Deriving Protocol Converters for Communications Gateways

GREGOR v. BOCHMANN

Abstract—**Gateways are introduced for interworking between several, possibly heterogeneous, distributed computer systems. A gateway has to provide for the necessary adaptation between the communication protocols used in the interconnected networks. This paper explains that the adaptation problem is best handled by considering the communication services of the interconnected systems. Once the problem is solved at this level, the remaining problem of conversion between the incompatible communication protocols used in the different systems can be solved automatically, as demonstrated for the case of a simple example.**

I. Introduction

A COMMUNICATION gateway is a component which allows for the interworking of different systems which use internally different, noncompatible communication protocols. The gateway, connected to both systems, provides the necessary conversion functions [7]. A framework for reasoning about conversion systems and their correctness was presented by Lam [10], [5], which uses an abstraction technique for comparing the incompatible communication protocols. However, their approach does not provide any constructive method for deriving a protocol converter from the specification of the given protocols. The purpose of this note is to indicate that the protocol conversion problem can be handled more easily through the consideration of the related communication services (as already discussed by Gien and Zimmermann [6]), and to show that once the problem is solved at the service level (independently of the protocols), the protocol converter can be derived automatically from the given protocol specifications.

II. Protocol Conversion Architectures

An overview of the issues involved in the interworking between heterogeneous computer systems is given in [7]. Much literature exists about solving particular interconnection problems (see, for instance, [8]). Some underlying principles for gateway designs are also discussed in [4]. In general, the purpose of an interconnection gateway is the provision of a *common communication service* throughout all parts of the interconnected systems. Therefore it seems sensible to start the gateway design process with the consideration of the communication services realized within the two systems to be interconnected [6].

To illustrate the issues, we use in the following the example of a data transmission service from a *sender* to a *receiver* process. As indicated in Fig. 1, the service involves the primitives *Dreq*, *Dind*, *Dresp*, and *Dconf* (in this order) for the transmission of a single data unit from the sender to the receiver. The primitives *Dreq* and *Dind* contain the data as a parameter, and the other primitives have a flow control and acknowledgment function. Two different versions $S1$ and $S2$ of this kind of service are considered: $S1$ is reliable, while $S2$ is unreliable; that is, in the case of $S2$ a *Dconf* primitive may occur after a *Dreq* even if no data were delivered to the receiver.

Paper approved by the Editor for Communication Protocols of the IEEE Communications Society. Manuscript received August 15, 1988; revised May 15, 1989 and November 15, 1989.

The author is with the Département d'Informatique et de Recherche Opérationnelle (I.R.O.), Universite de Montreal, Montreal, P.Q. H3C 3J7, Canada.

IEEE Log Number 9037842.

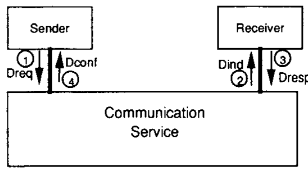

Fig. 1. A simple communication service.

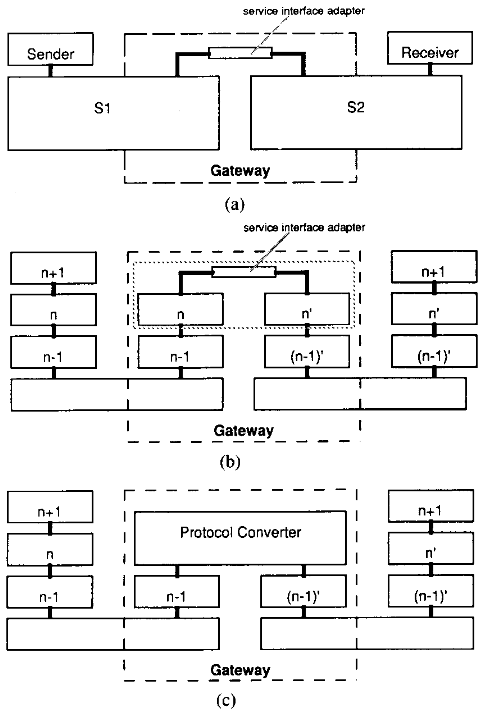

Fig. 2. (a) Architecture of a gateway using conversion at the service level. (b) Architecture of a gateway using conversion at the service level (for interworking at layer n). (c) Architecture of a gateway using conversion at the PDU level (for interworking at layer n).

We assume in the following that two distributed systems using the communication services $S1$ and $S2$, respectively, are to be interconnected, as shown in Fig. 2(a). We call this interconnection architecture *conversion at the service level* [4] (called "service interface conversion" in [12]). The figure shows a so-called *service interface adapter* which maps the service primitives relating to one of the services to the primitives relating to the other service.

In many cases, the primitives of the two respective services can be identified in a one-to-one correspondence. In this case, for each service primitive received, the gateway simply has to issue the corresponding primitive to the service at the other side. We call this *direct concatenation* of the two communication services [4]. For the example of the services $S1$ and $S2$ mentioned above, such a correspondence exists between *Dind* and *Dreq*, and *Dresp*, and

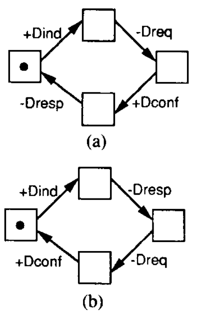

Fig. 3. (a) Specification of a service interface adapter (direct service concatenation). (b) Specification of a service interface adapter.

Dconf. A service interface adapter realizing direct concatenation would therefore operate like the finite state machine (FSM) shown in Fig. 3(a).

This service interface adapter provides end-to-end significance for acknowledgments since an acknowledgment is given to the sender (in the form of the *Dresp* primitive) only after the receipt of an acknowledgment from the receiver. This is not true in the case that the adapter of Fig. 3(b) is used. In this case, an acknowledgment is returned by the adapter towards the sender even before the data is forwarded towards the receiver (through the *Dreq* primitive). Clearly, both of these adapters provide a nonreliable service, since data may be lost during its transfer through $S2$. It is interesting to note that these considerations can be made independently of the communication protocols used within the respective networks.

Fig. 2(b) shows the general case of two layered protocol architectures interconnected through a gateway using conversion at the service level. It is assumed that the communication service interfaces at layer n are adapted between the two systems, and that the communication protocols in the layers above are compatible. If this architecture is used for the implementation of a communication gateway, existing implementations of the protocols to be interconnected can be combined within the gateway system, as shown in the figure. This may facilitate the gateway construction process. An interesting example of the necessary service adaptation is the identification of a common service subset for the interconnection of the OSI and DARPA TCP/IP protocol hierarchies at the transport level [9], [1].

It is important to note that gateways may take advantage of doing the conversion partly at a lower level. In this case, a so-called *protocol converter* considers the protocol messages (PDU's) exchanged within the interconnected systems, as shown in Fig. 2(c). We call this approach *conversion at the PDU level* [4]. This approach is assumed in [7], [10], [13], and [12], the latter considering conversion for several protocol layers together. Examples of various conversion alternatives for Transport gateways, taking advantage of conversion at the PDU level, are discussed in [2]. Although conversion at the protocol level may lead to more efficient gateways, this approach is generally more complex to be specified, and more difficult to be implemented, compared to conversion at the service level.

III. Derivation of Protocol Converters

A comparison of Fig. 2(b) and (c) shows that the subsystem composed of two protocol entities (defined by the respective protocol specifications) and the service interface adapter, as shown by the dotted box in Fig. 2(b), is a valid protocol converter. (Note: The protocol converter may have additional properties, e.g., concerning various optimization strategies [2], [12]). This observation leads to the automatic derivation of a specification for the protocol converter from the given protocol specifications to be interconnected and the definition of the service interface adapter, which defines how the communication services of the two interconnected systems are related to one another. The specification of the protocol converter is

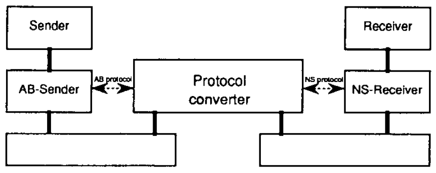

Fig. 4. Architecture showing gateway between the alternating bit (AB) and nonsequenced (NS) protocols.

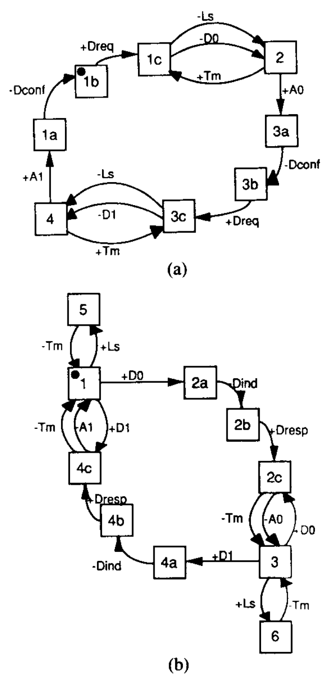

Fig. 5. Definition of an alternating bit (AB) protocol. (a) The AB-Sender entity. (b) The AB-Receiver entity.

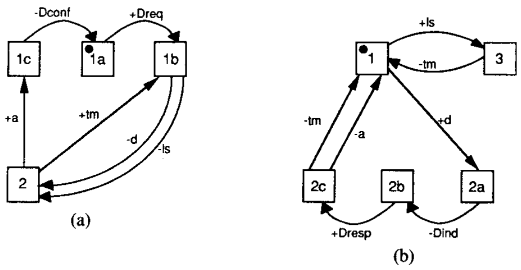

Fig. 6. Definition of a nonsequenced (NS) protocol. (a) The NS-Sender entity. (b) The NS-Receiver entity.

obtained by composing the specifications of the three components shown in the dotted box of Fig. 2(b).

Most specification languages include a constructive method for obtaining the specification of a composition from the specification of its components. In the case of finite state machine models with direct coupling (rendezvous interactions) between different machines, the composition of several machines is equivalent to their coupled product, as described in the first half of [11]. As discussed in [3], direct coupling is an appropriate model for an interface to access a communication service.

We demonstrate the derivation of the protocol converter by considering the interconnection of the services $S1$ and $S2$ intro-

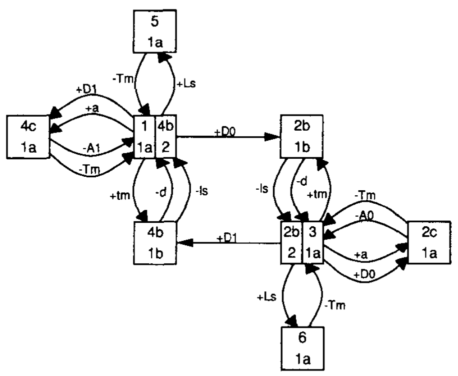

Fig. 7. Specification of the protocol converter in the case of direct service concatenation. *Note:* Each state of the converter contains as comments the corresponding states of the AB-Receiver and NS-Sender, respectively.

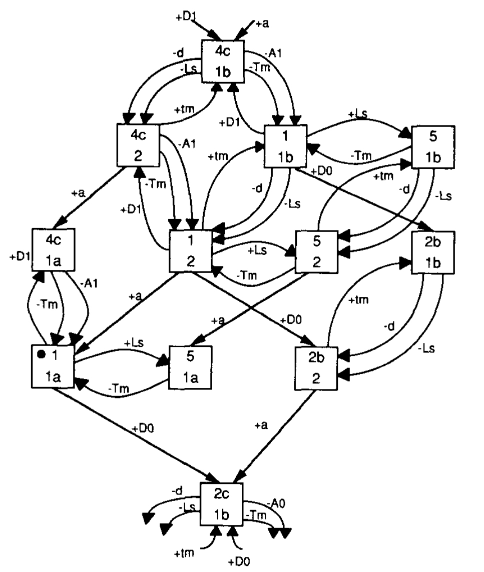

Fig. 8. Specification of the protocol converter in the case that the service adapter of Fig. 3(b) is used. *Note:* The behavior following the state $\langle 2c, 1b\rangle$ is the same as following state $\langle 4c, 1b\rangle$, except that the alternating bit is changed.

duced above. We assume that an alternating bit protocol and a protocol without sequence numbers are used within the two systems, respectively, as shown in Fig. 4. The definition of these protocols is given in Figs. 5 and 6. They are adapted from [10] by adding, in a straightforward manner, the service primitives introduced above. The transitions labeled with a minus sign denote the sending of a message, and the transitions labeled with a plus sign denote the corresponding receptions. Message loss and timeout events are modelled with "virtual messages" (*tm* and *ls* denote "timeout" and "loss," respectively). The possible loss of a data message is modeled by specifying a pair of transitions $-d$ (data) and $-ls$ in parallel, while the loss of an acknowledgment is modeled by a pair of transitions $-a$ (ack) and $-tm$. The event $+tm$ denotes a timeout occurence; note that premature timeout occurences are not allowed by this model, i.e., timeout occurs only if either a data message or an ack message has been lost.

For the case that the interface adapter of Fig. 3(a) is used the specification of the protocol converter is the composition of the AB-Receiver, the adapter, and the NS-Sender. After some simplifications related to the nonvisibility of the service primitives exchanged between these components, the specification of Fig. 7 is obtained. We note that the same adapter was already given in [10], however, no indication was given how it could be obtained in a systematic manner.

It can be expected that different protocol converters are obtained for the same protocols to be interconnected, if different service interface adapters are considered. In fact, the adapter of Fig. 3(b) yields the converter shown in Fig. 8. In this situation, the alternating bit protocol may already transmit a second data packet while the nonsequenced protocol still transmits the first packet. This kind of parallelism is visible in the specification of the figure.

References

[1] G. v. Bochmann, A. Jacques, and C. Kawa, "Transport relays," *Tech. Rep.*, Dep. d'IRO, Universite de Montreal, 1986.
[2] G. v. Bochmann and A. Jacques, "Gateways for the OSI Transport service," in *Proc. IEEE INFOCOM'87 Conf.*, San Francisco, CA, 1987.
[3] G. v. Bochmann and A. Finkel, "Impact of queued interaction of protocol specification and verification," in *Proc. Int. Symp. Interop. Inf. Syst. (ISIIS)*, Nov. 1988, Tokyo, Japan, pp. 371-382.
[4] G. v. Bochmann and P. Mondain-Monval, "Design principles for communication gateways," *IEEE J. Select. Areas Commun.*, to be published.
[5] K. L. Calvert and S. S. Lam, "An exercise in deriving a protocol conversion," in *Proc. ACM SIGCOMM Symp. Commun. Arch. Protocols*, 1987.
[6] M. Gien and H. Zimmermann, "Design principles for network interconnection," in *Proc. VI Data Commun. Symp. (IEEE/ACM)*, Nov. 1979, pp. 109-119.
[7] P. E. Green, "Protocol conversion," *IEEE Trans. Commun.*, vol. COM-34, pp. 257-268, Mar. 1986.
[8] P. E. Green, *Network Interconnection and Protocol Conversion.* New York: IEEE Press Reprint Book, Apr. 1987.
[9] I. Groenback, "Conversion between TCP and ISO transport protocol," *IEEE J. Select. Areas Commun.*, vol. SAC-4, pp. 288-297, Mar. 1986.
[10] S. Lam, "Protocol conversion," *IEEE Trans. Software Eng.*, vol. 13, pp. 353-362, Mar. 1988.
[11] P. Merlin and G. v. Bochmann, "On the construction of submodule specifications and communication protocols," *ACM Trans. PLS*, vol. 5, pp. 1-25, Jan. 1983.
[12] Y. Ohara, S. Yoshitake, and T. Kawaoka, "Protocol conversion method for heterogeneous systems interconnection in multi-profile environment," *Proc. IFIP Symp. Protocol Specific., Testing, Verification*: VII, Zurich, May 1987.
[13] K. Okumura, "A formal protocol conversion method," *Proc. ACM Conf. Commun. Protocols Architect.*, 1986.