Dynamic Configuration Modelling of Communication Protocols Using Numerical Petri Nets

Ajin Jirachiefpattana

Department of Computer Science
Faculty of Science, Prince of Songkla University
Hat Yai, Songkhla 90112 THAILAND
Tel/Fax: (66 74) 212 921 Email: ajin@ratree.psu.ac.th

Abstract

Typically a real-life protocol specification is composed of two or more protocol entities communicating with each other by exchanging messages through service access points (SAPs). These protocol entities can be created and released dynamically, and their connections can change over time. These dynamic behaviours have been one of the serious problems for protocol designers to model and analyze.

Petri nets and high-level Petri nets have been used commonly and widely in protocol specification and verification. Traditionally protocol designers use a place in Petri nets to represent a state of a protocol. By following this tradition, it is impossible to model the dynamic configuration of communication protocols. To overcome this problem, tokens in Numerical Petri Nets (NPNs) are used to model states and also to model protocol entities and their connections. A sliding-window protocol is used to illustrate this approach.

Keywords: Dynamic configuration modelling, Communication protocols, Numerical Petri Nets

1. Introduction

User requirements need to be precise and unambiguous, and formal specification and modelling methods must be employed in the rigorous design of computer network protocols [1]. Techniques for modelling, specification and verification of computer network protocols have progressed significantly in the past decade. The progress is largely due to the development of Petri nets [2] and high-level Petri nets [3-5] for modelling and for automating the verification process.

Real-life protocol specifications typically consist of two or more protocol entities communicating with each other by exchanging messages or interactions through service access points (SAPs). These protocol entities can be created and released dynamically, and their connections can change over time. These dynamic behaviours have been one of the serious problems for protocol designers to model and analyze.

The Petri net formalism has proven to be effective in analyzing system behaviours since its invention in the early 1960s [6]. It is particularly useful for asynchronous and concurrent system analysis. Many extensions have been proposed to the Petri net formalism, such as Coloured Petri Nets (CP-nets) [4], Predicate/Transition Nets (Pr/T-nets) [3] and Numerical Petri Nets (NPNs) [5], and many analysis techniques and tools have been developed [7]. The main reason for the great success of these kinds of net models is the fact that they have a graphical representation and well-defined semantics allowing formal analysis [4]. Traditionally protocol designers use a place in Petri

nets to represent a state of a protocol. By following this tradition, it is impossible to model the dynamic configuration of communication protocols.

The aim of this paper is to present a technique to model the dynamic configuration of communication protocols using Numerical Petri Nets. Instead of using places to represent states, tokens in NPNs are used to model states and also to model protocol entities and their connections. In addition to the dynamic configuration, control flow and data flow of protocols are included in this model. Finally, a sliding-window protocol is used to illustrate the approach.

2. Numerical Petri Nets (NPNs)

Numerical Petri Nets (NPNs), developed originally by Symons [5], are one of the many extensions of Petri nets. An NPN can be specified algebraically or graphically. It has been found possible with NPNs to retain the basic principles, symbols and modes of operation of Petri nets, while adding a considerable amount of modelling convenience. NPNs, which are claimed to be a generalization of Petri nets, overcome various limitations of Petri nets in modelling many practical systems, and also provide a more compact form of graphical representation.

The graphical NPN consists of a bipartite directed graph, fixed firing rules and an initial marking. An NPN may have global variables (called P-variables) which transitions can read and write. An NPN has transition enabling conditions and operations which refer not only to the tokens in the input places but may also refer to the global variables. By marking an NPN we mean a complete specification of the tokens in each of its places and the value of all of its global variables.

An NPN, labelled G, can be defined by a quintuple(P,T,Fi,Fo,Mo), where

- "P" is the set of all places in the net G, where a place is represented by a circle in the NPN graph and can be used to represent a machine state;
- "T" is the set of all transitions in the net G, where a transition is represented by a bar in the NPN graph and can be used to represent an event; each transition includes its transition conditions and transition operations;
- "Fi" are the input firing rules, usually written on the arcs between the input places and their respective transitions; this specifies what tokens are required to be in the input places to enable the transition and what tokens are destroyed in (i.e. removed from) the input places when the transition is fired;
- "Fo" are the output firing rules, usually written on the arcs between the transitions and their respective output places; this specifies what tokens are created in (i.e. added to) the output places;
- "Mo" is the initial marking of the net; this specifies all the tokens in each place and global variables.

NPNs have been successfully used in modelling some ISO protocols and the standard Formal Description Technique Estelle [8-10].

3. NPNs Modelling Dynamic Configuration

The behaviour of a protocol entity in a protocol specification is typically described using an extended finite state machine (EFSM), which can then be modelled and analyzed using NPNs. Almost all protocol designers use places of NPNs to represent states of protocol entities, and few use global variables (P-variables) of NPNs to represent states [9]. However, by simply using places or global variables of NPNs to represent states of protocol entities, it is impossible to model dynamic creation and release of such protocol entities because in NPNs all places and transitions are static, and only tokens are created dynamically during the execution time of Petri nets. Thus, in

this paper, tokens in NPNs are used to model states and also to model protocol entities and their connections instead of using places.

To exemplify the approach, Figure 1(a) gives an EFSM transition in Estelle language [11] of a protocol entity, Figure 1(b) shows how NPNs can be used to model the transition by using places to represent states, and Figure 1(c) shows how tokens in NPNs can be used to hold states and interactions.

As shown in Figure 1(c), there are three NPN places: one representing the type of the protocol entity and the other two representing the interaction points Tcep and Ncep, and there are two types of tokens: one holding the state of the protocol entity and the other holding the interaction exchanged with the other protocol entity through the interaction point. Each token has the first attribute holding a protocol entity name. Thus, the transition condition (pE = pI) is used to identify that two tokens belong to the same protocol entity.

4. An Example - A Sliding-Window Protocol

Figure 2 depicts a conceptual model of a sliding-window protocol. We consider the interaction between two sliding-window protocol machines SWA and SWB capable of exchanging data frames via the unreliable transmission medium TA. As such, the sender transmits data frames to the transmission medium whenever the user has data to send, and the receiver collects data frames from the transmission medium and forwards them to its users. Acknowledgments are sent to the sender from the receiver under the condition: if the window size is N; when there are N unacknowledged data frames in the buffer, immediately send an acknowledgment back.

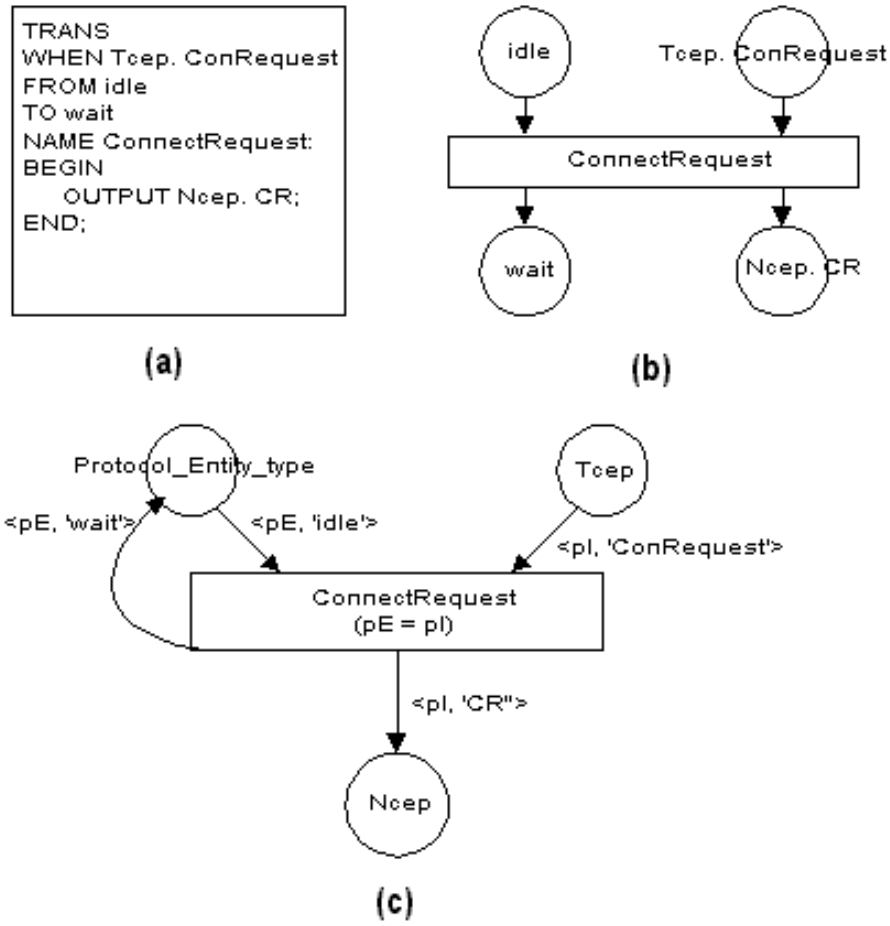

Figure 1: An example of using NPNs to model an EFSM transition

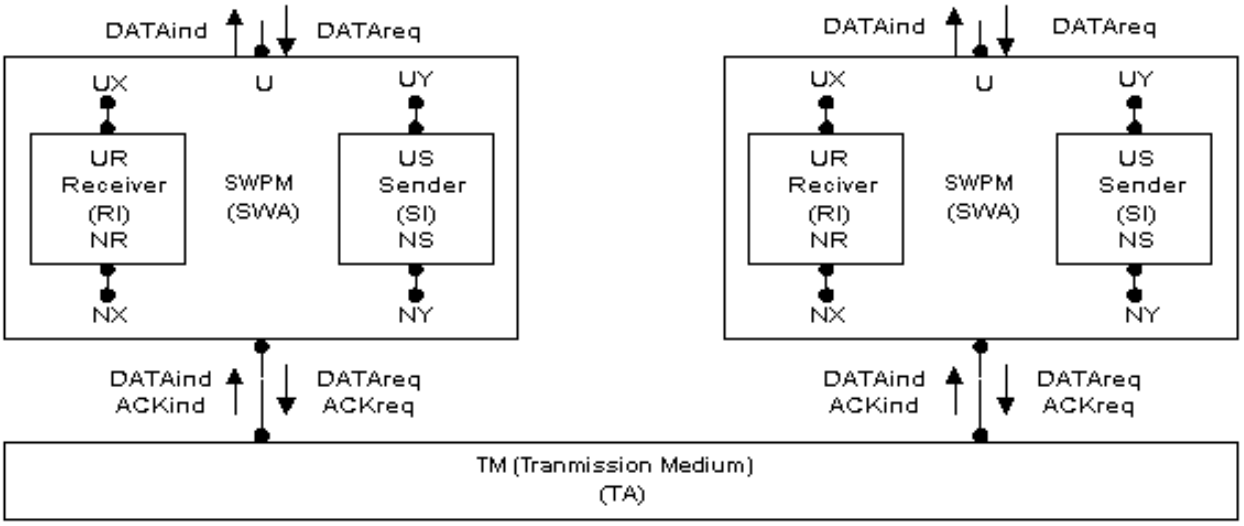

Figure 2: Architecture of a sliding-window protocol

Appendix A gives a specification of the sliding-window protocol machine (SWPM) described in Estelle, which is one of the Formal Description Techniques (FDTs) developed by ISO. For a more detailed description of Estelle, refer to [11]. The specification consists of two child modules (protocol entities): Receiver and Sender, two external interaction points U and N, and four internal interaction points UX, NX, UY and NY. The module Receiver has two external interaction points UR and NR, such that UR is connected to UX, and NR is connected to NX. The module Sender also has two external interaction points US and NS, such that US is connected to UY, and NS is connected to NY. These two child modules and their connections are established during the initialization of the parent module SWPM. The Estelle specification for the modules Receiver and Sender without priority clauses and delayed transitions was taken from [12].

Figure 3 and Figure 4 show an NPN model for the modules Receiver and Sender respectively. For the module Receiver, there are three NPN places: RB, NR and UR, and four transitions: t1, t2, t3 and initializeRB. For the module Sender, there are also three NPN places: SB, NS and US, and three transitions: s1, s2 and initializeSB. The transitions initializeRB and initializeSB are used to initialize the state and variables of the modules.

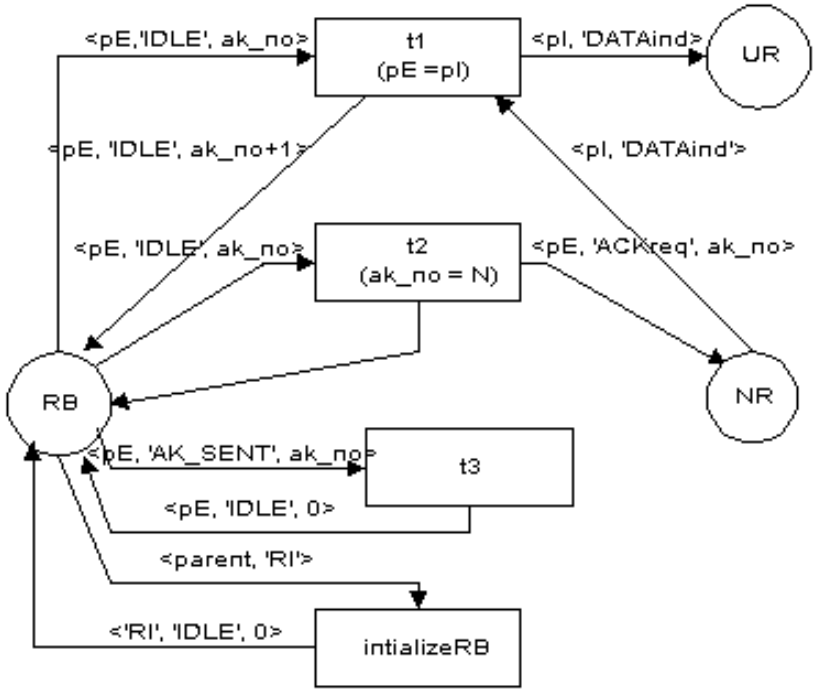

Figure 3: An NPN model for the module Receiver

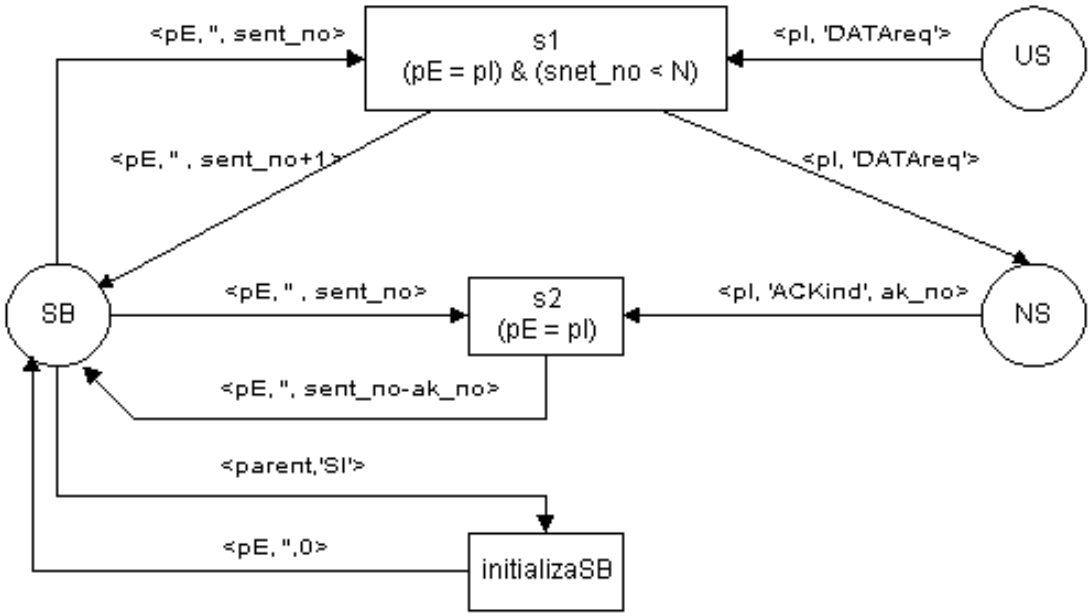

Figure 4: An NPN model for the module Sender

5. Conclusions

In this paper, we have described a technique to model communication protocols including dynamic configuration using Numerical Petri Nets (NPNs). This can be done by avoiding the traditional style of using places to represent states of protocol entities, and by using tokens to hold state, variables and interactions. The merits of this technique are that dynamic configuration can be modelled, the resultant NPN graph is very compact, and both data flow and control flow of the protocol system are combined in this model. Due to the limitations of the NPN formalism, time, priority and FIFO channels cannot be handled.

To demonstrate the viability of the technique, we have used the sliding-window protocol as an example. This technique has also been successfully applied to establish a relationship between Numerical Petri Nets and Estelle [10].

References

1. G. Bochmann and C.A. Sunshine, "Formal Methods in Communication Protocol Design", IEEE Transactions on Communication, C-28 (4), 1980, pp. 624-631.
2. J.L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1981.
3. H.J. Genrich, "Predicate/Transition nets", in Petri Nets: Central Models and Their Properties, W. Brauer, W. Reisig and W. Rozenberg (Eds.), Lecture Notes in Computer Science, Vol. 254, Springer-Verlag, 1987, pp. 207-247.
4. K. Jensen, Coloured Petri nets, EATCS monographs on theoretical computer science, Springer-Verlag, 1992.
5. F.J.W. Symons, Modeling and Analysis of Communication Protocols Using Numerical Petri Nets, PhD thesis, Dep. Elec. Eng. Sci., Univ. Essex, Telecommun. Syst. Group Rep. 152, May 1978.
6. J.P. Courtiat, J.M. Ayache and B. Algayres, "Petri nets are good for protocols", Computer Communica- tion Review, Vol. 14, No. 2, 1984, pp. 66-74.
7. F. Feldbrugge, "Petri Net Tools", Advances in Petri Nets 1985, G. Rozenberg (Ed.), LNCS 222, Springer-Verlag, 1985, pp. 203-223.
8. J. Billington, "Specification of the Transport Service using Numerical Petri Nets", C. Sunshine (Ed.), Protocol Specification, Testing and Verification II, North-Holland, IFIP, 1982.
9. R. Lai, T.S. Dillon and K.R. Parker, "Application of Numerical Petri Nets to Specify ISO FTAM Protocol", In Proceedings of the 1989 Singapore International Conference on Networks, Singapore, July 19-20, 1989.
10. A. Jirachiefpattana, Numerical Petri Nets Approach to Estelle Specification Verification, PhD thesis, School of Computer Science and Computer Engineering, La Trobe University, 1995.
11. ISO 9074, Information Processing Systems - Open Systems Interconnection - ESTELLE (Formal Description Technique based on an Extended State Transition Model), 1989.
12. S. Budkowski and P. Dembinski, "An Introduction to Estelle: A Specification Language for Distributed Systems", Computer Networks and ISDN Systems, Vol. 24, Elsevier Science Publishers B.V. (North-Holland), 1987, pp. 3-23.

Appendix A

An Estelle specification of the sliding-window protocol machine (SWPM)

```
module SWPM systemactivity ;
ip U : User_ Channel (provider) ;
N : Network_Channel (user) ;
end ;
body SWB for SWPM ;
ip UX : URecv_Channel (user) ;
UY : USend_Channel (user) ;
NX : NRecv_Channel (provider) ;
NY : NSend_Channel (provider) ;
module Receiver activity ;
output UR.DATAind ;
ak_no : = ak_no +1 ;
end ;
trans
from IDLE to AK_SENT
provided ak_no = N
name t2 :
begin
output NR.ACKreq (ak_no)
end ;
```

```
ip UR : URecv_Channel (provider) ;
NR : NRecv_Channel (user) ;
end ;
body RB for Receiver ;
const N = 7 ;
var ak_no : integer ;
state IDLE, AK_SENT ;
initialize to IDLE
begin
ak_no := 0
end ;
trans
from IDLE to IDLE
when NR.DATAind
name t1 :
begin
trans
when US.DATAreq
provided sent_no< N
name s1 :
begin
output NS.DATAreq ;
sent_no : sent_no + 1 ;
end ;
trans
when NS.ACKind
name s2 :
begin
sent_no := sent_no -ak_no ;
end ;
end ; {SB body }
modvar
RI : Receiver ;
SI : Sender ;
Initialize
begin
init RI with RB ;
init SI with SB ;
connect NX to RI.NR ;
connect UX to RI.UR ;
connect NY to SI.NS ;
connect UY to SI.US ;
end ;
trans
when U.DATAreq
trans
from AK_SENT to IDLE
name t3 :
begin
ak_no : = 0
end ;
end ; {RB body }
module Sender activity ;
ip US : USend_Channel (provide) ;
NS : NSend_Channel (user) ;
end ;
body SB for Sender ;
var sent_no : integer ;
initialize
begin
sent_no : = 0 ;
end ;
trans
when N.ACKind
name sw2 :
begin
output NY.ACKind ;
end ;
trans
when UX.DATAind
name sw3 :
begin
output U.DATAind ;
end ;
trans
when NX.ACKreq
name sw4 :
begin
output N.ACKreq ;
end ;
trans
when NY.DATAreq
name sw5 :
begin
output N.DATAreq ;
end ;
trans
when N.DATAind
name sw6 :
begin
```

```
name sw1 :
begin
output UY.DATAreq ;
end ;
```

```
output NX.DATAind ;
end ;
end ; { SWB body }
```

Webmaster Address: itrssm@au.ac.th