Design Principles for Communication Gateways

GREGOR v. BOCHMANN, SENIOR MEMBER, IEEE, AND PIERRE MONDAIN-MONVAL

Abstract—**The purpose of this paper is to define principles that apply to the design of communication gateways between heterogeneous computer systems, and to identify strategies to solve incompatibilities in a systematic manner. The importance of communication service common properties is explored in detail. The modification of the service available for interworking due to subset selection and service concatenation is discussed. Various methods of adaptation are described. Two architectural options for the design of communication gateways are explored, namely conversion at the service level or at the PDU level. While the former is conceptually simpler, various optimizations are possible through the latter approach. A method for deriving the specification of a protocol adapter from the two protocol specifications is also given. All these issues are explained with a number of simple examples.**

I. Introduction

THE concept of gateways was first introduced for the interconnection of different communication networks. Gateways, sometimes also called relays are now used for the interconnection of public packet-switching data networks and private data processing networks, often including local area networks. These gateways belong to the Network layer of the OSI reference model (OSI RM). In this context, any difference between the communication services provided by the different networks has a strong impact on the gateway design and functions. Interworking at other layers of the reference model has been considered, for instance, at the transport or application layers. An OSI-gateway for a non-OSI system is a typical example.

An overview of the issues involved in the interworking between heterogeneous networks is given in [6]. Much literature exists about solving interconnection problems in particular cases, and about incompatibilities between different systems and approaches to their resolution (see for instance [7]). The purpose of this paper is to explain principles that apply to the design of gateways, to identify strategies for solving incompatibilities in a systematic manner, and to provide clear definitions of the relevant concepts.

Already Gien and Zimmerman [5] pointed out that *interconnection of networks must be based on a common communication service* provided in both networks. This principle is further investigated in this paper. Some previous work does not explicitly take into account the communication service [9], [11]. We indicate in Section V that a specification of a protocol converter can be obtained automatically if a mapping between the respective services is defined.

In order to clarify the issues, much of the discussion of this paper is based on simple examples showing the principles involved. In some cases, examples from existing computer communication protocols are also discussed.

Section II presents some basic interconnection scenarios. The importance of a common (often subset) communication service is explored. Concatenation of services is defined in Section III. Different concatenation approaches depending on the types of interfaces and services are discussed. Concatenation properties are investigated. The concept of concatenation invariance is discussed with some examples.

While the discussion up to that point is mainly concerned with the service characteristics, Section IV deals with an alternative approach already widely used, protocol conversion, where the gateway needs to know in detail the different protocols to explicitly establish a correspondence between them. Advantages and drawbacks of this protocol level adaptation approach are discussed. A method for deriving the specification of an adaptation module from the interworking protocol specifications is also given.

Finally, Section V contains a list of general conclusions.

II. Interconnection of Communication Services

A. A Basic Example

This basic example is the interconnection of modules, called Basic Medium, providing a simple message transfer service (Fig. 1). Such a module has two ports through which it interacts with its users. The port on the left-hand side of the picture (sending port S) allows for a user to send a block of data through a Data-request event, while the port on the right-hand side (receiving port R) allows for a user to receive a data block through a Data-indication event. For a given module, blocks are delivered in FIFO order, i.e., blocks sent through successive Data-request events are delivered in the same order as Data-indication events at the other end of the medium. The communication service provided by the Basic Medium is used by two types of processes, sender and receiver processes, which execute an unspecified application protocol.

B. Service Concatenation

The first interconnection scenario is trivial. Two Basic Media are interconnected as shown in Fig. 2. Data-indi-

Manuscript received October 15, 1988; revised August 20, 1989.

The authors are with the Département d'Informatique et de Recherche Opérationelle, Université de Montréal, Montréal, P.Q., Canada.

IEEE Log Number 8932038.

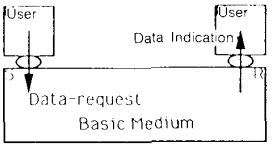


Fig. 1. Basic Medium.

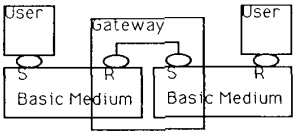


Fig. 2. Basic Media direct concatenation.

cation events at the receiving port of the left medium are identified with Data-request events at the sending port of the right medium. This kind of interconnection is called direct concatenation and is represented in Fig. 2 by the horizontal line between the two ports at the gateway site. Thus, a Data-request event at the sending port of the left medium leads to a Data-indication and a Data-request event at the interconnection point, and eventually to a Data-indication event at the receiving port of the right medium. Service concatenation and various related problems are investigated in Sections III and IV.

C. Service Enhancements

The following example introduces interconnection problems related to discrepancies in the communication services. The previous example is used with the following constraint. The left medium handles data blocks of size up to Large, while the right-hand side medium handles data blocks of size up to Small (with Small significantly smaller than Large). Therefore, data blocks of a size between Small and Large are handled by the left medium without any trouble, but cannot go further through the right medium. Three approaches can be used to solve this problem.

1) The simplest approach is to use the minimum service subset, i.e., to forbid users at the sending port of the left medium to request transmission for data blocks of a size greater than Small. However, this solution is not very powerful in the sense that the full capabilities of the communication services provided by the left-hand side medium are not used any more. Interconnection realized in such a way would force applications using these services to be modified, going against the transparency requirements.

2) A regional complementation protocol can be built on top of the less powerful medium, namely the right one, to segment blocks received from the left medium into blocks of a size no longer than Small (Fig. 3). This approach thus upgrades the lower level service to the level of the other one. An entity must reside on all systems within the less powerful network to realize this protocol. The dashed line in Fig. 3 between the two entities indicates that a protocol exists between these two entities, but

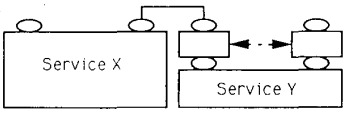


Fig. 3. Regional complementation protocol.

does not represent a real connection. All protocol data units (PDU's) are exchanged through the underlying service *Y*.

3) The last approach is quite similar to the second one. A complementation protocol is introduced throughout the entire system, on top of the minimum service subset (Fig. 4), and may also provide additional services not provided by any of the interconnected components. Additional entities in Fig. 4 realize a global complementation protocol providing a common service to all users.

Although quite simple, this example shows different problems related to transparent interconnection of communication services.

- It is necessary to cope with applications using existing services when interconnecting these services. In order to avoid a great deal of burden for interconnected applications, solution of the first type is to be avoided.
- The second solution is better suited, but the following question arises:
  - —What is the needed enhancement function to bridge the gap between the two services, i.e., to upgrade the lower level service up to the level of the other one?
- The third solution requires in addition the consideration of a common service subset:
  - —What is the common service subset?
  - —What are the desired services to provide?
  - —What is the necessary function to offer on top of the common service subset?

As previously shown, service enhancement approaches are based on concatenation. The second approach relies on concatenation of one "original" service with an upgraded service. The third approach relies on services concatenation with restriction to a common service subset; then additional facilities are built up on top. Therefore, concatenation is a central issue when interworking with different communication services.

The discussion in this paper mainly focuses on the services concatenation principles. It will be shown that careful analysis of concatenation properties is necessary before defining any complementation protocol.

D. Realistic Examples

The classical examples of gateways are used for computer network interconnection.

1) Network Layer: Within the OSI reference model, Network interconnection is foreseen through so-called Network relays. The general protocol hierarchy (Fig. 5) within the Network layer (OSI NL) foresees a combination of the approaches of Figs. 3 and 4.

- The SNAP (Sub-Network Access Protocol) sublayer

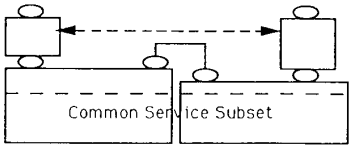


Fig. 4. Global complementation protocol.

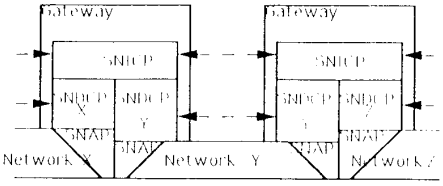


Fig. 5. OSI Network sublayers.

provides the interface to access a particular network (e.g., X.25).

- The SNDCP (Sub-Network Dependent Convergence Protocol) sublayer, corresponding to the complementation protocol of Fig. 3, is used to provide a uniform service over all networks participating in the interconnection, which individually may provide different types and grades of communication services.
- Sublayer SNICP (Sub-Network Independent Convergence Protocol) provides addressing (OSI NA) and other global functions for internetworking.

2) Transport Layer: Another example is interconnection at the Transport level. The OSI Transport service (OSI TS), similar in nature to the Network service, is a common service provided by the different classes of OSI Transport protocols (OSI TP) over different network configurations, and is used by all higher level protocols within the OSI reference model. For the interconnection of OSI systems using different classes of Transport protocols, e.g., class 4 within a local area network and class 0 or 2 over long distance networks, concatenation of transport services can be envisaged [3].

Interconnection at the Transport level can also be considered between the OSI and DARPA TCP/IP protocol hierarchies. TCP is a protocol similar to the OSI class 4 protocol, and its service is similar to the OSI Transport service. The selection of a suitable service subset including normal data transfer is, however, not straightforward [8], [2]. Some difficulties are related to the identification of message boundaries and addressing. The graceful close facility, in addition to the abort included in the subset, can be provided by use of the OSI Session Release function, according to Fig. 3.

3) Messaging Services: Another example corresponding to the protocol architecture of Fig. 3 is the Teletex access procedure to the message handling systems (MHS) as defined by CCITT in [18]. The messaging service to be accessed by that procedure includes functions not provided by Teletex, such as delivery confirmation, message store and retrieve, probes, · · · . Therefore, the approach of Fig. 3 was chosen where the defined complementation procedure is executed between the Teletex terminal and the MHS-Teletex conversion facility, which plays the role of a gateway between the MHS systems and the Teletex network.

III. Service Concatenation

Service concatenation allows us to build up a global communication service from two or more basic communication services. The interfaces of the global service offered to users are the same as those offered by the basic services. Concatenation is thus mainly concerned with "connecting" interfaces at gateway site(s).

To define services concatenation, we must first investigate how a service is defined, i.e., what its interfaces are, which events can occur at these interfaces, and which properties about occurrences of these events can be stated.

We then define three concatenation methods coping with connecting: 1) services with identical interfaces; 2) equivalent services with different interfaces; 3) services of different types.

The concept of concatenation invariance is then introduced and discussed with several examples.

A. Service Definition

Service definition includes two parts. The first is the abstract definition of the interfaces of the communication service. The second consists of the description of global properties relating events at the different interfaces.

1) User–Provider Interface: An interface does not always explicitly exist but defines the logical exchanges of information occurring between the user and the provider of the service. User and provider might be two logical parts of the same process, or two different processes communicating through an implementation of a real interface.

2) User-Provider Interactions: An interaction through an interface consists of an exchange of information. One side of the interface is responsible for initiating the exchange and selecting the appropriate values for the parameters to be exchanged. Therefore, an interaction—sometimes called service primitive—is described by its name, its parameters, and its initiator. Interactions are supposed to be executed in rendez-vous. Conceptually, only one interaction may occur at one time.

3) Temporal Ordering: The service specification also defines which sequences of interactions are legal. We can consider two types of ordering. Local ordering describes the possible sequences at one interface of a communication service. Global ordering describes the possible sequences of interactions at both interfaces of the communication service. Such orderings can be described by means of a finite state machine or any other suitable notation.

4) Other Service Properties: Usually, more specific properties must be defined for each interaction, in addition to the temporal ordering. Very often, service properties will correlate parameters of interactions at one interface of the service with parameters of other interactions at the same interface or at the remote one. As in the case of temporal ordering, the properties can be classified into local constraints pertaining to a single interface and global

constraints. In the following, we use the term global properties to denote the combination of the global ordering rules and the other global constraints. For instance, the Basic Medium satisfies the following two global properties.

- Every Data-request interaction initiated at the sending port eventually generates a Data-indication interaction at the receiving port (error-free medium).
- Two successive Data-request interactions initiated at the sending port eventually generate two successive Data-indication interactions (FIFO order).

5) Example: OSI Transport Service: As a realistic example, we consider the OSI Transport service as shown in Fig. 6. This service distinguishes two kinds of ports, so-called calling and called ports.

Interactions: A definition of this service for the calling user describes the possible interactions at the calling service interface and their parameters.

- *Interactions initiated by the calling user*
  - —TCONreq(calling, called: transport-address; QoS: quality-of-service; options: user-options; data: data-type)
  - —TDATAreq(user-data: data-type)
  - —TEXDTreq(user-data: data-type)
  - —TDISCreq(reason: data-type)
- *Interactions initiated by the provider at the calling port*
  - —TCONconf(calling, called: transport-address; QoS: quality-of-service; options: user-options; data: data-type)
  - —TDATAind(user-data: data-type)
  - —TEXDTind(user-data: data-type)
  - —TDISCind(origin: (user, provider); reason: data-type)

Interface for the called user is quite similar although the connection establishment phase involves different interactions.

- *Interactions initiated by the called user*
  - —TCONresp(calling, called: transport-address; QoS: quality-of-service; options: user-options; data: data-type)
  - —TDATAreq(data: data-type)
  - —TEXDTreq(user-data: data-type)
  - —TDISCreq(reason: data-type)
- *Interactions initiated by the provider at the called port*
  - —TCONind(calling, called: transport-address; QoS: quality-of-service; options: user-options; data: data-type)
  - —TDATAind(data: data-type)
  - —TEXDTind(user-data: data-type)
  - —TDISCind(orgin: (user, provider); reason: data-type)

Local Ordering: This "service definition" is completed with the "legal" sequences of interactions represented as a finite state machine in Fig. 7. The ordering rules for both kinds of interface are represented by the same diagram. Interactions that may occur at the calling port only are identified by a star symbol. Those occurring at the called port only are identified by an apostrophy symbol. It is to be noted that sequences of interactions occurring at both interfaces are not described.

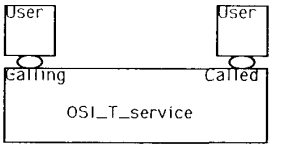


Fig. 6. OSI Transport service.

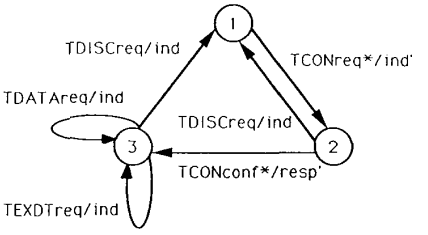


Fig. 7. Local ordering of Transport primitives.

Local Constraints: Additional constraints on interactions parameters are stated as follows.

- The *QoS* and *options* parameters of a TCONconf interaction must be "smaller than or equal to" the corresponding parameters of the TCONreq interaction.
- The *QoS* and *options* parameters of a TCONresp interaction must be "smaller than or equal to" the corresponding parameters of the TCONind interaction.

Global Properties: Global properties of the Transport services are informally stated.

1) Connection Establishment:

- A TCONreq interaction at the calling port implies a subsequent TCONind at the called port, unless a TDISCreq or TDISCind interaction occurs at the calling port.
- A TCONresp interaction implies a TCONconf interaction at the calling port unless a TDISCreq or TDISCind interaction previously occurred at that port.

2) Data Transfer:

- A TDATAreq (respectively, TEXDTreq) interaction generates a remote TDATAind (respectively, TEXDTind) interaction with the same *data* parameter value (reliable transfer).
- Two successive TDATAreq interactions from the same user will generate two successive TDATAind interactions, in the same order (FIFO property—normal data)
- Two successive TEXDTreq interactions from the same user will generate two successive TEXDTind interactions, in the same order (FIFO property—expedited data).
- A TEXDTreq following a TDATAreq may generate a TEXDTind before the TDATAind is generated (expedited data transfer may bypass normal data transfer).

3) Disconnection:

- A provider-initiated disconnection leads to the oc-

currence of a TDISCind interaction at the ports where no TDISCreq interaction already occurred.

- A TDISCreq interaction will generate a remote TDISCind interaction if no provider-initiated disconnection occurs at the same time and no TDISCreq interaction occurred at the remote port.
- A provider or user initiated disconnection may imply some loss of in-transit data.
- The TDISCind interaction indicates with the *origin* parameter whether the disconnection was initiated by the remote user or by the service provider. In the first case, the *reason* parameter is the *reason* parameter of the corresponding TDISCreq interaction. In the latter case, the *reason* parameter is selected by the service provider. When two TDISCind interactions are initiated at both ends, the *origin* parameter value must be "provider."

B. Concatenation Methods

Three different methods of concatenation are discussed, depending on the services to be concatenated (type of provided services, interfaces).

1) Direct Concatenation: The example of the interconnection of two Basic Media is very simple since interactions at both interfaces can be directly identified. We call direct concatenation an interconnection of two communication services where each interaction of one interface is identified with (mapped onto) a corresponding interaction of the other connected interface. Another example is the Transport service previously described. The mapping to be performed by the gateway between the two interfaces (one calling, the other called) is simply the following.

- TCONind interactions are mapped onto TCONreq interactions.
- TCONconf interactions are mapped onto TCONresp interactions.
- TDATAind interactions are mapped onto TDATAreq interactions.
- TEXDTind interactions are mapped onto TEXDTreq interactions.
- TDISCind interactions are mapped onto TDISCreq interactions.

In general, two interfaces A and B can be directly concatenated if the following conditions hold.

1) Interactions initiated by the provider at interface A (respectively, B) can be mapped onto interactions initiated by the user at interface B (respectively, A); this should include a mapping of corresponding parameters of the mapped interactions.

2) Legal sequences of interactions initiated by the provider at interface A (respectively, B) are "consistent" with the legal sequences of interactions that could be initiated by a user at interface B (respectively, A).

It is interesting to note that direct concatenation is not completely possible for the OSI Transport service. In fact, the parameters of a TDISCind cannot be directly mapped onto a parameter of the TDISCreq. According to the OSI specification, a TDISCreq is always interpreted by the service provider as having a user origin. If a TDISCind indicates a provider-initiated disconnection, the information cannot be transmitted in the parameter of the corresponding TDISCreq.

2) Interface Adaptation: As an example of different interfaces providing the same abstract service, we consider an alternative description of the Transport service interface. The TCONreq and TCONconf interactions are replaced by the following procedure.

- TCALL (calling, called: transport-address;
  proposed-QoS: quality-of-service;
  selected-QoS: quality-of-service;
  proposed-options: user-options;
  selected-options: user-options;
  status: boolean)

The *status* parameter is returned as true (respectively, false) after successful (respectively, failed) establishment of the Transport connection. The *proposed-QoS* and *proposed-options* parameters are set to appropriate values by the calling user. The *selected-QoS* and *selected-options* parameters indicate which values have been selected for the established connection. We note that calling this procedure is equivalent to initiating the TCONreq interaction, and that returning from it is equivalent to the occurrence of the TCONconf (respectively, TDISCind) interaction.

This alternative Transport interface cannot be directly concatenated with the interface described in Section III-A-5). However, service concatenation may be realized by an interface adapter, as shown in Fig. 8, which calls the TCALL procedure after the occurrence of a TCONind at the called interface, and initiates a TCONresp or TDISCreq at this interface when the procedure returns control.

This interface adaptation provides a local mapping within the gateway between the concatenated service interfaces. It differs from the function of an enhancement entity which executes a protocol in relation to other remote system components, as shown in Fig. 3.

In general, identical services can be concatenated through an interface adapter module if:

1) abstract interactions can be "extracted" from the two interface specifications; and

2) the conditions defined for direct concatenation in Section III-B-1) hold for this set of interactions.

3) Service Adaptation: In the above example the two concatenated services are not identical, but abstractly equivalent since some equivalence relation between the interactions at the different interfaces could be defined. The same approach can also be used for the minimum subset service if the services to be concatenated are not equivalent. As an example, we consider the concatenation between the Basic Medium and the OSI Transport service, as shown in Figs. 9 and 10. Without using a service enhancement, as discussed in Section II-C, we are restricted to using the minimum service subset, which is in this case equal to the service provided by the Basic Medium. We therefore consider the TDATAreq (respec-

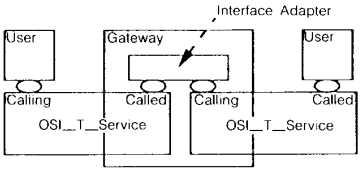


Fig. 8. Concatenation with interface adapter.

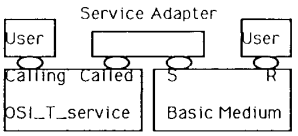


Fig. 9. OSI Transport and Basic Medium.

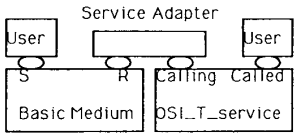


Fig. 10. Basic Medium and OSI Transport.

tively, TDATAind) interaction equivalent to the Data-request (respectively, Data-indication) interaction. If these were the only interactions to be considered, the direct concatenation approach could be used for the gateways in Figs. 9 and 10 for providing the basic subset service.

However, the OSI Transport service requires a connection to be established before the TDATAreq and TDATAind interactions can occur. Therefore, the gateway architecture with an adapter module (see Fig. 8) may be used where these additional interactions of the Transport service are handled by the adapter module. If the left user in Fig. 9, for example, wants to send data to the right user, he/she will first establish a connection to the gateway through the Transport service. The adapter module will accept the incoming TCONind interaction by issuing a TCONresp if the parameters are acceptable, e.g., the called address is recognized as reachable through the Basic Medium service; otherwise a TDISCreq will be issued. Incoming TDATAind and TEXDTind interactions will be forwarded by the adapter as Data-requests over the Basic Medium, while an incoming TDISCind interaction can be ignored.

In the case of data transfer in the opposite direction (see Fig. 10), the adaptation is less straightforward. The adapter module in the gateway has to decide when and for how long to establish Transport connections to the receiving user. When it receives a Data-indication and no Transport connection is established, a new connection must be set up before the data can be forwarded. Possible strategies for terminating the connection are the following, all transparent to the users.

- The Transport connection is maintained forever, or at least until the Transport service user explicitly requests a disconnection.
- The connection is kept, based on a time limit or an expected number of data transfer requests.
- The connection is released immediately after the transfer of one block of data.

C. Concatenation Invariance

We say that a global property of a communication service is concatenation invariant if it remains satisfied on an end-to-end basis over several concatenated communication services and it is satisfied by each service individually. In the following we consider different examples of concatenation to question whether invariance can be guaranteed. As will be shown, the answer turns out to be "sometimes."

1) An Example: Delivery Confirmation: The first example is derived from the Basic Medium described in Section II-A. A new interaction called Data-confirmation is added which provides the sending user with a kind of delivery confirmation. The following three different types of confirmation can be considered.

- *Type A:* This first type of confirmation informs the user that a data block has been received by the network and will be forwarded to the destination user (Fig. 11).
- *Type B:* It informs the sending user that a data block has been received by the receiving user (Fig. 12).
- *Type C:* This third type of confirmation is only issued after the receiving user has explicitly acknowledged the receipt of the data by initiating a Data-response interaction (Fig. 13).

Clearly, these different confirmations have different semantics. The first type of confirmation has a local confirmation meaning, the second one has a remote confirmation meaning, and the third one has a user-to-user confirmation meaning. In Figs. 11–17, solid (respectively, dashed) arrows denote Data-request and Data-indication (respectively, Data-response and Data-confirmation) interactions. Numbers associated with arrows represent a relative temporal ordering. An arrow without any associated number denotes an interaction for which no relative temporal ordering is defined.

We consider in the following four interconnection scenarios. The three first scenarios consider concatenation of identical services. The mapping occurring at the gateway site is explained case by case. The last scenario considers interconnection of two different services.

- *Type A Confirmation:* The gateway maps Data-indication interactions from the left service to Data-request interactions to the right service. The Data-confirmation interaction is not mapped onto any other interaction (see Fig. 14). Here the Data-confirmation has no global significance, only a local one. Therefore, concatenation invariance holds vacuously.
- *Type B Confirmation:* The same mapping as in the previous case is considered. Here, a Data-confirmation delivered to a sending user does not imply that the sent data were delivered to the receiving user (Fig. 15); it only implies that the data were received by the gateway. Globally, therefore, the meaning of the confirmation loses its

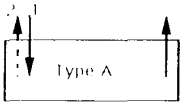


Fig. 11. Local delivery confirmation.

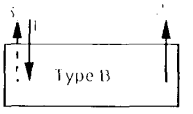


Fig. 12. Remote delivery confirmation.

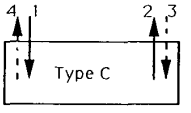


Fig. 13. User-to-user delivery confirmation.

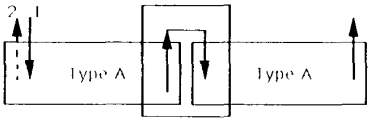


Fig. 14. Type A service concatenation.

"remote" character; however, it implies a Type A confirmation. We conclude that the global property of the Type B Data-confirmation is not concatenation invariant.

- *Type C Confirmation:* A direct mapping between a Data-confirmation interaction from the right service onto a Data-response interaction to the left service ensures that the Data-confirmation delivered to the sending user has a user-to-user meaning (Fig. 16). Here the sending user receives the confirmation only after the receipt of the data by the receiver. The global confirmation property is therefore concatenation invariant.
- *Type A and C Confirmation:* In this case (Fig. 17), it is obvious that the user-to-user confirmation meaning cannot be provided whatever the mapping at the gateway site is. This was expected since one of the concatenated services alone does not provide such a confirmation.

2) Other Examples: It seems that most global properties of communication services are concatenation invariant. Simple data transfer, as represented by the Basic Medium clearly is. The previous discussion shows that delivery confirmation of Type C is a "good" service since it is concatenation invariant, which simplifies interworking.

Another important example is flow control. The OSI Network and Transport services have back-pressure flow control at the interfaces. The local constraint requires that no data can be sent over an interface when the receiving side invokes flow control. The global property of the communication service postulates that the service provider may—and eventually will—invoke flow control at its interface with a sending user in response to flow control invoked by the receiving user at the remote interface. In

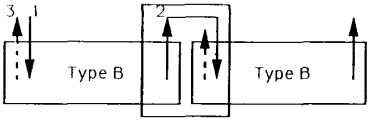


Fig. 15. Type B service concatenation.

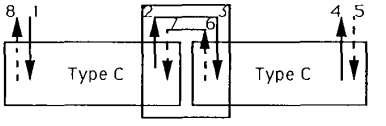


Fig. 16. Type C service concatenation.

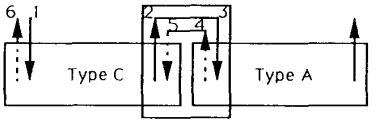


Fig. 17. Type A and C service concatenation.

a straightforward implementation of flow control without buffering in the gateway, a gateway will invoke flow control at the interface where it receives data when flow control is invoked at the interface where it sends. This means that flow control invoked by a receiving user over a concatenated communication service usually will give rise to flow control invoked at the interface of the sending user. This means that back-pressure flow control is concatenation invariant.

In the case of the OSI Transport protocol discussed in Section III-A-5), all global properties are concatenation invariant, except the significance of the *origin* parameter of the TDISCind interaction. As already mentioned in Section III-B-1), this parameter does not allow for direct concatenation. Here, problems arise when one of the two Transport services initiates a disconnection due to some internal error. A TDISCind interaction occurs at both ends of the service, and the gateway thus has to trigger a TDISCreq interaction at the interface of the other service. The *origin* parameter of the TDISCind interaction indicates a provider-initiated disconnection, with the *reason* parameter detailing the cause of the disconnection. The TDISC-req interaction eventually generates a TDISCind interaction to the remote user with the *origin* parameter indicating a user-initiated disconnection (Fig. 18). Thus, the two users have a different view of the disconnection. Properties stated for the disconnection service are not maintained when Transport services are concatenated.

The easiest solution to this problem seems to be the addition of an *origin* parameter to the TDISCreq interaction such that direct concatenation becomes possible at the gateway. The gateway may then use "provider" as a value of the *origin* parameter of the TDISCreq generated in the case described above.

3) Handling Noninvariant Properties: As shown by the example of interworking between two networks providing Type B confirmation, not all global properties of con-

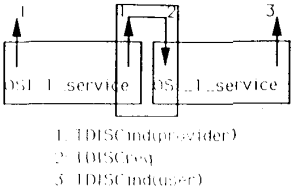


Fig. 18. Transport disconnection.

munication services remain invariant under concatenation. If such noninvariant properties are nevertheless required for the resulting end-to-end communication service, it is necessary to use a complementation protocol, as discussed in Section II-C. In the case of Type B confirmation, there are two possible architectures.

- A new Data-confirmation primitive with end-to-end significance may be provided by a global complementation protocol on top of the common subset service as shown in Fig. 4. In this case the confirmation services of the concatenated services are not used at all.
- A complementation protocol that operates only over the left network shown in Fig. 15 (similar to Fig. 3) provides a Type C confirmation. This enhanced service is used by the gateway to forward the Type B confirmation received from the right network of Fig. 15 to the sending user on the left network.

It is to be noted that both of these complementation protocols require a data transmission service from right to left from the underlying network, while the Basic Medium assumed in our example only provides this service from left to right. However, any realistic network will provide data transfer in both directions; our example is over-simplified.

4) Quantitative Properties: Noninvariant: The global properties discussed above are of qualitative nature. Quantitative properties are also important, in particular for performance considerations. It is important to note that the latter properties are not concatenation invariant; instead they behave in an additive manner, or some other numerical fashion.

The most important quantitative properties of a communication link are its delay and maximum throughput. If we concatenate two such links with delays $D1$ and $D2$, and maximum throughputs $T1$ and $T2$, respectively, and if we can ignore the degrading influence of the gateway, we obtain an end-to-end communication service with a delay equal to $D1 + D2$ and a maximum throughput equal to the minimum of $T1$ and $T2$. This shows that the delay is additive under concatenation, while the throughput follows the minimum limit—similar to the maximum data block length discussed in Section II-C.

IV. Service Adaptation Versus Protocol Conversion

The adaptation issues so far discussed rarely required the consideration of protocols but only of services. One objective of this paper is to point out that most compatibility issues for interworking and gateways can be discussed in terms of compatibility of services.

A. Service Adaptation

The easiest way to convert between different protocols is through a common service boundary, as previously discussed. The example of the OSI Transport service and protocols showed this clearly. Fig. 19 shows a possible architecture for interworking between different protocol hierarchies used in different networks. The conversion is done at a level n assuming that the protocols in the two networks above that level are compatible. The gateway includes implementations of both protocol hierarchies, up to layer n, and provides for concatenation at the (N)-service level. We call this approach a gateway based on service adaptation (called service interface conversion in [10]).

B. Protocol Conversion

It is possible to discuss the interworking of networks with different protocols directly at the level of the involved protocols. We call protocol conversion or conversion at the PDU level a situation where the gateway function is defined explicitly in terms of the PDU's which are exchanged within the two interconnected networks according to their respective protocols. This approach is assumed in [8], [9], and [11]. Various alternatives for Transport gateways, for instance, are discussed in [3].

It is important to note that conversion at the PDU level is generally more complex to specify and more difficult to implement. The advantage of service adaptation is that existing protocol implementations can be used within the gateway; only an adaptation between the two service interfaces is to be provided. The latter can be quite simple if the service interfaces of the respective protocol implementations are similar.

An architecture for PDU-level conversion is shown in Fig. 20. The gateway includes separate implementations of the lower layers protocols, up to and including layer $(N = 1)$, and the protocol adapter module which does the PDU conversion for the respective level N protocols. Adapters providing PDU conversion for several layers of protocols together can also be considered [10]. The comparison of Figs. 19 and 20 yields the following important fact about the protocol converter. The behavior of this converter module should satisfy all the properties defined by the behavior of the two respective protocol entities interconnected at their respective service interfaces through the service interface adapter as shown by the dotted box in Fig. 19.

The above fact can be used to derive a specification of the protocol adapter from the two protocol specifications for which interworking is desired [4]. Such a specification is simply given by the composition, as shown by the dotted part of Fig. 19, of the protocol specifications—defining the behavior of the protocol entities at layer N—with the specification of the service interface adapter.

In addition to the properties that follow from the spec-

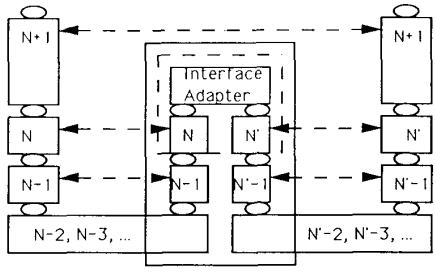


Fig. 19. Service adaptation architecture.

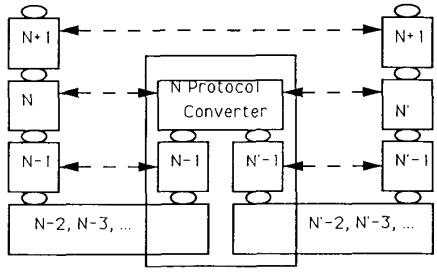


Fig. 20. Protocol conversion architecture.

ifications of the interconnected protocols, a protocol converter may satisfy certain other properties which are related to interworking strategies and optimizations. Different optimizations concerning acknowledgments are discussed in [3]. Different ways to optimize the implementation structure of the protocol adapters are discussed in [10].

V. Conclusion

The main conclusions from the discussion in this paper are the following.

1) Interworking is relatively straightforward for the subset of communication services which are available in all interworking systems.

2) Interworking may be realized at different levels within the protocols hierarchy. Above the chosen level, the protocols of the interworking systems must be compatible.

3) Certain differences in the communication services for which interworking is desired can be accommodated through additional complementing protocol sublayers which are especially introduced in one of the networks to obtain a communication service equivalent to the one available in the other one. This requires additional sublayer entities in the gateway and the host computers participating in interworking activities.

4) Those properties of the interconnected communication services that do not have the so-called concatenation invariance will not be preserved in the case of interworking. If all properties are concatenation invariant the user processes using the service do not see any qualitative difference between "local" communication and "far distance" interworking.

5) Gateway construction is the simplest if a service concatenation approach is used (service adaptation). In this case, it is assumed that access to the communication services of both systems is available within the gateway through existing protocol implementations. The gateway design is particularly simple in the case of direct concatenation which requires certain matching conditions for the service interfaces.

6) If the gateway considers in detail the PDU's exchanged below the service level at which interworking is sought (conversion at the PDU level), it may provide a more efficient and powerful adaptation function; but it is in most cases more difficult to design and build.

7) In general, the specification of the protocol converter for PDU-level conversion can be obtained automatically from the two protocol specifications and the interface adapter.

It is interesting to note that only in points 2), 6), and 7) the detailed consideration of the respective protocols is important; all other points are related to the definition of the communication services used in the interconnected systems. This emphasizes the importance of service specifications for the design of distributed systems. While meaningful communication in a distributed system depends on compatible implementations of an agreed-upon protocol, the possibility of building gateways that extend a communication service into adjacent distributed systems depends essentially on the compatibility of the communication services used in the different systems.

If we consider, on the other hand, the direct interworking (without gateways) between different computer systems, the conditions for compatibility are related mainly to the communication protocols. For example, the condition that a system can exchange and access files with another OSI FTAM system includes the following:

1) the OSI protocols for all layers must be implemented correctly, including all options required by FTAM; and

2) the operation of the system as seen through these protocols must conform to the FTAM virtual file system.

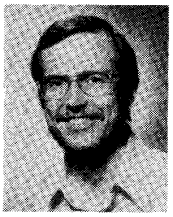
While point 1) includes seven layers of protocols, point 2) corresponds to the conformance with a (local) service specification, namely, the service provided by the file system.

It is important to note that service specifications are not only essential for interworking considerations, but also for the design and validation of a hierarchy of protocols [1]. In fact, the service specification is not only the reference for interworking considerations but also for validating the protocol which is intended to provide the service and for checking that it is sufficient for the operation of the user processes. The service specifications for the different layers also provide the framework in which new protocols may be introduced in a specific layer without affecting other layers.

References

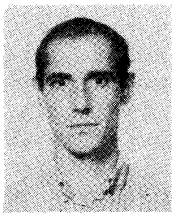
[1] G. v. Bochmann and C. A. Sunshine, "Formal methods in communication protocol design," *IEEE Trans. Commun.*, vol. COM-28, pp.

624–631, Apr. 1980; reprinted in *Communication Protocol Modelling*," C. Sunshine, Ed. New York: Artech, 1981.
[2] G. v. Bochmann, A. Jacques, and C. Kawa, "Transport relays," Univ. Montreal, Tech. Rep., 1986.
[3] G. v. Bochmann and A. Jacques, "Gateways for the OSI Transport Service," in *Proc. IEEE INFOCOM'87 Conf.*, San Francisco, CA, 1987.
[4] G. v. Bochmann, "Deriving protocols converters for communication gateways," *IEEE Trans. Commun.*, to be published.
[5] M. Gien and H. Zimmerman, "Design principles for network interconnection," in *Proc. VI Data Commun. Symp. (IEEE/ACM)*, Nov. 1979, pp. 109–119.
[6] P. E. Green, "Protocol conversion," *IEEE Trans. Commun.*, vol. COM-34, pp. 257–268, Mar. 1986.
[7] —, *Network Interconnection and Protocol Conversion*. New York: IEEE Press, Apr. 1987.
[8] I. Groenback, "Conversion between TCP and ISO transport protocols," *IEEE Trans. Select. Areas Commun.*, vol. SAC-4, pp. 288–297, Mar. 1984.
[9] S. Lam, "Protocol conversion," *IEEE Trans. Software Eng.*, vol. 13, pp. 353–362, Mar. 1988.
[10] Y. Ohara, S. Yoshitake, and T. Kawaoka, "Protocol conversion method for heterogeneous systems interconnection in multi-profile environment," in *Proc. IFIP Symp. Protocol Specification, Testing and Verification: VII*, Zurich, May 1987. Amsterdam, The Netherlands: North-Holland, pp. 405–418.
[11] K. Okumara, "A formal protocol conversion method," in *Proc. ACM Conf. Commun. Protocols and Architectures*, 1986.
[12] ISO, "Addendum to the Network Service Specification covering Network Layer Addressing," ISODP 8348, Apr. 1984.
[13] ISO, "Internal Organization of the Network Layer," ISOTC97/SC6 N3141, Apr. 1984.
[14] ISO, "Basic Reference Model for OSI," IS 7498, 1982.
[15] ISO, "Connection Oriented Transport Protocol Specification," IS 8073.
[16] ISO, "Connection Oriented Transport Service Specification," IS 8072.
[17] C. Vissers and L. Logrippo, "The importance of the concept of service in the design of data communication protocols," presented at the IFIP Workshop on Protocol Specification, Verification and Testing, Toulouse, France. Amsterdam, The Netherlands: North-Holland, 1985.
[18] CCITT Recommendations, "Message handling systems: Access protocol for Teletex terminals."
[19] S. Zatti and P. Janson, "Interconnecting OSI and non-OSI networks using an Integrated Directory Service," *Comput. Networks and ISDN Syst.*, vol. 15, pp. 269–283, 1988.



Gregor v. Bochmann (M'82–SM'85) received the Diploma degree in physics from the University of Munich, Munich, West Germany, in 1968 and the Ph.D. degree from McGill University, Montreal, P.Q., Canada, in 1971.

He has worked in the areas of programming languages, compiler design, communication protocols, and software engineering and has published many papers in these areas. He is currently a Professor in the Département d'Informatique et de Recherche Opérationelle, Université de Montréal, Montréal, P.Q., Canada. His present work is aimed at design models for communication protocols and distributed systems. He has been actively involved in the standardization of formal description techniques for OSI. From 1977 to 1978 he was a Visiting Professor at the Ecole Polytechnique Federale, Lausanne, Switzerland. From 1979 to 1980 he was a Visiting Professor in the Computer Systems Laboratory, Stanford University, Stanford, CA. From 1986 to 1987 he was a Visiting Researcher at Siemens, Munich.



Pierre Mondain-Monval received the Maitrise degree in computer science from the University of Bordeaux, Bordeaux, France, in 1983 and the Doctor degree from the University of Toulouse, Toulouse, France, in 1987.

From 1984 to 1987, he was at the Laboratoire d'Automatique et d'Analyse des Systemes, Toulouse, France. In 1988 he was a Visiting Professor at the University of Delaware, Newark, and is currently a Professor in the Département d'Informatique et de Recherche Opérationelle, Université de Montréal, Montréal, P.Q., Canada. His research topics include standardized OSI protocols design and specification, distributed systems, and software engineering.