ELSEVIER

The Journal of Systems and Software 47 (1999) 53–69

The Journal of Systems and Software

www.elsevier.com/locate/jss

Synthesis of communications protocol converters using the timed Petri net model

Mansour Jaragh, Kassem Saleh [*]

Department of Electrical and Computer Engineering, Kuwait University, P.O. Box 5969, 13060 Safat, Kuwait

Received 17 June 1997; received in revised form 5 November 1997; accepted 8 December 1997

Abstract

The proliferation of heterogeneous, distributed computer networks has led to an urgent need for constructing reliable and efficient communication protocol converter, to facilitate the internetworking between such networks. Most existing converter design methods are based on the communicating finite state machine (CFSM) model as the formal description technique to describe the protocols, their services and the converter design. Two drawbacks of CFSM model are the state explosion problem and the inability of the model to express concurrent behaviors of protocols and services. These drawbacks may be overcome by using Petri nets as the formal description technique. Moreover, a reliable converter must perform its conversion functions in a timely manner satisfying the timing constraints of both protocol architectures. Our paper adopts the Timed Petri net (TPN) model to fulfill this requirement. The paper also highlights the dynamicity of the derived converter. We illustrate the converter design method using an example showing the dynamicity and timeliness features of the converter. © 1999 Elsevier Science Inc. All rights reserved.

1. Introduction

The proliferation of heterogeneous, distributed network architectures has made the communication between users residing on different networks often impossible. Although proprietary communication architectures often offer similar services, the protocols implementing them are incompatible for trivial or non-trivial reasons. Current and projected future trends in the information technology market are favorable for the provision and implementation of value-added services and applications across various communication architectures, networks and platforms. To ensure the universality of such services, there is a need to interconnect the networks providing such services. This can be achieved using protocol converters. The development of such converters should be considered as a short to mid-term solution in contrast to the long term solution involving the standardization of network interfaces over which value-added services can be supported Saleh and Ural (1994).

The need for developing formal methods for protocol converter design was first pointed by Green (1986). In his paper, he defined the problem of protocol conversion and the need for facilitating the interoperability between different network protocols (Liu, 1990; Stallings). Since then, the issue of converter design and network interoperability has been addressed extensively in the literature, and various formal design techniques have been proposed Peyravian and Lea (1993) and Lam (1988). Two major concerns are being addressed. First, the architectural concerns which deal with the identification of the layer at which a converter must exist (Peyravian and Lea, 1993; Bochmann and Mondain-Monval, 1990; Tillman and Yen, 1990; Ohara et al., 1987). Second, the behavioral concerns which deal with the reconciliation between the different behaviors of protocols manifested by the different protocol message formats and their orderings. Many techniques address the second concern. These techniques are either top-down or bottom-up. Top-down service-oriented techniques attempt to generate communication gateways from formal service specification (Kristol et al., 1993; Bochmann, 1990; Calvert and Lam, 1989; Jeng and Liu, 1992). Bottom-up protocol-oriented techniques do not consider the complete service specification, but attempt to reconciliate at the protocol messages level (Lam, 1988; Calvert and Lam, 1990a; Rajagopal and Miller, 1991). Other service-based approaches use the concept of a service relay which translates each of the

[*] Corresponding author. Tel.: +965 481 1188; fax: +965 481 7451; e-mail: ksaleh@eng.kuniv.edu.kw

service primitive emanating from one network to a format acceptable by the other network. However, this is not a simple task since services can have different options, quality of services and classes. Other solutions are mainly based on local or global service complementation in which additional sub-layer(s) may be required to ensure the service compatibility (Bochmann and Mondain-Monval, 1990). Surveys of existing converter design techniques can be found in Liu (1990), Peyravian and Lea (1993) and Calvert and Lam (1990b).

A new approach to protocol conversion, namely the hybrid approach, has also been advocated. The hybrid technique has adopted some features of the top-down as well as the bottom-up approaches (Kristol et al., 1993; Okumara, 1990). The drawbacks of the two approaches can be overcome by this technique. In this approach, the converter synthesis procedure requires the formal service specifications as well as the protocol specifications as input requirements at the initial stages of the converter design. A desirable feature of hybrid conversion is its high degree of transparency at each stage of the conversion – the details of message conversions and interactions at the protocol level are exposed, and the service user no longer views the protocol as a hidden 'black box'.

In this paper, we introduce a new converter synthesis technique. We have adopted a hybrid approach. Our method starts with the specifications of the service to be provided by the communicating protocols and later applies the protocol specifications projecting peer-to-peer protocol interactions, to derive the required protocol converter. This method is an enhancement of our earlier work using the communicating finite state machine model (CFSM) (Saleh et al., 1995). The synthesis problem is formulated as: 'the design of a protocol converter for the interworking between two incompatible protocols, at layers N and M, starting from the formal specification of these protocols and the services they provide'. Petri nets will be used to model all communications components needed for the construction of the converter. As mentioned earlier, this model is well-known for its formal mathematical and semantic foundations, and can express concurrency, communication and synchronization in a natural way.

The rest of this paper is organized as follows. Section 2 provides some preliminary background on networks, their architectures and their interconnections. Also, a brief review of existing work on protocol conversion is presented. Section 3 discusses converter properties and various design issues to be considered during the converter design process. Section 4 provides a formal definition of the specification model and its related operations to be used in the design of Petri net-based protocol converters. Section 5 highlights two desirable features of the converter to be designed, namely, the dynamicity and the timeliness. Section 6 introduces the Petri net-based converter synthesis technique and analyzes its complexity. Section 7 provides an example to illustrate our proposed technique. Finally, we conclude the paper in Section 8.

2. Background

In this section, we provide some preliminary background on the issues related to protocols and protocol conversion.

2.1. The protocol and service concepts

A network provides an infrastructure to facilitate the communication among distributed computer systems. A protocol consists of a specified set of rules which govern the orderly exchange of messages among the users of these systems. The underlying network providing the communication services to the users is organized as a stack of layers to reduce the design complexity. Each layer offers services to the next higher layer. The active elements in each layer are called entities. An entity in layer N provides services to the entity in layer $(N + 1)$. The SAPs (service access points) of layer N are the logical places through which layer $(N + 1)$ can access the services offered by layer N.

The relationship between protocols and services can be described at two levels of abstraction. At a high level of abstraction, a communication system can be viewed as a service provider which offers some specified communication services to a number of service users $(U_1, U_2, \ldots, U_n)$ who access the system through many geographically distributed service access points $(\mathrm{SAP}_1, \mathrm{SAP}_2, \ldots, \mathrm{SAP}_n)$. At a lower level of abstraction, the communication system can be seen to consist of a number of cooperating protocol entities (PEs) which exchange protocol messages, called protocol data units (PDUs), that are not observable to the users at the upper access points. The PEs exchange these messages over a reliable communication medium according to a First-In-First-Out (FIFO) discipline. These PEs have their own service access points for accessing the FIFO medium called lower SAPs.

The communication service specification describes the distributed functions provided by the communication system to its service users, whereas, the communication protocol specification describes the behavior of each of the protocol entities, each servicing a particular access point. A protocol entity specification describes the behavior of that entity with respect to its upper interface (SAPs) and with respect to its lower interface with the underlying service provider.

2.2. The protocol conversion problem

Due to the incompatibility between protocols, protocol conversion has become a necessity for the internetworking in a heterogeneous network environment. The term 'protocol conversion' implies constructing a converter as a mediator between two given protocols. It accepts messages from either protocol, interprets them and delivers appropriate messages to the other peer protocol. Green (1986) analyzed different internetworking architectures and situations in which conversion has to be performed. Suppose we have two networks based on different layered communication architecture. Two distributed users, each belonging to a different network, will not be able to interconnect and communicate because of the incompatibilities at either or both the protocol or service levels. A protocol converter can be used here to reconciliate between the messages exchanged at the boundaries of both networks. A converter communicates with each network in its own language and translates between incompatible networks. Thus it provides a common communication service throughout the heterogeneous interconnected system. As the different networks may have different protocols, a converter performs conversion between incompatible protocols. The design of the converter functions depends mainly on the extent of disagreement between the protocol layers at which the conversion is made.

2.3. Converter design approaches

The converter can be designed according to two architecturally and fundamentally distinct design approaches, namely, the service level and the protocol level. A third approach is the hybrid approach, which merges the first two approaches.

In service level conversion, corresponding (common) services from the two protocols are concatenated. This method maps the service primitives of the two protocols. The conversion is performed at level N assuming that the protocols of the networks above that level are compatible. The gateway consists of the implementations of the two protocols hierarchies up to the conversion level, and includes a service interface adaptor at (N)-service level. Fig. 1 shows a service level conversion architecture. This method does not include the details of the PDU sequences and control associated with these message exchanges in the early design stage.

A drawback of the service-level approach is that the interactions of the protocol converter with the two networks are strictly confined to the service user level. Therefore, the converter can only play a passive role in the conversion process, i.e., it can never have a choice of initiating an action by delivering a PDU from one network to the other, since this method does not involve transactions at the protocol level.

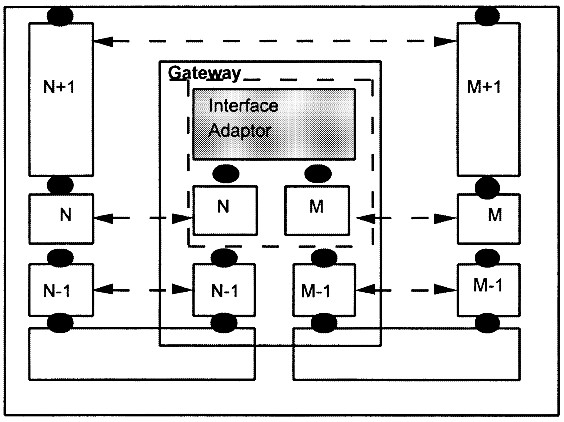


Fig. 1. Service-level conversion in a global communication service.

In protocol- or PDU- level conversion, the interoperability is based on a conversion at the level of the protocols involved. The gateway function is defined explicitly in terms of the PDUs exchanged within the two interconnected networks at protocol layer N, above which all protocols are compatible. Fig. 2 shows the protocol level conversion, in which a converter exists at layers N and M of Networks A and B, respectively.

An equivalent abstract representation of the system, in which only the incompatible protocol layers appear, is shown in Fig. 3. This figure also shows various observation points which will be used later to synthesize a converter. This method may provide a more efficient and powerful conversion, but it is more complex to be specified and more difficult to implement. For more details on related issues and comparison between the two conversion approaches, refer to Bochmann's paper (Bochmann and Mondain-Monval, 1990).

One disadvantage of choosing a protocol-level converter synthesis method is that the generated converter may remain a passive one, i.e., it may fail to exhibit a dynamic behavior. This is because the protocol-level conversion method ignores all transactions between the two networks at the service user level, and as a result,

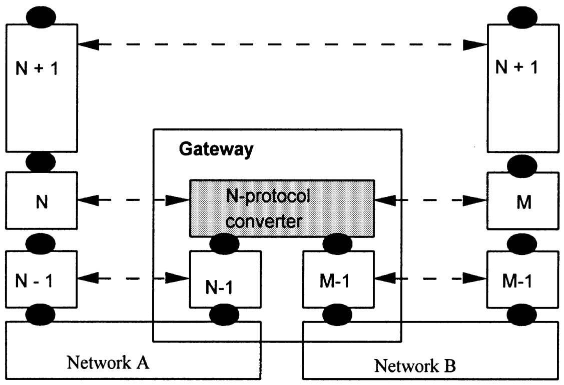


Fig. 2. Protocol-level conversion in a global communication system.

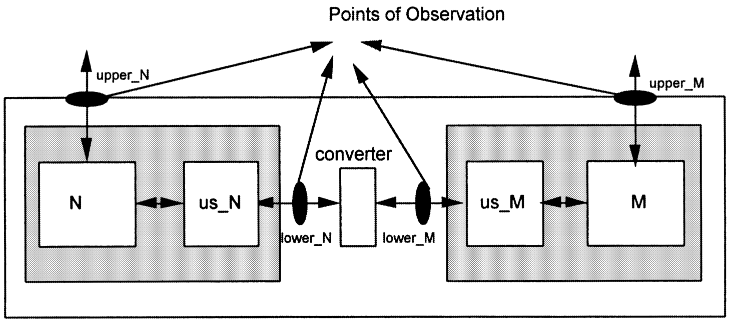


Fig. 3. Abstract representation of a global communication service.

the synthesized converter merely performs a 'blind', end-to-end data translation and transfer between the two incompatible protocols. Hence, such a converter may never be able to initiate an action during the conversion process.

Hybrid conversion merges the service level approach and the protocol level approach. The converter synthesis process begins at the service level, but continues down to the protocol level, and the generated converter reflects transaction details at the service level as well as the protocol level. Hence, the drawbacks of the two approaches can be overcome by adopting the hybrid approach. Moreover, a converter synthesized using the hybrid approach will be able to function dynamically. This paper introduces a protocol converter design method that uses the hybrid approach.

2.4. An overview of existing work

Green's converter design approach in 1986 was soon followed by several formal methods for constructing protocol converters. Though most of the initially proposed methods provided ad-hoc solutions, a number of conversion algorithms have been proposed during the past decade based upon two fundamentally different approaches, namely, the protocol data unit (PDU) level approach and the service level approach. The PDU level approach was introduced in the late 1980s. Okumara (1986) proposed the use of a finite state machine-based input specification, called the 'conversion seed'. The conversion seed specifies the constraints on the order of messages exchanged between the two protocols via the converter. Lam (1988) proposed a formal model based upon a projection theory, for reasoning about the semantics of different protocols and conversion between them. In 1990, Yao, Chen and Liu proposed a modular approach to construct a converter for two given protocols which passes through different phases, performing separate functions in each phase (Jeng and Liu, 1992). Rajagopal and Miller (1991) proposed an approach to synthesize a protocol converter from executable protocol traces. They derived the message relationships between the given two protocols by using a protocol analysis methodology. Their method includes the disjoint functions into the converter.

Recently, more attention has been paid to the service level approach. These methods aim at deriving a protocol converter assuming that the communication services of the interoperating systems are given as input specifications. In 1989, Calvert and Lam proposed a quotient algorithm to synthesize a converter from the given protocoal and service specifications (Calvert and Lam, 1990b). Bochmann (1990) suggested a gateway synthesis approach using a service interface adaptor that maps compatible service primitives between the two systems. Jeng and Liu (1992) proposed a five-step algorithm to formally derive a protocol converter from the service specifications. It has been observed that a service-level algorithm has less complexity at the design stage.

Most of the methods proposed over the last decade used finite state machines as the modeling formalism for the protocol converter design. Very recently, the concept of the Petri net model has been used. Behavioral and structural properties of communicating protocols can be exhibited using the Petri net model. Juanole and Faure (1989) have used Predicate Petri nets in an informal method for defining a converter that connects a local area network and a remote computer through ISDN. A communicating Petri net model was proposed by de Jong for the design of concurrent asynchronous modules (de Jong and Lin, 1994). Tao and Goossens (1992) have used high level Petri nets for protocol conversion. Their method synthesizes the final converter by initially synthesizing a preliminary service converter, which is then transformed into a protocol converter. Their method also states the conversion problem by using the model of stimuli/responses. They have proposed the conversion at the service user layer via the conversion of service primitive flow. In order to convert the synchronization messages in the way of conversion invariant, the service converter is transformed into a protocol converter by tracing along the path of the service primitives with the help of relationships between service specification and protocol specification. The tracing is done according to the protocol interface mapping between the service primitives and the operation of protocol entities. Their approach is not clear as to how the conversion function converts the messages in the resulting converter machine after applying the algorithm.

3. Converter properties and design criteria

Irrespective of the design approach used, there are three general properties that must be guaranteed in a converter design:

1. *The safety properties of the protocol converter*: A safety property ensures that the converter never enters an undesirable state, that is, something bad never happens (the converter is safe). These properties include freeness from deadlocks, freeness from livelocks, and completeness (that is, absence of unspecified reception errors).
2. *The liveness properties of the protocol converter*: A liveness property ensures that the converter will eventually enter a desirable state, that is, something good will eventually happen (the converter is live), meaning that the converter successfully performs its intended functions.
3. *The timeliness properties of the converter*: A timeliness propety ensures that the converter respects the timing and response time requirements of the two protocol and service specifications. In this case, we say that the converter is responsive, i.e., the converter executes its functions efficiently within the specified time constraints, and thereby provides an effective and responsive service to the interacting incompatible protocols.

The design of a protocol converter is generally based on certain criteria that play a key role in determining the quality and performance of the converter. These criteria include the following:

(a) *Modeling Formalism*: An appropriate formal description technique should be selected in the initial stages of the design of a communication protocol converter. This technique may be used to model the heterogeneous protocols and services, as well as the synthesized converter. Popular formal description techniques used are communicating finite state machines (CFSM), Petri nets and CCS-CSP-LOTOS based models.

(b) *Converter Design Approach*: As discussed in the latter part of Section 2, there are three different approaches, namely, the service level conversion approach, the protocol level conversion approach and the hybrid conversion approach.

(c) *Design Methodology*: Two design methodologies can be used, namely, the analytic and the synthetic methodology. The analytic design is based on a trial-and-error adhoc approach. A preliminary version of the converter is first obtained by defining both the messages and the process of message translation and transfer. The sequence of redesign, analysis, error detection and correction is applied iteratively until the protocol design becomes error-free. Therefore, this approach can be time-consuming. The synthetic design is based on a systematic, step-by-step construction of an error-free protocol converter. Safety, liveness and timeliness properties of the derived converter are taken into account at the initial stage of the converter synthesis procedure.

(d) *Information Transfer Issues*: The efficiency of information transfer between incompatible protocols also depends on whether the type of conversion is direct or indirect. Direct conversion involves no buffers; the messages produced in one protocol are converted and passed immediately to the other protocol. Real-life applications generally adopt indirect conversion. Here, the messages produced in one protocol are first stored in a non-FIFO buffer, re-ordered, and then converted and transmitted to the other protocol.

(e) *Synchronization Issues*: Translation and synchronization are two important aspects of a converter design. A converter may be designed to carry out the mapping of compatible messages between two given protocols. A mapping set can be used as a conversion specification to specify the relations of significant messages between the two protocols. The messages are represented as sets of traces. The purpose of trace synchronization is to shuffle, reorder and hold the matched messages to carry out the translation; in other words, synchronization ensures the availability of compatible messages during conversion. Synchronization also enables the resulting converter to perform concurrent processing, and consequently, work more efficiently.

(f) *Timeliness*: In an ideal protocol communication setup, two communicating networks interconnected by a perfect, non-lossy communication channel, can carry out a smooth, uninterrupted bi-directional flow of data transfer. But this may precisely not be the case in real-life applications. Data flow can be interrupted by message losses over faulty communication channels; data request messages may not reach the destination protocol at the desired time; delays can occur in the reception of data acknowledgments by the sender protocol from the receiver protocol. In such cases, a 'time' factor can be incorporated during the design of a protocol converter. Unexpected data losses over faulty communication channels cannot go undetected in such a design. Section 5 elaborates on the timeliness behavior of the converter design.

(g) *Concurrency issues*: An efficient converter may be expected to perform concurrent functions. Concurrency is an important issue in converter synthesis. Concurrency features incorporated into a converter design greatly reduces the conversion time complexity. To implement concurrent features in a converter design effectively, a right choice of the formal description technique must be made to represent the interacting machines. In general, concurrency enhances the overall converter performance.

(h) *Dynamicity*: A protocol converter, in general, is designed to function just as it is expected to – data received from one protocol entity is 'translated and transferred' to the other protocol entity by the converter. Such a converter may never get an authoritative role to play in the conversion process, and hence remains passive throughout the process. However, certain interoperability problems between two incompatible protocols can be solved by using a more active converter, that is, a *Dynamic Converter*. A dynamic converter can initiate an action during the conversion process. Section 5 highlights the dynamic nature of a converter.

(i) *Complexity*: The extent of complexity of a protocol converter design can vary depending on the design approach and the modeling formalism chosen. Common instances of increase in complexity is during the computation of the global reachability system or a complex cartesian product during the converter synthesis.

4. Model and definitions

In this section, we provide a formal definition of the labeled, timed Petri net model. We also introduce some useful operations and transformations on the model used in the converter synthesis (Berthomieu and Diaz, 1991; Merlin and Segall, 1976).

Definition 1. A labeled Timed Petri Net is a tuple denoted by: $\text{TPN} = (P, T, A, \text{PRE}, \text{POST}, T_p, T_t, M_0)$, where: (i) P is a non-empty set of places (represented by circles) such that $\text{card}(P) = p$, (ii) T is a non-empty set of transitions (represented by bars) such that $\text{card}(T) = t$, (iii) A is a non-empty set of action labels associated with transitions, (iv) PRE is a precondition incidence function, denoted by PRE: $P \times T \to N$, where N is the set of integers, and determines the existence of arcs emanating from places to transitions, (v) POST is a postcondition incidence function, denoted by POST: $P \times T \to N$, and determines the existence of arcs emanating from transitions to places, (vi) T_p is a $t \times 1$ vector, denoted by $T_p{:}T \to \{P, \text{true}\}$, and determines the predicate associated with each transition in the net, (vii) T_t is a $t \times 1$ vector, denoted by $T_t{:}T \to \{[t_{\text{MIN}}, t_{\text{MAX}}]\}$, where $[t_{\text{MIN}}, t_{\text{MAX}}]$ is the time interval associated with a transition in the net within which the transition must fire. t_{MIN} is the minimum time permitted for an enabled transition to wait before it fires, and t_{MAX} is the maximum time an enabled transition can wait before it fires. Finally, (viii) M_0 is a $p \times 1$ vector called the initial marking (state) of the net, and determines the initial distribution of tokens in the different places of the net, such that $M_0(p_i) = n$ means that place p_i initially has n tokens.

An action labeling a transition in a TPN is normally associated with an access or observation point (op) at which the action occurs. The action label can be either an input or output action label. Input (output) action labels are preceded by + (−).

Definition 2. Let s be a sequence of transitions $t_1, t_2, \ldots, t_k$. s is called a legal firing sequence starting from M if there exists a sequence of markings $M, \ldots, M_i$ such that: $M \to M_1 \to \ldots \to M_i$. $M \to M_1$ means that M_1 is reachable from M by firing transition t_1. Marking M_i is said to be reachable from M by firing s denoted by M-$*s \to M_i$.

Definition 3. For a Petri net PN, the set of traces is defined as $L(\text{PN}) = \{a_1, a_2, \ldots \mid \exists M'{:}\ (M_0, \langle a_1, a_2, \ldots \rangle, M') \in \text{RG(PN)}\}$, where RG(PN) denotes the reachability graph of PN.

Definition 4. A timed transition t is enabled and firable when the following pre-conditions are satisfied:
1. Each of its input place contains at least one token.
2. The predicate associated with the transition is satisfied.
3. t_{MIN} has elapsed since the two above conditions were met.

Definition 5. Given two Petri nets $\text{PN}_i = (A_i, P_i, T_i, M_{0i})$ for $i = \{1,2\}$ with $P_1 \cap P_2 = \Phi$, the parallel composition of the two nets PN_1 and PN_2 is defined as: $\text{PN}_1 \parallel \text{PN}_2 = (A_1 \cup A_2, P_1 \cup P_2, T', M_{01} \cup M_{02})$, where $T' = \{(I,a,O) \in T_1 \cup T_2 \mid a \notin A_1 \cap A_2\} \cup \{(I_1 \cup I_2, a, O_1 \cup O_2) \mid a \in A_1 \cap A_2 \text{ and } (I_i, a, O_i) \in T_i\}$. (A similar definition can be found in Glabeek and Vaandrager (1987).)

Definition 6. A trace t of a Petri net is a sequence of action labels (events) that may be observed during the execution of the net. Depending on the location of the observation points, these events can either be protocol messages or service primitives.

$\text{Pos}(e, t)$ is a function that returns the position of an action label e in trace t. Also, $|t|$ denotes the number of action labels in trace t.

Definition 7. The projection of a PN P over a set of observation points (sop), denoted $\Pi_{\text{sop}}P$, is a new net P' in which we replace the actions (transmission or reception) occurring at an observation point not belonging to sop by an ε event. Then, the net is compacted by removing ε-cycles and ε-transitions to obtain P'. Similarly, the projection of a trace t over a sop, denoted $t' = \Pi_{\text{sop}}t$, is a subtrace of t in which only the actions that could be observed at the specified observation points are included while preserving their order of occurrence.

Definition 8. The complement of a trace t, denoted by $\sim t$, is a trace in which each input (output) action label in t is transformed to an output (input) action label in $\sim t$.

Definition 9. Two communication traces t_1 and t_2 are said to be *mutually compatible* if:

1. $\forall\ -e$ in t_1, $\exists\ +e$ in t_2 / $\mathrm{Pos}(+e, t_2) = \mathrm{Pos}(-e, t_1)$, and $\forall\ +e$ in t_2, $\exists\ -e$ in t_1 / $\mathrm{Pos}(-e, t_1) = \mathrm{Pos}(+e, t_2)$.
2. $\forall\ -e$ in t_2, $\exists\ +e$ in t_1 / $\mathrm{Pos}(+e, t_1) = \mathrm{Pos}(-e, t_2)$, and $\forall\ +e$ in t_1, $\exists\ -e$ in t_2 / $\mathrm{Pos}(-e, t_2) = \mathrm{Pos}(+e, t_1)$.
3. If $-e_i$ and $-e_j$ in t_1 and $\mathrm{Pos}(-e_i, t_1) > \mathrm{Pos}(-e_j, t_1)$ then $\mathrm{Pos}(+e_i, t_2) > \mathrm{Pos}(+e_j, t_2)$.

Definition 10. The synchronization or schuffling of two compatible traces t_1 and t_2 (i.e., $t = t_1 \otimes t_2$) produces the set of traces satisfying the following conditions:

1. $|t| = |t_1| + |t_2|$, and
2. the reception of a message ($+m$) in t_1 is preceded by its transmission ($-m$) in t_2 or vice versa.

If t_1 is a trace from PN P_1 and t_2 is a trace from PN P_2, then $t = t_1 \otimes t_2$ contains all the traces from $C = P_1 \parallel P_2$ (i.e., $\otimes$ for traces is analogous to $\parallel$ for PNs).

Definition 11. A PN reaches a deadlock state if, during the execution of the net, a state (or marking) is reached from which no transition is – or will become – fireable in the entire PN.

Definition 12. A PN contains a livelock if it reaches a state (or marking) from which it cycles in a closed set of markings with no possibility of exiting the cycle. This is a dynamic deadlock in which the net is not doing any thing except progressing inside this cycle.

Definition 13. A PN contains a liveloop if it enters a cycle of transitions after which a transition allowing a way out of the cycle is fired.

Definition 14. A place invariant is a subset of the places of the net in which the total number of tokens remains constant at any time during the execution of the net.

Definition 15. A transition invariant is a sequence of transition firings starting from a marking M that will return the net to the same marking M.

Definition 16. A Petri net is said to be safe if it is free from deadlocks and livelocks and satisfies its predefined invariants.

Definition 17. A Petri net is said to be semantically correct if it satisfies its predefined liveness properties and invariants. Such a PN is live (i.e. something good eventually happens).

Definition 18. A Petri net is said to be syntactically correct if it is free from deadlocks and livelocks, i.e., it satisfies its predefined safety properties.

The main advantage of using the Petri net model to construct a converter is that its correctness can be verified easily using structural and enumerative analysis techniques (Berthomieu and Diaz, 1991). Also, Petri nets allow concurrency, a feature highly desirable for real-time applications.

5. Dynamicity and timeliness of the converter

Intuitively, a converter that allows two protocols to interoperate is a communication system that merely performs a ‘blind’ end-to-end translation and transfer of data between the incompatible protocols. However, converters may be required to play a more active role in order to solve the interoperability problem. An efficient converter design should possess two important features: (i) dynamicity, meaning that it may play an active role in the communication and (ii) timeliness, meaning that the converter must perform within well-defined timing constraints required by the two protocols.

A dynamic converter is necessary in certain real-life conversion applications since some interoperability problems can only be solved using an active converter. A dynamic converter may initiate an action during the conversion process in addition to reacting to input stimuli incoming from either end-protocols. This action can either be a control message or a protocol message. In certain situations, the converter may be required to continuously transmit these control/protocol messages; after each transmission, the converter waits for a response (i.e., ack) to arrive within a specified time interval from the receiving protocol. Arrival of the acknowledgment from the receiver protocol completes the first step of a successful data transfer, and in turn, establishes the converter’s twofold role as conversion initiator and facilitator.

Another important converter feature required for a reliable data transfer between two protocols is the ‘timeliness’ feature. Two types of timing constraints are to be determined when designing a converter: (i) internal timing parameters specific to the correct functioning of the converter itself and (ii) timing parameters to satisfy the timing requirements of both protocols. In this paper, we only deal with the first type of timing constraint. The internal timer of the converter may be represented as a range, its lower boundary being the minimum permitted time limit (t_{m}), and its upper boundary being the maximum permitted time limit (t_{M}). In most real applications, the internal timer of the converter is a retransmission timer. All internal timers of the converter are initialized to zero. The value of a timer is equal to the time elapsed since the last time it was reset. A coherent version of the expected message should arrive before the time t_M expires. Failure to receive this data within the prescribed time range is considered a 'time-out' (Tm).

A typical example of a setup where such constraint would be required is when the converter's internal timer expires after waiting for an acknowledgment message to arrive from the receiver protocol. Such a requirement can be met in the Petri net model by adding a timed transition in the converter. The converter can now re-transmit the same message and restart its retransmission timer. The associated time interval must be carefully designed and should reflect the physical network characteristics in which the converter operates.

A similar situation occurs when the converter waits for a specific message from the sender protocol, but fails to receive it within the specified time range. When the timer expires, the converter sends the sender protocol a 'time-out' signal, requesting for re-transmission of the expected data. Unexpected data losses over faulty communication channels can therefore be avoided in a timed Petri net design.

6. The converter synthesis technique

A protocol converter can be considered as a communication entity that must be syntactically and semantically correct, i.e., it must be deadlock-free, complete and live so that a smooth and transparent interoperability is guaranteed between two incompatible protocols. The technique used in this paper to synthesize a converter is hybrid, i.e., it uses both the service specifications and the protocol specifications.

In the following sub-sections, we first introduce some concepts and procedures that are needed for the construction of a communication converter. First, we introduce the concept of the greatest common service definition given two service definitions provided by the two network protocols. Then we discuss the issues of trace generation at various subsets of observation points and their synchronization. Finally, we introduce our synthesis procedure to obtain a Petri net-modeled converter specification given a set of synchronized traces. Proofs of correctness of the technique are provided in an appendix.

6.1. The greatest common service definition

To find a communication converter, the minimal and necessary requirement is that the communication services to reconciliate must not be disjoint. The greatest common service definition (gcsd) is the basis of most top-down converter synthesis methods. To obtain it, one must rely on the designer's deep understanding of the services offered by the two protocols. In this paper, we introduce a procedure similar to that introduced in Kristol et al. (1993) to compute the gcsd.

Suppose the service provided by N (M) is specified as a PN S_N (S_M). The input/output (IO) operations of S_N and S_M are the service primitives of N and M, respectively. Let the set of input operations of N_1 (M_1) with its local user or the upper layer be $IN_1(IM_1)$. Similarly, let the set of output operations of M_1 (N_2) with its local user or the upper layer be $OM_1(ON_2)$. The designer has to map the elements of ON_2 (OM_1) to IM_1 (IN_2). In most cases, this is a straightforward one-to-one mapping. However, the mapping may be more complex. For example, two service primitives in one service can be equivalent to only one primitive in the other service. Other service primitives of N may involve functions that are not offered by any combination of primitives in M. These functions will not be reconciliated and should never be invoked while using the greatest common subset of services. The mapping functions for SPs belonging to both systems are modeled by a PN called a service interface converter (SIC).

Next, we contract or prune both service definitions by removing from S_N and S_M all the transitions corresponding to the primitives that were not mapped in the SIC. The resulting pruned service PNs are S'_N and S'_M. The greatest common service definition or the global service can be computed by gcsd $= P_{\{\text{upper\_}N,\text{upper\_}M\}}(S'_N \parallel \text{SIC} \parallel S'_M)$ (Fig. 4), in which disjoint services of M and N are discarded.

6.2. Trace generation and collection

Two sets of traces of interest in our procedure correspond to the traces that can describe the events occurring at each of the networks separately and that contribute to the gcsd introduced in Section 6.1. Let T_N be the set of traces observed at upper_N and lower_N, and T_M be the set of traces observed at upper_M and lower_M. The set T_N (similarly for T_M) can be obtained by analytically computing $N'_1 \parallel$ us_N where N'_1 is a

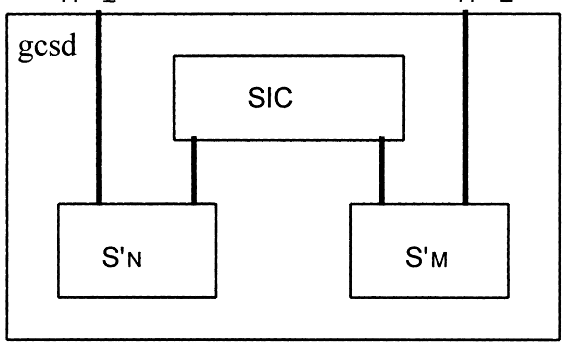


Fig. 4. The greatest common service definition.

pruning of N_1 that only shows the contribution of N'_1 to S'_N, and us_N is a composition of lower services and the bi-directional communication channel.

6.3. Trace synchronization

Given the two sets of traces T_N and T_M, obtained as described in Section 6.2, the corresponding sets of traces which include the input and output events (or protocol data units) observed at lower_N and lower_M are $\mathrm{TR}_N = P_{\{\mathrm{lower\_}N\}}\ T_N$ and $\mathrm{TR}_M = P_{\{\mathrm{lower\_}M\}}\ T_M$, respectively. The converter must be able to synchronize the complements of these two sets of traces. First, let the two partial sets of traces of the converter TC_N and TC_M be the complement of the two sets of traces in TR_N and TR_M, respectively. Then, each input event in a trace belonging to TC_N has to be matched with an output event in a trace belonging to TC_M, and vice versa. Schuffling the events in all the traces of the converter is called trace synchronization.

The communication traces in TC_N and TC_M are related to the patterns of the two types of communication services belonging to the gcsd. A communication service can be either confirmed or unconfirmed. In a confirmed service, there is a request, an indication, a response and a confirm service primitives and their related PDUs. A typical example of a confirmed service is a CONNECT service that requires prior mutual agreement to establish a connection between two communicating entities. However, in an unconfirmed service, there is only a request and an indication. Services such as the Data Transfer and DISCONNECT can be either confirmed or unconfirmed services, depending on whether or not the sender requires an acknowledgment. Consequently, the converter has to synchronize two sets containing traces corresponding to different types of the same service.

In the following, we list all possible patterns that can be observed at lower_M and lower_M and their schuffling. We refer to an event e sent (received) by the converter at lower_N as $N - e$ $(N + e)$. These patterns are also shown in Fig. 5.

Rules a to m list the possible traces of TC_N and TC_M and their synchronizeation. The trace synchronization or schuffling operator $\otimes$ is used between two traces: $t = t_1 \otimes t_2$, where $t_1 \in \mathrm{TC}_N$ and $t_2 \in \mathrm{TC}_M$.

Unconfirmed service in both networks N and M:
(a) $(N + m) \otimes (M - m) = (N + m,\ M - m)$

Confirmed service in M (with possible acknowledgment) and unconfirmed service in N:
(b) $(N + m) \otimes (M - m,\ M + \mathrm{confm})$
$= (N + m,\ M - m,\ M + \mathrm{confm})$ or
(c) $(N + m) \otimes (M - m,\ M + \mathrm{confm},\ M - \mathrm{ack})$
$= (N + m,\ M - m,\ M + \mathrm{confm},\ M - \mathrm{ack})$

Confirmed service in N (with possible acknowledgment) and unconfirmed service in M:
(d) $(N + m,\ N - \mathrm{confm}) \otimes (M - m)$
$= (N + m,\ M - m,\ N - \mathrm{confm})$ or
(e) $(N + m,\ N - \mathrm{confm},\ N + \mathrm{ack}) \otimes (M - m)$
$= (N + m,\ M - m,\ N - \mathrm{confm},\ N + \mathrm{ack})$

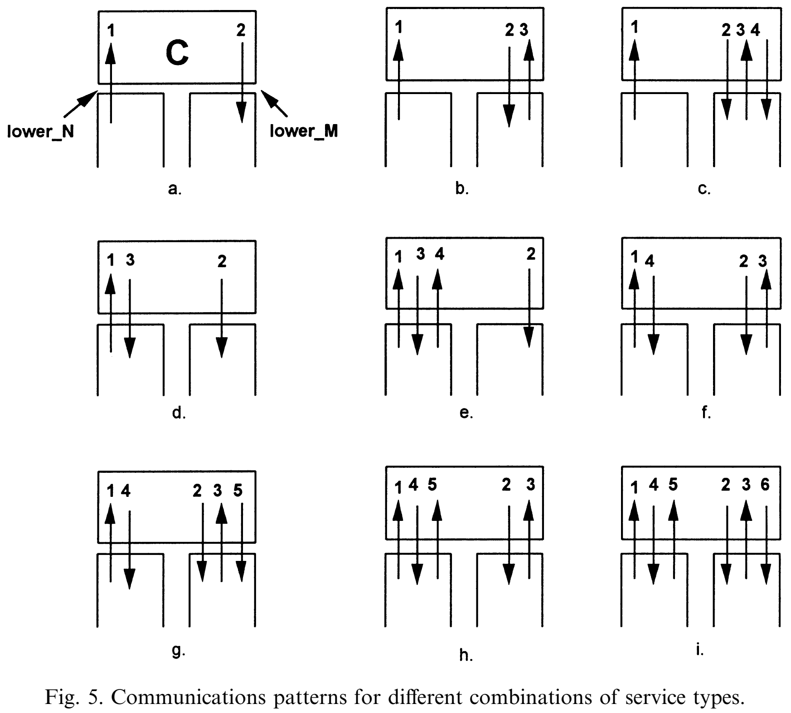


Fig. 5. Communications patterns for different combinations of service types.

Confirmed service in both networks N and M, with possible acknowledgments at either or both networks:

(f) $(N + m, N - \text{confm}) \otimes (M - m, M + \text{confm}) = (N + m, M - m, M + \text{confm}, N - \text{confm})$

(g) $(N + m, N - \text{confm}) \otimes (M - m, M + \text{confm}, M - \text{ack}) = (N + m, M - m, M + \text{confm}, N - \text{confm}, M - \text{ack})$

(h) $(N + m, N - \text{confm}, N + \text{ack}) \otimes (M - m, M + \text{confm}) = (N + m, M - m, M + \text{confm}, N - \text{confm}, N + \text{ack})$

(i) $(N + m, N - \text{confm}, N + \text{ack}) \otimes (M - m, M + \text{confm}, M - \text{ack}) = (N + m, M - m, M + \text{confm}, N - \text{confm}, N + \text{ack}, M - \text{ack})$

Confirmed service in N with negative acknowledgment or positive acknowledgment and unconfirmed or confirmed service(with possible acknowledgement) in M:

(j) $(N + m, N - \text{confm}, N + \text{ack}/(N + \text{nack}, N + m, N - \text{confm}, N + \text{ack})) \otimes (M - m) = (N + m, M - m, N - \text{confm}, N + \text{ack})/(N + m, N - \text{confm}, N + \text{nack}, N + m, M - m, N - \text{confm}, N + \text{ack})$

(k) $(N + m, N - \text{confm}, N + \text{ack}/(N + \text{nack}, N + m, N - \text{confm}, N + \text{ack})) \otimes (M - m, M + \text{confm}) = (N + m, M - m, M + \text{confm}, N - \text{confm}, N + \text{ack})/(N + m, N - \text{confm}, N + \text{nack}, N + m, M - m, M + \text{confm}, N - \text{confm}, N + \text{ack})$

(l) $(N + m, N - \text{confm}, N + \text{ack}/(N + \text{nack}, N + m, N - \text{confm}, N + \text{ack})) \otimes (M - m, M + \text{confm}, M - \text{ack}) = (N + m, M - m, M + \text{confm}, N - \text{confm}, N + \text{ack}, M - \text{ack})/ (N + m, N - \text{confm}, N + \text{nack}, N + m, M - m, M + \text{confm}, N - \text{confm}, N + \text{ack}, M - \text{ack})$

Concurrent message reception. Here, the messages at lower_N and lower_M either simultaneously or consecutively.

(m) $(N + m_1) \otimes (M + m_2) = (N + m_1, M + m_2)/(M + m_2, N + m_1)$

Rules a to m represent different possibilities of an event being received by the converter from either of the networks. Furthermore, we can list different possibilities where the converter may be required to play a more active role. A list of rules thus obtained would represent the dynamic behavior of the converter. As an example, we consider rule n. The service pattern may be as follows:

(n) $(N - m, N + \text{confm}) \otimes (M - m) = (N - m, N + \text{confm}, M - m)$

Here, the converter initiates the communication by transmitting message m to N, receives a confirmed response from N and delivers the 'converted' message to M. This pattern is shown in Fig. 6.

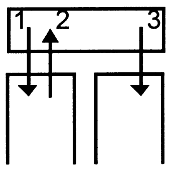


Fig. 6. Pattern n.

In addition to the above loopless traces, other traces containing repetitions of one or more events (or subtrace) can also be obtained. A repeated subtrace t is represented using the regular expression repetition operator * (i.e., $(t)^*$) in the generated traces or their complemented sets TC_N and TC_M. t may include one of the following types of patterns:

(i) the transmission of one or more messages, i.e., $(N - m_1, N - m_2, \ldots, N - m_n)$;

(ii) the reception of one or more messages, i.e., $(N + m_1, N + m_2, \ldots, N + m_n)$;

(iii) the reception and transmission of one or more messages, i.e., $(N + m_1, N - m_2, \ldots,)$;

The following rules apply for the synchronization of traces containing repeated subtraces. Let t_1 and t_2 be traces in TC_N and TC_M, respectively.

Rule 1: If t_1 (or t_2) contains a repeated subtrace, the repetition operator is eliminated and the resulting trace will be synchronized with t_2 (or t_1) using one of the rules a to m for loopless traces.

Rule 2: If both t_1 and t_2 contain loops, then we first eliminate the repetition of the loops and the resulting traces will be synchronized ($t = t_1 \otimes t_2$) using one of the rules a to m for loopless traces.

Rule 3: After the application of either Rule 1 or 2, the following should be applied: if the (eliminated) repeated subtrace is of type (i) or (ii), and appears between messages x and y, then insert the loop before y in trace t. However, if it is of type (iii), then insert the loop before x in trace t.

6.4. Synthesis of the Petri Net model of the converter

Given a set of communication traces (CT), obtained by schuffling traces of TC_N and TC_M, it is possible to construct a minimal PN which represents the expected behavior of the converter. In our methodology, we compute the composition $N'_2 \parallel M'_1$, where N'_2 and M'_1 are pruned, and they only show the contribution of N_2 to S'_N and M_1 to S'_M, respectively. Then we prune it by deleting the traces which do not belongs to the set of communication traces (CT), obtained as described in Section 6.3, to obtain a final converter.

6.5. Outline of the procedure

The converter synthesis procedure proceeds as follows.

Step 1.

Obtain the gcsd PN (Section 6.1).

If a gcsd is empty or cannot be found, then terminate the procedure, else generate the set of possible global service traces T_S from the gcsd.

Step 2.
Obtain the set of possible traces, T_N and T_M observed at upper_*N*, lower_*N* and upper_*M*, lower_*M*, respectively (Section 6.2).

Step 3.
Initialize T_C to ε.
For each trace t_s in T_S:
3.1. Obtain a subtrace t'_{sN} corresponding to the contribution of t_s to the upper service interface of
N, that is,
$t'_{sN} = \Pi_{\text{upper\_}N} t_s$.
Obtain the set of traces:
$\text{TR}_N = \{\text{tr}_N | (\text{tr}_N = \Pi_{\text{lower\_}N} t)$ AND
$(t \in T_N)$ AND $(\Pi_{\text{upper\_}N} t = t'_{sN})\}$
Let TR_N be the Complement of TR_N.
3.2. Obtain a subtrace t'_{sM} corresponding to the contribution of t_s to the upper service interface of
M, that is,
$t's_M = \Pi_{\text{upper\_}M} t_s$.
Obtain the set of traces:
$\text{TR}_M = \{ \text{tr}_M | (\text{tr}_M = \Pi_{\text{lower\_}M} t)$ AND
$(t \in T_M)$ AND $(\Pi_{\text{upper\_}M} t = t'_{sM})\}$
Let TR_M be the Complement of TR_M.
3.3. Synchronize the traces in TR_N and TR_M to obtain some traces corresponding to the converter behavior. Add those traces to the set T_C.
Endfor.

Step 4.
Synthesize the Petri net specification of the converter (as described in Section 6.4) from the synchronization traces T_C obtained in Step 3.3.

6.6. Complexity of the methodology

To analyze the complexity of our converter synthesis methodology, we consider the complexity of each of its steps. The following computations are performed:

Step 1.
(1) The SIC is computed manually, and its complexity depends on the set of common services.
(2) The two refined services S'_N and S'_M that exclude uncommon services.
(3) Compute $\text{gcsd} = \Pi_{\{\text{upper\_}N,\ \text{upper\_}M\}}\ S'_N \parallel \text{SIC} \parallel S'_M$, the complexity of which depends on the set of common services.

Step 2.
(4) Obtaining N'_1 and M'_2 as the pruning of N_1 and M_2, respectively.
(5) $\Pi_{\{\text{upper\_}N,\text{lower\_}N\}} = N'_1 \parallel \text{us\_}N$
(6) $\Pi_{\{\text{upper\_}M,\text{lower\_}M\}} = M'_2 \parallel \text{us\_}M$

Step 3.
(7) Schuffling traces, the complexity of which depends on the number of traces in either TC_N or TC_M.

Step 4.
(8) Compute $M = \Pi_{\{\text{lower\_}N,\text{lower\_}M\}N'_2 || M'_1}$
(9) Prune the machine *M*.

The computations involving projections or pruning are straightforward and their complexities are linearly related to the number of transitions (or events) in the PN (or traces) to prune (or project onto). The complexity of the composition of the pruned service definitions depends only on the complexity and size of the gcsd, and not on the size of the original protocols.

7. Example

In this section, we present an example illustrating the application of our synthesis procedure. This example demonstrates the dynamic and timeliness behaviour of a protocol converter. The example considers the conversion between two simple protocols, namely, the Alternating Bit Protocol (ABP) and the POLL-END (PE) protocol.

Fig. 7 shows the temporal sequence of possible internal events in the data transfer phase of the Alternating Bit Protocol. The operation of the AB protocol is described as follows. The sender sends a 'DATA' message to the receiver and simultaneously enables its internal retransmission timer. Each DATA message sent by the sender to the receiver is accompanied by an additional bit '*b*'. The value of this control bit alters with each new DATA message. The receiver, on receiving the DATA, acknowledges it by sending an ACK message accompanied by the same control bit *b* that accompanied the DATA. If no acknowledgment reaches the

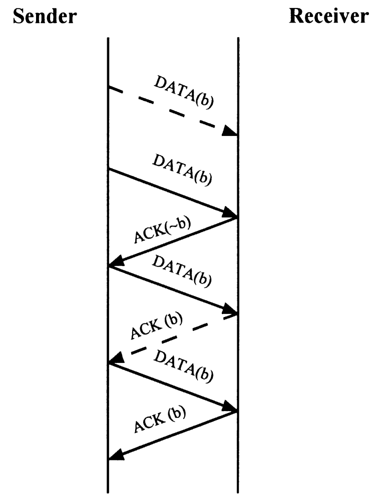
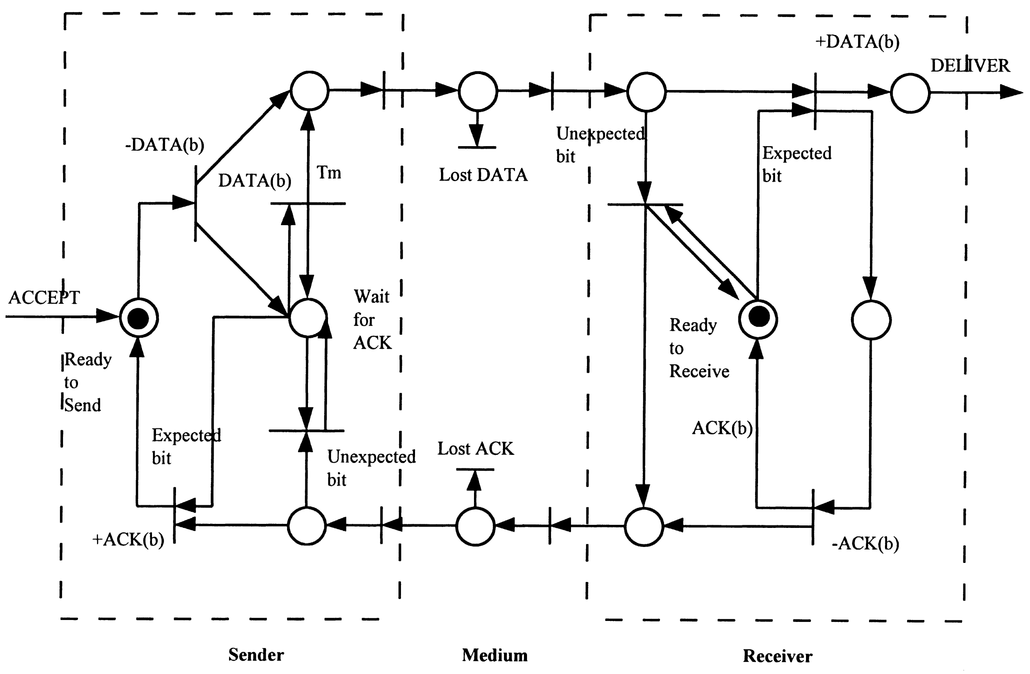
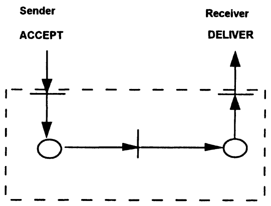


Fig. 7. Temporal sequence of possible internal events in the ABP.



(a). Alternating Bit Protocol



(b). Service interface specification for the Alternating Bit Protocol

Fig. 8. AB Protocol and its Service Definition.

sender before the timer expires, a copy of the message is sent after each timeout repeatedly, until its reception is acknowledged.

Fig. 8 shows the Petri net model of the Alternating Bit Protocol and its service definition. Fig. 9 shows the temporal sequence of possible internal events in the POLL-END protocol. The POLL-END protocol is described as follows. The messages POLL and END are used for flow control purposes. The receiver polls the sender requesting data, and simultaneously enables its internal retransmission timer. If no response is received from the sender before the timer expires (timeout), the timer is reinitialized and the ‘poll’ message is retransmitted. The sender, upon receiving the ‘poll’, sends ‘data’ to the receiver. Finally, to terminate data transfer, the sender sends an ‘end’ message. The receiver disables its timer on receiving a ‘data’ or an ‘end’ message. We assume that an end message will be sent eventually after finishing sending data messages. Fig. 10 shows the Petri net model of the POLL-END protocol, and its service definition.

The dynamic behavior of the converter is clearly portrayed the situation where the PE protocol is the sender and the ABP is the receiver. The converter initiates a poll command and delivers it to the sender PE, requesting data, if any, to be transferred to the ABP

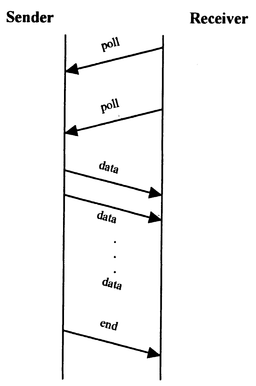
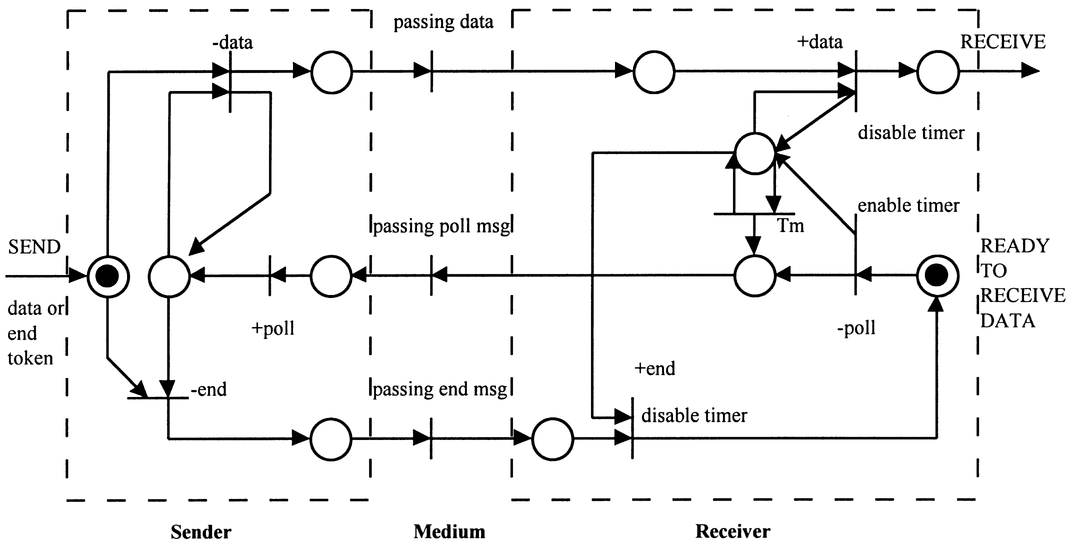
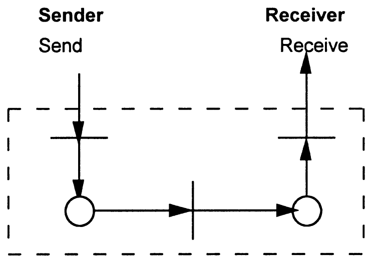


Fig. 9. Temporal sequence of possible internal events in the POLL-END protocol.



(a). POLL-END Protocol



(b). Service interface specification for the POLL-END Protocol

Fig. 10. POLL-END Protocol and its Service Definition.

protocol. The sender PE, upon receiving the poll request, considers it as a peer protocol entity request. The requested data is then passed on to the converter. Once the message relationships are established, the converter passes on the data received from the sender PE to the receiver ABP. The converter performs its conversion functions in a timely fashion, satisfying the timing constraints of both protocol architectures.

Using the procedure of Section 6.1, the greatest common service definition is shown in Fig. 11.The traces T_{ABP} (T_{PE}) obtained at LSAP and USAP of both sender and receiver of the Alternating Bit (POLL-END) protocol entities are:

$$T_{\text{ABP}} = \{\text{ACCEPT} - \text{DATA(bit)}(+\text{ACK}(\sim \text{bit}) - \text{DATA(bit)})^* + \text{ACK(bit)}, +\text{DATA(bit)} (-\text{ACK}(\sim \text{bit}) + \text{DATA(bit)})^*\text{DELIVER} - \text{ACK(bit)}, +\text{DATA(bit) DELIVER}(-\text{ACK} (\sim \text{bit}) + \text{DATA (bit)})^* - \text{ACK(bit)}, + \text{DATA(bit)}(+\text{DATA(bit)})^*\text{DELIVER} - \text{ACK(bit)}, +\text{DATA(bit) DELIVER} (+\text{DATA(bit)})^* - \text{ACK(bit)}\}$$

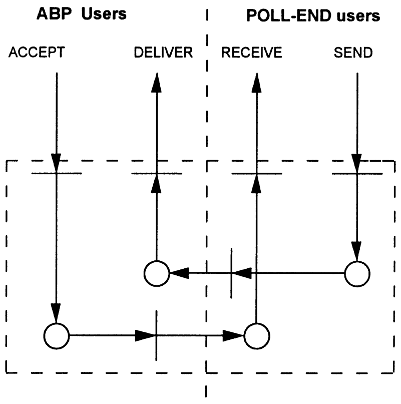


Fig. 11. gcsd of ABP and POLL-END Protocols.

$$T_{\text{PE}} = \{\text{SEND} + \text{poll}(-\text{data SEND})(-\text{data SEND})^* - \text{end}, +\text{poll SEND}(-\text{data SEND}) (-\text{data SEND})^* - \text{end}, (+\text{poll})^*, - \text{poll}(+\text{data RECEIVE})(+\text{data RECEIVE})^* + \text{end}, (-\text{poll})^*\}$$

The sets of traces of the converter, TC_{ABP} and TC_{PE}, are:

$$TC_{ABP} = \{\text{lower\_ABP}\} \sim T_{ABP} = \{-\text{DATA(bit)}(+\text{ACK}(\sim\text{bit}) - \text{DATA(bit)})^* + \text{ACK(bit)}, + \text{DATA(bit)}(-\text{ACK}(\sim\text{bit}) + \text{DATA(bit)})^* - \text{ACK(bit)}, -\text{DATA(bit)}(-\text{DATA(bit)})^* + \text{ACK(bit)}\}$$

$$TC_{PE} = \{\text{lower\_PE}\} \sim T_{PE} = \{-\text{poll} + \text{data}(+\text{data})^* + \text{end}, (-\text{poll})*, +\text{poll} - \text{data}(-\text{data})^* - \text{end}, (+\text{poll})*\}$$

The Petri Nets corresponding to these traces are shown in Figs. 12 and 13. These traces are synchronized to obtain a set of communication traces CT (as described in Section 6.3.). In the figure, the notation following each action label of the traces denotes the protocol which the converter is communicating. *S* stands for the sender protocol and *R* stands for the receiver protocol.

The converter traces are:

$$T_C = \{(-\text{poll})^*, -\text{poll}\ (+\text{data} - \text{DATA(bit)}(+\text{ACK}(\sim\text{bit}) - \text{DATA(bit)})^* + \text{ACK(bit)})^* + \text{end}, - \text{poll}(+\text{data} - \text{DATA(bit)}(-\text{DATA(bit)})^* + \text{ACK(bit)})^* + \text{end}, (+\text{poll})^*, +\text{poll}(+\text{DATA(bit)}(-\text{ACK}(\sim\text{bit}) + \text{DATA(bit)})^* - \text{ACK(bit)}) - \text{data}(-\text{data})^*)^* - \text{end}, + \text{DATA} + \text{poll}(-\text{ACK}(\sim\text{bit}) + \text{DATA(bit)})^* - \text{ACK(bit)} - \text{data})(+\text{DATA}(b) - \text{ACK}(\sim\text{bit}) + \text{DATA(bit)})^* - \text{ACK(bit)} - \text{data})^* - \text{end}\}$$

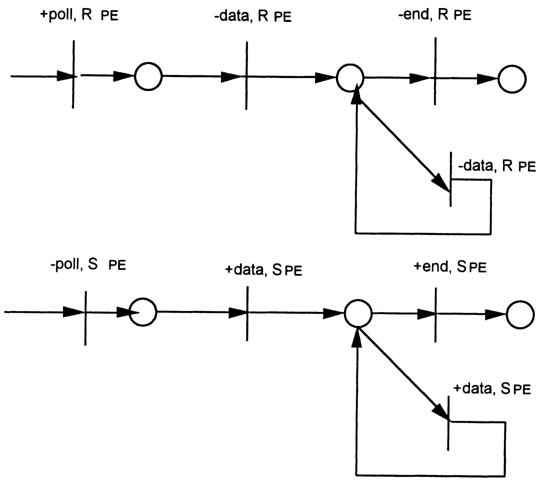


Fig. 12. Converter traces (TC_{PE}) related to the POLL-END Sender and Receiver.

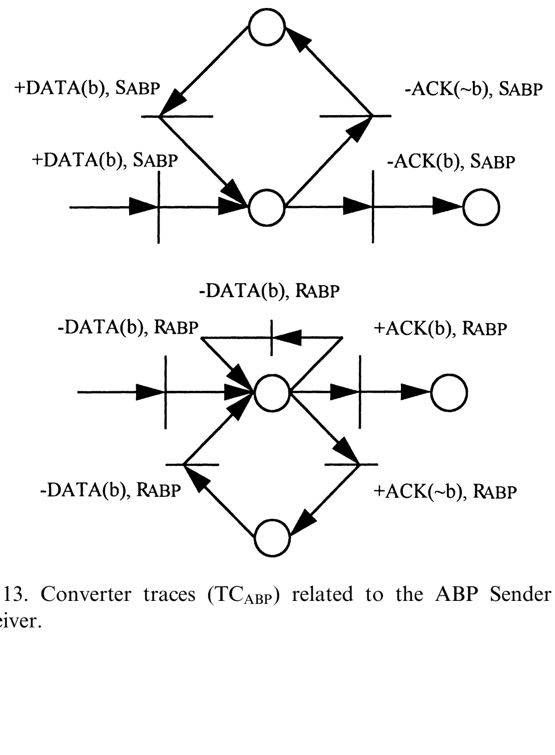


Fig. 13. Converter traces (TC_{ABP}) related to the ABP Sender and Receiver.

Fig. 14 shows the Petri Net converter synthesized from the set of communication traces CT (as described in Section 6.4.). The internal retransmission timer of the converter is enabled each time the converter sends a 'poll' message to the sender PE. The converter waits for a response till the time period expires (timeout).The timer is reinitialized, and the above process is repeated, i.e., multiple poll messages are sent, till a response is received. The timer is disabled as soon as a 'data' or an 'end' message is received.

8. Conclusions

In this paper, we have introduced a protocol converter synthesis procedure that ensures the interoperability between incompatible protocols providing related communication services. The procedure uses a blend of various techniques and operations to manipulate Petri net descriptions of the communication services and protocols for which a converter is needed. Our procedure follows the hybrid approach since it starts with the service definitions, and relies on the protocol messages and their relationships with service primitives, to synthesize the final converter. Our procedure integrates the most efficient components of other gateway design methods to efficiently synthesize a 'correct, safe and live' protocol converter. We have also demonstrated by means of two examples that the use of Petri nets as a modeling formalism allows the design and analysis of more efficient converters applicable in real-life applications, where concurrent communications services and protocols are required.

Acknowledgements

The authors thank the anonymous referees for their suggestions, and acknowledge the partial support of this work by Kuwait University Research Administration Grant EE 064.

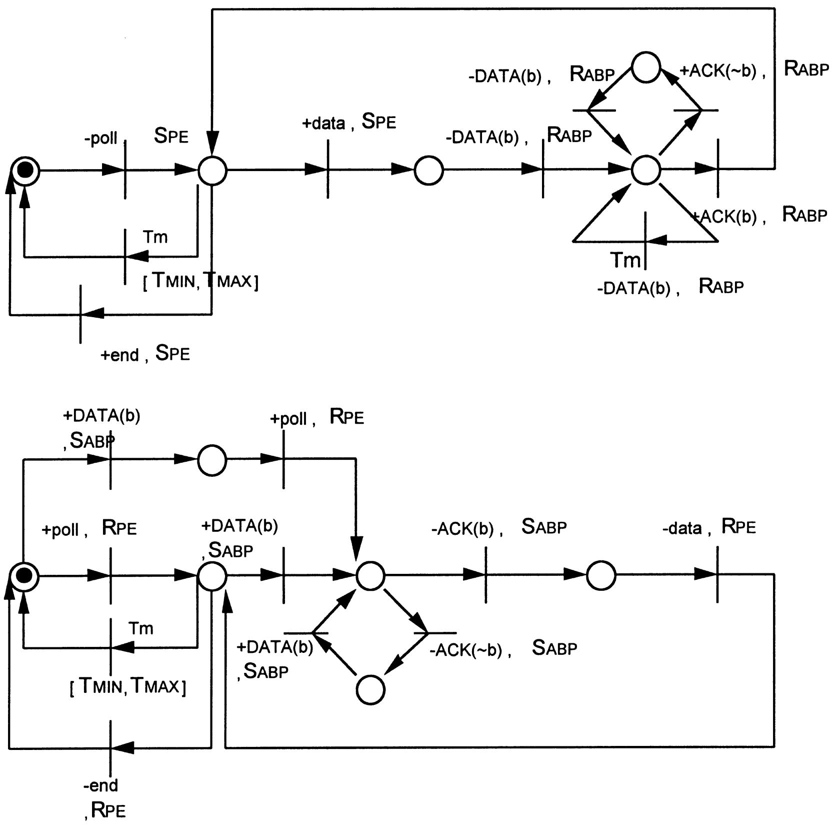


Fig. 14. The converter Petri Net.

Appendix A. Formal Proofs of Correctness

In this appendix, we present formal proofs to show that our proposed converter synthesis technique always constructs a converter which satisfies the safety, liveness and timeliness properties. Assuming that the two interacting incompatible networks are syntactically and semantically correct, if it is possible to compute their greatest common service definition, we can prove that a converter that is syntactically as well as semantically correct can be synthesized.

Lemma 1. *The converter is free from deadlocks.*

Proof. From the definitions of synchronization and parallel composition it is observed that synchronization for traces is analogous to the parallel composition for Petri nets. According to our synchronization rules, the pairs of traces considered are mutually compatible traces. After relaying each message, the converter reverts back to a state at which it is ready to synchronize the subsequent incoming messages transmitted by either network protocol interfaces. From the schuffling of traces as shown in Section 6.3, it can be seen that for any transition there exists a firing sequence which allows to fire this transition, thus it will never enters a situation when a transition cannot fire. As a result any transition will be eventually fireable. Moreover, we have assumed that the Petri net specifications of both network protocols satisfy the required invariants. The converter will therefore remain deadlock-free as long as the network protocols remain deadlock-free.

Lemma 2. *The converter is free from livelocks.*

Proof. We have initially assumed that the original network protocols are syntactically correct. The synchronization rules applied in our synthesis procedure ensure that no new loops are added to the converter if they do not belong to either or both livelock-free protocols. Moreover, we have assumed that the Petri net models of the two network protocols satisfy the transition invariant condition, that is, a sequence of transition firings starting from a marking will return the net to the same marking. Hence, after applying the synchronization rules, the converter Petri net is modeled in such a manner that there is no possibility of any sequence of transitions to infinitely progress within the net with no means of exiting the cycle. Thus no livelock would exist in the converter.

Lemma 3. *The converter satisfies its required invariants.*

Proof. According to our initial assumptions, the Petri net models of the two network protocols satisfy the required invariants – the place invariants and the transition invariants. The rules for synchronization in our synthesis procedure schuffle and reorder the mutually compatible traces of the two network protocols in such a way that the set of converter traces obtained after synchronization consists of all the traces derived from the parallel composition of the two network Petri nets. We have seen from the definition of parallel composition (Definition 5) and its corresponding example (Fig. 3) that when two nets are parallelly composed, the sequence of transition firings is not altered – before and after parallel composition, each sequence of transition firings returns to the same marking from which it started. This satisfies the transition invariant. Moreover, we observe that parallel composition does not alter the boundedness of the resulting net, that is, the total number of tokens in any subset of places of the Petri net remains constant at any time during the execution of the net. Thus, the place invariant is satisfied. Since the converter derived by our synthesis procedure is a parallelly composed product of the two network protocols, we can conclude that the converter satisfies its required invariants.

Theorem 1. *The converter is safe.*

Proof. A Petri net is safe if it is free from deadlocks and livelocks and satisfies its required invariants (Definition 16). Therefore, the results of the above lemmas imply that the converter is safe.

Theorem 2. *The converter is live.*

Proof. A Petri net is live if all its transitions are eventually fireable. Assuming that the Petri net models of the two networks are deadlock-free, Lemma 1 avoids all possibilities of a non-live transition. Also, according to Lemma 3, the converter satisfies the transition invariant. Therefore, there is at least one reachable marking for each transition is which it enabled and fireable. Moreover, a safe converter only acts as a message relay – it relays a message from one protocol to the other without performing any modification to the message. The converter is therefore live as well.

References

Berthomieu, B., Diaz, M., 1991. Modeling and verification of time dependent systems using time Petri nets. IEEE Trans. Software Eng. 17 (3), 259–272.

Bochmann, G.V., 1990. Deriving protocol converters for communications gateways. IEEE Trans. Commun. 38 (9), 1298–1300.

Bochmann, G.V., Mondain-Monval, P., 1990. Design principles for communication gateways. IEEE J. Selected Areas on Commun. 8 (1), 12–21.

Calvert, K., Lam, S., 1989. Deriving a protocol converter: a top down method. Proceedings of the ACM SIGCOMM'89, pp. 247–258.

Calvert, K., Lam, S., 1990a. Adaptors for protocol conversion. Proceedings of the INFOCOM'90 vol. 2, pp. 552–560.

Calvert, K., Lam, S., 1990b. Formal methods for protocol conversion. IEEE Trans. Selected Areas Commun. 8 (1), 127–148.

de Jong, G., Lin. B., 1994. A communicating petri net model for the design of concurrent asynchronous modules. EURO ASIC Symposium.

Glabeek R., Van, Vaandrager, F., 1987. Petri net modules for algebraic theories of concurrency. Proceedings of PARLE, Lecture Notes in Computer Science, vol. 259.

Green, P., 1986. Protocol conversion. IEEE Trans. Commun. 34 (3), 257–268.

Jeng, H.J., Liu, M.T., 1992. From service specification to protocol converter: a synchronization transition set approach. Proceedings of the 12th International Symposium on Protocol Specification, Testing and Verification.

Juanole, G., Faure, C., 1989. On gateway for internetworking through ISDN: architecture and formal modelling with petri nets. Proceedings of the ACM SIGCOMM'89, Austin, TX.

Kristol, D., et . 1993. A polynomial algorithm for gateway generation from formal specifications. IEEE/ACM Trans. on Networking 1 (2), 217–229.

Lam, S., 1988. Protocol conversion. IEEE Trans. on Software Eng. 13 (3), 352–363.

Liu, M.T., 1990. Protocol conversion. In: Proceedings of the International Computer Symposium, pp. 28–9.

Merlin, P.M., Segall, A., 1976. Recoverability of communication protocols – implications of a theoretical study. IEEE Transactions on Communications, 1036–1043.

Ohara, Y. et al., 1987. Protocol conversion method for heteregeneous systems interconnection in multi-profile environment. Seventh International Symposium on Protocol Specification, Testing and Verification.

Okumara, K., 1986. A formal protocol conversion method. Proceedings of the ACM SIGCOMM'86, pp. 30–37.

Okumara, K., 1990. Generation of proper adaptors and converters from a formal service specification. Proceedings of the IEEE INFOCOM'90, vol. 2, pp. 564–571.

Peyravian, M., Lea, C.T., 1993. Construction of protocol converters using formal methods. Computer Commun. 16 (4), 215–228.

Rajagopal, M., Miller, R.E., 1991. Synthesizing a protocol converter from executable protocol traces. IEEE Trans. Computers 40 (4), 487–499.

Saleh, K., Ural, H., 1994. Formal specification of an information gateway service interface in Estelle. Comput. Standards Interfaces 16 (44).

Saleh, K., Jaragh, M., Rafiq, O., 1995. A methodology for the synthesis of communication gateways for network interoperability. Computer Standards and Interfaces 17 (2), 193–207.

Stallings, W., Data and Computer Communications. Macmillan, New York, pp. 568–607.

Tao, Z.P., Goossens, M., 1992. Synthesizing communication protocol converter: a model and method. ACM Computer Science Conference, pp. 17–24.

Tillman, M., Yen, D., 1990. SNA and OSI: three strategies for interconnection. Commun. ACM 33 (2), 214–224.

Kassem Saleh obtained his B.Sc., M.Sc. and Ph.D. from the University of Ottawa in Canada in 1985, 1986 and 1991, respectively. He worked for Bell Canada from 1985 to 1991 and at Concordia University for one year before joining Kuwait University in 1992. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, College of Engineering and Petroleum at Kuwait University. Dr. Saleh was placed in 8th position among the top scholars in the Field of Systems and Software Engineering in an annual assessment published by the Journal of Systems and Software in October 1997 and 1998. He was awarded the IBM telecommunications Software Scholarship in 1988, the George Franklin Prize for the best student paper in 1990 from the Canadian Interest Group on Open

Systems (CIGOS), the distinguished young researcher prize in 1994 and the distinguished teacher prize in 1996 both from the College of Engineering and Petroleum. His current research and teaching activities concentrates on software engineering, communications software, distributed systems and internet programming. Dr. Saleh has presented many tutorials at international conferences and universities worldwide.

Mansour H. Jaragh received his B.Sc. in electrical engineering from Tulane University, and the M.Sc. and Ph.D. with honors in Computer Engineering from New Mexico State University, in 1979 and 1983, respectively. He joined the Ministry of Communication in Kuwait in July 1975 and was in charge of testing the International Telephone Exchange (M.O.C.) in 1977. In 1983, he joined the department of electrical and computer engineering at Kuwait University where he is currently an associate professor. He is a member of the board of IEEE Region 8, Kuwait Section. His research interests include computer architecture, systolic array architecture and protocol engineering.