model will be given in the Chapter 3, where we also consider the general properties which might be desirable for services in distributed systems.

1.3 Techniques for Actual Protocols

The central chapters of the book are devoted to a presentation of techniques for providing particular types of service by the use of appropriate protocols. This presentation is illustrated by theoretical analysis of some of the protocol techniques, and by a classification of some protocols used in practice today, according to the techniques on which they are based.

The presentation falls into four parts. In Chapter 4 we discuss a number of basic mechanisms for use in 2-peer point-to-point communication protocols, and the relation of these mechanisms to required properties of the service, particularly resilience to simple faults such as corruption or loss of messages.

Chapter 5 considers the problems associated with providing a service to more than two parties, and in particular the problem of getting several parties to agree in the presence of faults. Here we shall extend our repertoire of permitted faults to include arbitrary, possibly malicious faults – the so-called Byzantine errors.

In Chapter 6 we turn our attention to another form of malicious attack to which distributed systems are exposed – attempts by unauthorised persons to read or alter information to which they are not supposed to have access. This is the problem of computer security, whose solution, as we shall see, requires special protocols and a careful use of cryptographic methods.

Finally, in Chapter 7 we consider what techniques are available for locating an intended participant within a distributed system, and for organising transmission of messages so that they reach the recipient reliably and with a minimum of delay. This is the problem of naming, addressing and routing, which is interesting not only because it is a real, practical problem to be solved, but also because the solutions illustrate many of the strategic choices to be taken when decisions have to be made in distributed systems.

1.4 Real Protocols

After considering in a rather abstract manner the techniques available for constructing protocols with particular properties, the final chapters of the book will be devoted to looking at a selection of real protocols, and to analysing how the general techniques are deployed in them.

We start this part of the book by looking, in Chapter 8, at principles used for encoding the messages used in protocols. Then we go on to look at each of the layers of the OSI Reference Model in turn, presenting commonly used protocols, many of them internationally standardised, and classifying them according to the type of

service which they support and the protocol mechanisms used in order to supply this service. Chapter 9 deals with the so-called OSI Lower Layers, which are the layers up to and including the Transport layer. Chapter 10 describes protocols in the OSI Upper Layers – the Session, Presentation and Application layers of the Reference Model – which provide general support for applications. And finally, Chapter 11 presents a number of important protocols associated with specific applications, including file transfer, mail transfer, transaction processing, document access via the World Wide Web and Web services.

This book is not a catalogue of standards, and many protocols of potential interest, particularly in the Application layer, have had to be left out. Even so, the bibliography at the end of the book contains references to more than 130 national and international standards, chosen because they illustrate interesting principles of design. On the other hand, we do not discuss any of the multitude of commercially available protocols from specific suppliers, nor do we enter into detailed presentations of particular protocols. For this kind of specific information, you will need to read the original descriptions of the protocols concerned.

1.5 Reader's Guide

This book deals with both theory and practice, and some readers may prefer to omit one or other of these subjects on the first reading. If you prefer to omit as much theory as possible, you can skip Sections:

- 2.1.2, which deals with process algebra,
- 2.2 and 2.3, which deal with the logic used to prove properties of systems of processes,
- 3.1, which gives an example of a proof that a protocol enjoys a particular property,
- 6.4.3, which deals with the logic used to prove the correctness of authentication protocols.

If on the other hand you prefer to think about the theory and are not much concerned with practice, then you can skip:

- Chapter 8, which deals with encoding of protocols, and
- Chapters 9, 10 and 11, which deal with real protocols used in the various layers of the OSI Reference Model.



<http://www.springer.com/978-3-540-77540-9>

Principles of Protocol Design

Sharp, R.

2008, XII, 402 p. 172 illus., Hardcover

ISBN: 978-3-540-77540-9