Towards Formal Modelling and Analysis of SCTP Connection Management

Somsak Vanit-Anunchai

School of Telecommunication Engineering
Institute of Engineering
Suranaree University of Technology
Muang, Nakhon Ratchasima, Thailand
Email: somsav@sut.ac.th

Abstract. The Stream Control Transmission Protocol (SCTP - RFC 2960) is a reliable unicast transport protocol originally specified by the Internet Engineering Task Force (IETF) in 2000. Its design rationale aims to overcome the weaknesses of the Transmission Control Protocol (TCP). However, after years of implementing and testing, defects and errors in RFC 2960 were reported and later fixed in RFC 4460. Incorporating those suggested fixes, IETF revised the SCTP specification and in September 2007 published RFC 4960, which replaces RFC 2960. This paper presents a Coloured Petri Net (CPN) model of the revised version of SCTP's connection management from RFC 4960. In particular we model the revised Tie-Tag mechanisms that differ significantly from those specified in RFC 2960. By following a procedure-based approach, each CPN page relates to only a few sections in RFC 4960, which aids validation. Preliminary results from state space analysis reveal two potential problems. First, SCTP association establishment can terminate in a half open state, with one side in CLOSED but the other in ESTABLISHED. Second, simultaneous establishment can lead to states in which both sides are in ESTABLISHED but the verification tags in both Transmission Control Blocks do not match.

Keywords: SCTP, Formal Methods, Coloured Petri Nets, State space methods.

1 Introduction

Motivation The Stream Control Transmission Protocol (SCTP) [6] is a transport protocol that was approved by the Internet Engineering Task Force (IETF) as Request for Comment (RFC) 2960 in 2000. It was originally designed by the Signalling Transport working group for transporting telephony signalling messages over UDP. These signalling messages have stringent timing requirements which are difficult to meet when using TCP. Foreseeing its significance and great potential to become a major transport protocol, IETF decided to operate SCTP over IP instead. Despite its potential to replace TCP, several years of implementation and testing revealed fifty-two defects in the SCTP specification, RFC 2960. Solutions were gathered and discussed in RFC 4460 [5]. The IETF has published a revised version of the SCTP specification, RFC4960 [4], in September 2007, and RFC 2960 has become obsolete. This revised specification raises two questions. Firstly, are there any unknown defects left? Secondly, are any new defects introduced in the new specification?

In addtion to the motivation for formal validation of SCTP, we wish to experiment with a new modelling approach called the "procedure-based" approach. In [2] Billington and Vanit-Anunchai discussed the advantages and

disadvantages of state-based and event-processing CPN modelling styles. The state-based modelling approach has the advantage of readability, and unspecified actions can be easily discovered. Its disadvantage comes from redundant specification of actions that are common to several states. While the event processing modeling approach has the advantage of folding common actions across several states, which makes the CPN model easier to maintain, it has some drawbacks with respect to readability. Thus [2] proposed the procedure-based modelling approach, which structures the CPN model according to the protocol's functionality. This modelling style has two merits. Firstly, the CPN model is easy to maintain. Secondly, the procedure-based CPN model comprises typical - simple procedures and unexpected - complex procedures (error handling). Beginners can pay attention to the typical scenarios before getting into the complex procedures later. The procedure-based CPN model of DCCP connection management presented in [2] evolved from a previous version modelled using the state-based approach. In this paper, we wish to gain experience building a procedure-based CPN model directly from an informal specification.

Previous work Since published in 2000, SCTP has been an attractive research topic. Although there has been a lot of work on SCTP regarding its security, performance, and extensions of SCTP functionality, we have found only one article (in Portuguese) [3] modelling SCTP connection management using Coloured Petri Nets (CPN). The CPN model we propose in this paper differs from [3] in three aspects. Firstly, we build the CPN model according the revised specification, RFC 4960, while [3] uses RFC 2960 which is now obsolete. Secondly, while the CPN model in [3] did not include the procedures for when SCTP nodes receive duplicated or unexpected messages, our model includes these events (described in Sections 5.2 and 9.2 in RFC 4960). Thirdly, the CPN model in [3] follows the state-based approach, whereas our model uses the procedure-based approach of [2].

Contribution The difficulty of designing a protocol is again witnessed by the defect list in RFC 4460 [5]. Despite many years of implementing and testing, it is still important to have a proper formal model and to perform formal analysis of SCTP connection management, especially when SCTP is designed for reliable data transfer such as signalling in Public Switching Telephone Networks. The contribution of this paper is three-fold. Firstly, we propose a CPN model of SCTP connection management based on Internet Standard RFC 4960. The model provides good insight into how to manipulate and use the Tie-Tags. Secondly, even though no errors in the shutdown scenarios are found, we discover an error in section 5.2.4 of RFC 4960 and two potential defects in the association establishment scenarios. Thirdly, for readers who are not interested in SCTP, this paper demonstrates an example of modelling a transport protocol using the procedure-based approach.

Organisation This paper is organised as follows. Section 2 provides an overview of SCTP association set up and graceful shutdown. Modelling assumptions are

listed in Section 3. The description of the CPN model of SCTP connection management and its declarations is given in Section 4. Section 5 presents the analysis results and a discussion of terminal markings. Section 6 presents conclusions and future work.

2 Stream Control Transmission Protocol

The Stream Control Transmission Protocol (SCTP) [6] is a unicast connection oriented transport protocol. Like TCP, SCTP provides an error-free reliable flow of data, without loss or duplication, between a client and a server. To ensure that the behaviour of SCTP's traffic mimics that of TCP, it uses the same congestion control algorithm as TCP.

SCTP has three distinctive features. Firstly, SCTP introduces the concept of multiple streams to reduce the problem of head-of-line blocking. It delivers its user messages in sequence within a given stream. Multiple streams are bundled into a single SCTP packet. The number of streams in an association is set up during startup. If one stream is blocked, delivery on the other streams can still proceed. Secondly, SCTP has the ability to support multiple IP addresses, called multi-homing. Several paths may exist between two nodes but only the primary path is used for transferring data. Other paths are redundant and are used when the network fails. Thus an SCTP connection is referred as an "association" between two sets of IP addresses. Thirdly, it defends against state-exhaustion attacks using a *cookie* mechanism and a four way handshake during connection establishment.

2.1 SCTP Packet Format

An SCTP packet comprises a common header and one or more chunks as shown in Fig. 1. The SCTP header contains 16 bit source and destination port numbers, a 32 bit verification tag and a 16 bit checksum. The verification tag is used to protect an association from blind attacks. Each end point keeps two values of verification tag: "My Verification Tag" and "Peer's Verification Tag". In general, any received packets containing a verification tag differing from "My Verification Tag" will be discarded. On the other hand, sent packets will carry a verification tag equal to "Peer's Verification Tag". These tag values are randomly selected at initialization and exchanged between the end points during association set up.

A Chunk is an information unit. There are 12 different control chunks but only one data chunk. The control chunks are Init[1], InitAck, SACK, Heartbeat, HeartbeatAck, Abort, Shutdown, ShutdownAck, Error, CookieEcho, CookieAck and ShutdownComplete. Control chunks are used to setup and shutdown the association, selectively acknowledge, report error messages, monitor reachability of the peer, etc. Association setup uses a four-way handshake comprising four control chunks: Init; InitAck; CookieEcho and CookieAck. Graceful closing uses

[1] Chunk names in the RFC are shown in all uppercase letters. To increase readability and distinguish them from SCTP States, the chunk names in this paper are given with only the first letters capitalized instead.

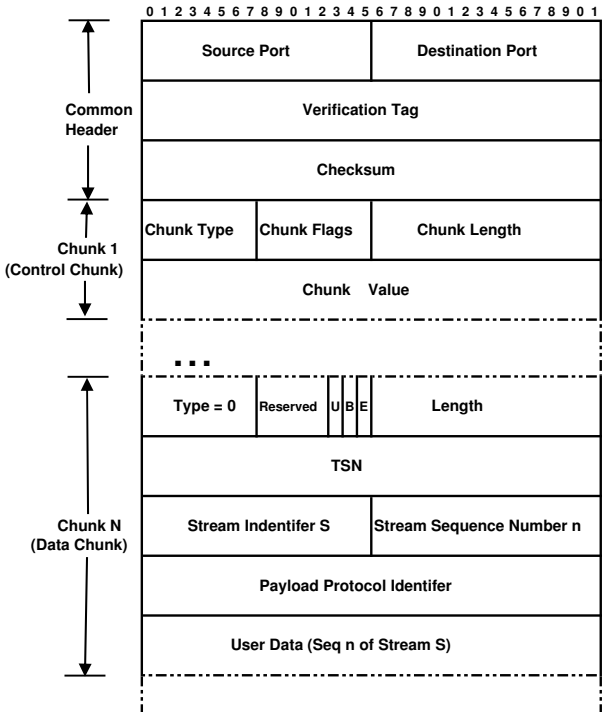

Fig. 1. SCTP Packet Format.

a three-way handshakes comprising three control chunks: the Shutdown; ShutdownAck and ShutdownComplete chunks. The Data transfer phase involves Data and SACK (Selective Acknowledgement) chunks. Further detail of the structure of chunks can be found in [4].

2.2 SCTP Connection Management Procedures

The state diagram shown in Fig. 2 illustrates the connection management procedures of SCTP. It comprises eight states: CLOSED; COOKIE-WAIT; COOKIE-ECHOED; ESTABLISHED; SHUTDOWN PENDING; SHUTDOWN-SENT; SHUTDOWN-RECEIVED and SHUTDOWN-ACK-SENT. The typical association establishment and close down procedures are shown in Fig. 3.

Normal Association Establishment Figure 2 (a) shows the association set up state diagram, and Fig. 3 (a) shows a typical set up procedure. An association between two nodes, A and Z, is initiated by a SCTP user on node "A" issuing an "ASSOCIATE" command. After receiving the "ASSOCIATE" primitive, node A sends an SCTP packet with a verification tag (VTAG) equal to zero. This SCTP packet contains only an Init chunk with an initial tag to specify the verification tag of incoming packets. Then node A enters the COOKIE-WAIT state. On receiving the Init chunk, node Z replies with an InitAck chunk indicating that it is willing to communicate with node A. The response includes node Z's initial tag number and encrypted cookie containing enough information to create node Z's Transmission Control Block (TCB). To prevent state

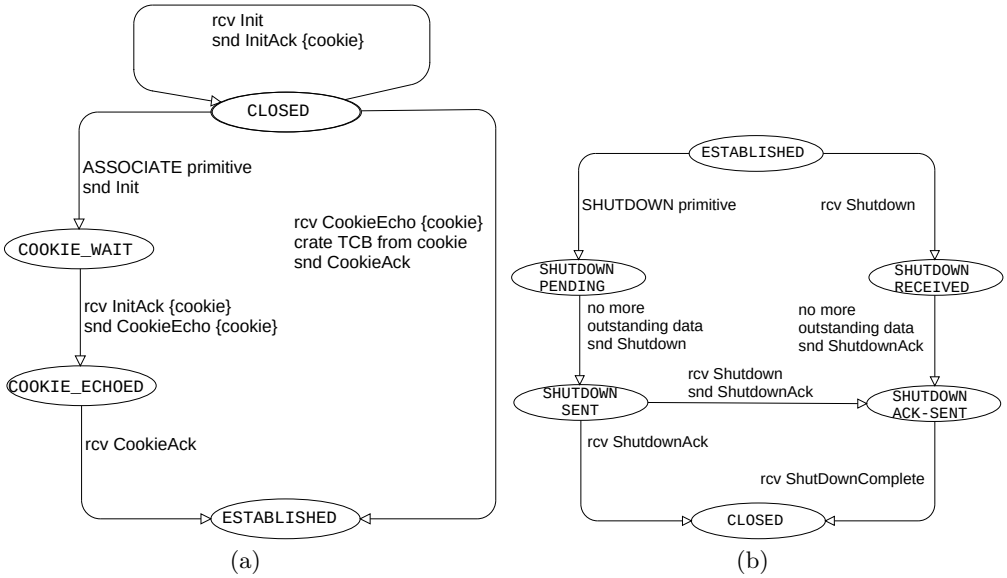

Fig. 2. SCTP State Diagram (a) association set up (b) closing down.

exhaustion attacks node Z is still in CLOSED after replying with an InitAck. To acknowledge the InitAck, node A returns the cookie in a CookieEcho chunk and enters COOKIE-ECHOED. When carrying an Init or InitAck chunk, the SCTP packet comprises only one chunk. When sending a CookieEcho chunk, the SCTP packet may enclose Data chunks after the CookieEcho chunk. On receiving a CookieEcho from node A, node Z creates its TCB from the received cookie, enters the ESTABLISHED state, replies with CookieAck and is ready for data transfer. After receiving CookieAck, node A enters ESTABLISHED indicating that the association is established. During data transfer, endpoint nodes A and Z may exchange Data and SACK chunks.

Graceful Association Shutdown Figure 2 (b) shows the association close down state diagram, and Fig. 3 (b) shows a typical graceful close down procedure. When the application at node A issues a "SHUTDOWN" command, node A enters the SHUTDOWN PENDING state and waits for all outstanding data chunks to be acknowledged. This end point stops accepting new data from the user. After all remaining data is acknowledged, node A sends a Shutdown chunk and enters the SHUTDOWN-SENT state. When node Z receives a Shutdown chunk, it must enter SHUTDOWN-RECEIVED, stop accepting new data from its user, and remain in this state until all outstanding data chunks are acknowledged. After all remaining data is acknowledged, node Z sends a ShutdownAck chunk and enters the SHUTDOWN-ACK-SENT state. After node A receives a ShutdownAck chunk, it must respond with a ShutdownComplete chunk and enter the CLOSED state. When node Z receives the ShutdownComplete chunk in SHUTDOWN-ACK-SENT, it enters the CLOSED state.

Besides the typical association set up and closing down procedures, the SCTP specification allows the simultaneous opening and simultaneous closing down of associations. For instance, when an endpoint in SHUTDOWN-SENT

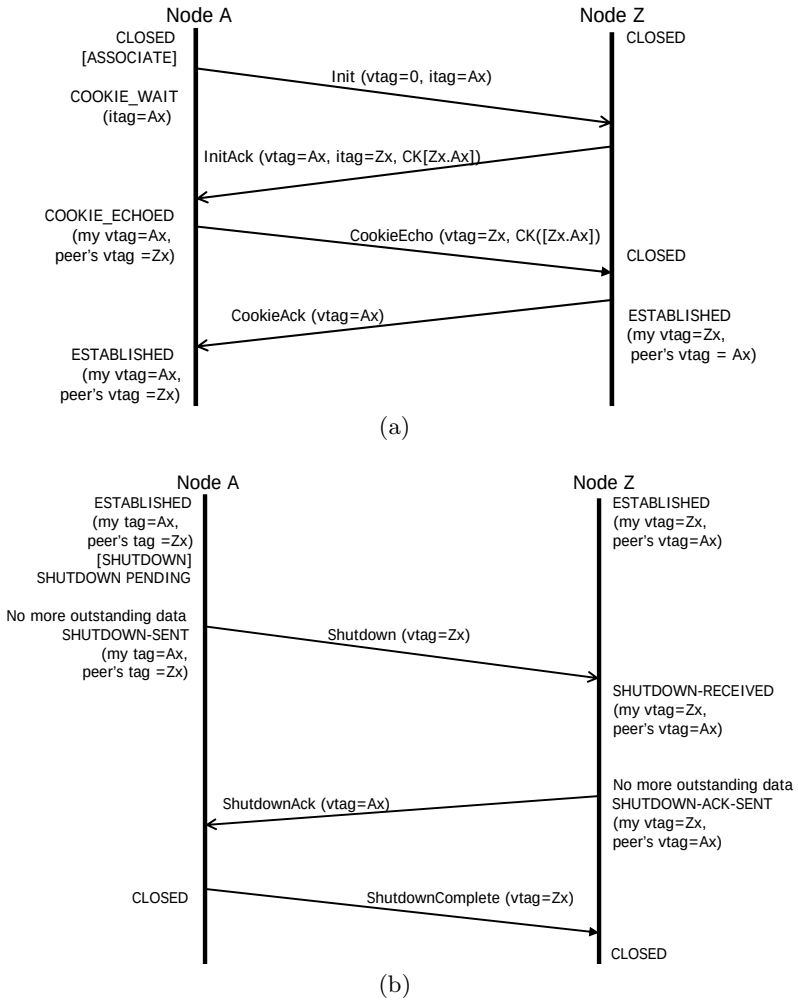

Fig. 3. Typical message sequence charts (a) association set up (b) closing down.

receives a Shutdown chunk, it sends a ShutdownAck in response and enters SHUTDOWN-ACK-SENT.

Handling Unexpected Init, InitAck, CookieEcho, and CookieAck Besides typical set up and closedown procedures, RFC 4960 specifies the rules to handle duplicate and unexpected Init, InitAck, CookieEcho, and CookieAck chunks in Section 5.2. These rules are intended to identify and solve problems that occur in the following scenarios.

1) An association is already established and both sides are in ESTABLISHED. An end point crashes and attempts to restore the association by sending a new Init chunk with a new Initial Tag.

2) Both end points attempt to open the association simultaneously.

3) The control chunk used to establish the association is stale.

4) An attacker generates a false SCTP packet.

5) The peer keeps retransmitting CookieEcho chunks and never receives a CookieAck.

Section 5.2 of RFC 4960 discusses the definition of *Tie-Tags*. *Tie-Tags* are copies of two verification tags (my verification tag and peer's verification tag). Actions specified in RFC 4960 that significantly differ from RFC 2960 are how to store and use the Tie-Tags. RFC 2960 specifies the Tie-Tags being stored in the cookie only but RFC 4960 requires to store the Tie-Tags in both cookie and TCB. The Tie-Tags in the TCB are called "Local Tag" and "Peer's Tag". The Tie-Tags in the cookie are called "Local Tie-Tag" and "Peer's Tie-Tag". The Tie-Tags are used to tie the received cookie of the new association with the old association. The Local Tie-Tag and the Peer's Tie-Tag (in the cookie) are compared with the Local Tag and the Peer's Tag (in the TCB) to ensure that the cookie belongs to the current association.

Section 5.2.4 of RFC 4960 discusses how SCTP responds when receiving an unexpected CookieEcho chunk. The Tie-Tags and verification tags in the cookie are compared with the verification tags in the existing TCB to identify which scenario occurs. Thus the received CookieEcho chunk can be correctly handled. An example of the scenarios is an association restart. When one side crashes and loses its existing TCB, Tie-Tags are used to link the restart association to the original association without shutting down and starting a new association.

3 Modelling Scope and Assumptions

Our model comprises all the state transitions of Fig. 2, and incorporates the narrative description from the RFC 4960 [4] Section 5.1, 5.2, 8.4, 8.5, 9.1 and 9.2. We also make the following assumptions regarding SCTP connection management when creating our CPN model.

1. We only consider a single association instance, while ignoring the procedures for data transfer, congestion control and other options. One SCTP packet contains only one chunk. A SCTP packet is modelled by chunk type, verification tag, initial tag and cookie. A cookie is modelled by "My Verification Tag" and "Peer's Verification Tag", "Local-Tie-Tag" and "Peer's-Tie-Tag".

2. Other fields in the SCTP packet are omitted because they do not affect the operation of the connection management procedure.

3. We do not consider misbehaviour or malicious attack.

4. Reordered or lossy channels may mask out possible deadlock, such as unspecified receptions. Thus we follow the incremental procedure outlined in [1] and analyse the CPN model with the following channel characteristics: FIFO without loss, reordered without loss, FIFO with loss, and reordered with loss, using the method proposed in [8]. However due to space limitations and for the sake of readability, we only discuss the case when the communication channels can delay and reorder packets without loss.

4 CPN Model of SCTP Connection Management

This section describes our CPN model of SCTP association establishment and shutdown procedures. Influenced by [2,7], the hierarchical structure of our CPN model is a procedure-based style. It comprises four hierarchical levels, 6 places,

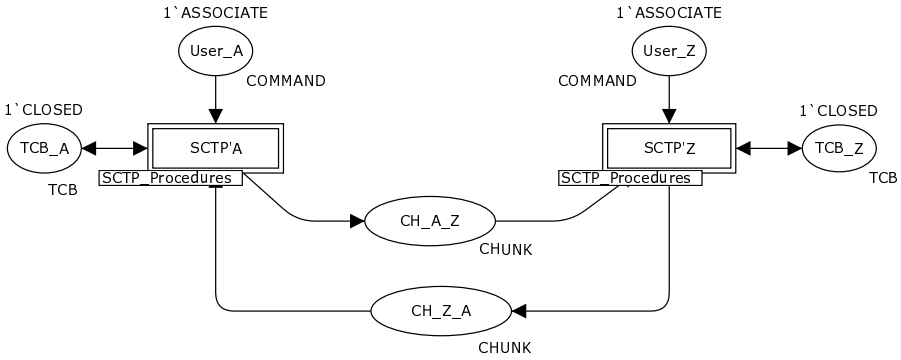

Fig. 4. The Top-level CPN page.

16 substitution transitions, 54 executable transitions and 2 ML functions. The top-level page of the SCTP-CPN model is illustrated in Fig. 4. Two substitution transitions (SCTP'A and SCTP'Z) represent the SCTP end point nodes, A and Z. Each side connects to four places. One place represents an application user typed by COMMAND. Another models a Transmission Control Block typed by TCB. Both end points are connected via two channel places, CH_A_Z and CH_Z_A. We assume that during association set up and closing down a packet contains only one chunk. Thus the channel places are typed by CHUNK, defined in Fig. 7. The layout of the top level CPN page also reflects the well-known model of the n-layer in a layered protocol architecture. The application layer is placed on the top while the underlying medium layer is below the protocol entity.

The substitution transitions, SCTP'A and SCTP'Z, are linked to the second level page named SCTP_Procedures, shown in Fig. 5. We divide SCTP_Procedures into five categories: normal events; unexpected events; retransmission; abort and checking Invalid Tags. Shown in Fig. 6 (a), the normal events comprise NormalEstablish and NormalShutDown of an association. The unexpected events are when the end points receive unexpected packets. We group the unexpected events into three CPN substitution transitions according to chunk types: UnexpectedIntIntAck; UnexpectedCookieEchoCookieAck and UnexpectedShutdown, shown in Fig. 6 (b). Space limitation prevents us from including all CPN model pages. Thus this paper will illustrate only five CPN pages: Normal'Establish, Normal'Shut_Down, Unexpected'Int_IntAck, Unexpected'CookieEcho_CookieAck and Unexpected'Shutdown.

With a state-based approach a CPN page represents several actions that may be scattered through the narrative specification. When modelling SCTP connection management with the procedure-based style, actions in each CPN page are confined to only a few sections in RFC 4960, as illustrated in Table 1. This makes our CPN model easier to understand when reading it alongside RFC 4960.

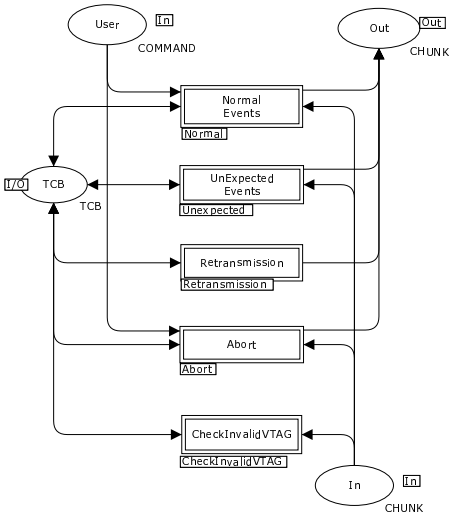

Fig. 5. The SCTP_Procedures page.

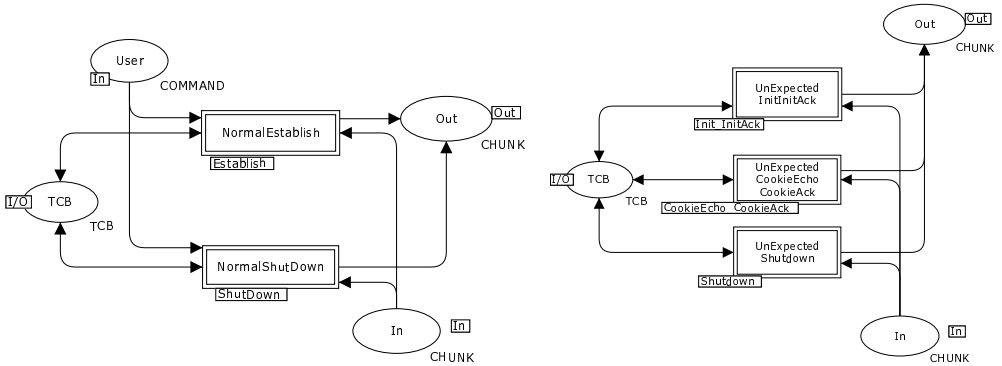

Fig. 6. (a) The Normal page (b) The Unexpected page.

4.1 Definition of CHUNK

When considering an abstract representation of SCTP packets, unlike the packets of TCP and DCCP, we found that Transmission Sequence Numbers (TSN) and Stream Sequence numbers are used only during the data transfer phase and are not relevant to SCTP connection management. On the other hand, Verification Tags (VTAG) play a major role during SCTP association establishment. Figure 7 defines the data structure of an SCTP packet called CHUNK (recall that we model each packet as a single chunk only). Many chunks have a different formats, thus we declare CHUNK (line 10) as the union of nine colour sets: VTAGxITAG (for Init chunk), VTAGxITAGxCOOKIE (for InitAck chunk), VTAGxCOOKIE (for CookieEcho chunk), four sets of VTAG (for CookieAck, Data, Shutdown and ShutdownAck chunks) and two sets of TFLGxVTAG (for Abort and ShutdownComplete chunks). Chunk types are distinguished by the ML selectors shown in line 10 to 14.

Procedures	CPN Page	Relevant Sections
Normal Event		
Normal establishment	Establish	5.1
Normal Shutdown	Shut_Down	9.2
Unexpected Events		
Receiving unexpected Init chunk	Init_InitAck	5.2.1, 5.2.2
Receiving unexpected InitAck chunk	Init_InitAck	5.2.3
Receiving unexpected CookieEcho chunk	CookieEcho_CookieAck	5.2.4
Receiving unexpected CookieAck chunk	CookieEcho_CookieAck	5.2.5
Receiving unexpected Shutdown chunk	Shutdown	9.2
Receiving unexpected ShutdownAck chunk	Shutdown	9.2
Receiving unexpected ShutdownComplete chunk	Shutdown	9.2
Abort	Abort	9.1
Retransmission	Retransmission	5.1, 9.2
Validation of VTAG	CheckInvalidVTAG	8.4, 8.5

Table 1. Relationship between SCTP procedures, CPN pages and sections in RFC 4960.

```
 1: colset VTAG = int;
 2: colset VTAGxITAG = product VTAG * VTAG; (* Init and InitAck Chunk *)
 3: colset MyVTAGxPeerVTAG = product VTAG * VTAG;
 4: colset COOKIE = record CK_TAG:MyVTAGxPeerVTAG
 5:                        * CK_TT:MyVTAGxPeerVTAG;
 6: colset VITAGxCOOKIE = record VI:VTAGxITAG * CK:COOKIE; (* InitAck chunk *)
 7: colset VTAGxCOOKIE = record VT:VTAG * CK:COOKIE; (* CookieEcho Chunk *)
 8: colset TFLAG = with T_ON | T_OFF;
 9: colset TFLAGxVTAG = product  TFLAG * VTAG;
10: colset CHUNK = union Init:VTAGxITAG + InitAck:VTAGxITAGxCOOKIE
11:                    + CookieEcho:VTAGxCOOKIE + CookieAck:VTAG
12:                    + Data:VTAG + Abort:TFLAGxVTAG
13:                    + Shutdown:VTAG + ShutdownAck:VTAG
14:                    + ShutdownComplete:TFLAGxVTAG;
```

Fig. 7. The definition of SCTP Chunk.

Line 1 defines the basic unit, VTAG, as the set of integers. Although an integer in CPN Tools is not a 32-bit unsigned integer like the VTAG field shown in Fig. 1, this does not affect the analysis of the model. Notice that according to our model abstraction, besides T-Flag, CHUNK comprises only VTAG and the compositions of VTAGs.

Each endpoint shall keep *my* and *peer's verification tags* in its TCB. When an endpoint sends out a packet, the value of *peer's verification tag* is copied into VTAG field of the outgoing chunks. If an endpoint receives an SCTP packet of which VTAG field does not match *my verification tags*, the packet shall be silently discarded. Peer's verification tags of both sides are initialized by exchanging the value in the Initial Tag (ITAG) field of the Init and InitAck chunks (see Fig. 3 (a)). Thus the data structure of the Init chunk is defined as the product of two verification tags (VTAGxITAG - line 2). The InitAck chunk (line 6) is modelled by a record of VTAGxITAG and COOKIE. Despite a cookie containing a lot of parameters, we model COOKIE (line 4) as a record of two pairs of VTAG: (*My Verification Tag*, *Peer's Verification Tag*) and (*Local-Tie-Tag*, *Peer-Tie-Tag*).

Line 7 defines CookieEcho chunk as the record of VTAG and COOKIE, whereas CookieAck, Shutdown and ShutdownAck are modelled by only VTAG.

Abort and ShutdownComplete chunks use T-Flag to indicate whether, when sending out these chunks, the TCB exists or not. If the TCB does exist, the Abort and ShutdownComplete chunks have T-Flag set to off and VTAG equal to the peer's verification tag. When TCB does not exist, T-flag is on and the VTAG field equals the verification tag of the received packet the SCTP entity is responding to. Thus Abort and ShutdownComplete chunks (line 9) are defined by the product of TFLAG and VTAG (TFLAGxVTAG).

4.2 Definition of TCB

During each stage of association establishment the TCB of a SCTP endpoint may record different data. CLOSED means there is no connection, and no state parameters exist. A SCTP node in the COOKIE-WAIT state has no knowledge of the *peer's verification tag* but knows only *my verification tag*. Both verification tags are known in the COOKIE-ECHOED state but a SCTP node may retransmit the CookieEcho chunk together with the echoed cookie. Thus the SCTP node needs to store the cookie for the purpose of retransmission because the content in the cookie is encrypted and cannot be recreated.

According to the differences in the TCB data structure we describe above, we divide SCTP states into four groups: CLOSED, COOKIE-WAIT, COOKIE-ECHOED and the states after association established. Figure 8 declares TCB in line 12 as a union set of empty set, COOKIEWAIT_CB,COOKIE_ECHOED_CB and SCTP_CB. SCTP's state are distinguished by ML selectors as defined in line 12. Similar to CHUNK, besides the retransmission counter, a TCB comprises only VTAG and the compositions of VTAGs.

```
 1: colset RCNT = int;
 2: colset COOKIEWAIT_CB = record Rcnt:RCNT * myvtag:VTAG
 3:                              * SV_TT:MyVTAGxPeerVTAG;
 4: colset SV_in_TCB = record Rcnt:RCNT
 5:                         * SV_VT:MyVTAGxPeerVTAG
 6:                         * SV_TT:MyVTAGxPeerVTAG;
 7: colset COOKIE_ECHOED_CB = product SV_in_TCB * COOKIE;
 8: colset TCB_EXIST_STATE = with ESTABLISH | SHUTDOWN_PENDING
 9:                             | SHUTDOWN_RECEIVED | SHUTDOWN_SENT
10:                             | SHUTDOWN_ACK_SENT;
11: colset SCTP_CB = product TCB_EXIST_STATE * SV_in_TCB;
12: colset TCB = union CLOSED + COOKIE_WAIT:COOKIEWAIT_CB
13:                 + COOKIE_ECHOED:COOKIE_ECHOED_CB
14:                 + TCBExist:SCTP_CB;
15: colset COMMAND = with ASSOCIATE | SHUTDOWN | ABORT;
```

Fig. 8. The definition of SCTP's Transmission Control Block (TCB).

Differing from RFC 2960, in addition to only storing Tie-Tags in the cookie, Section 5.2.2 of RFC 4960 defines the Local Tag and Peer's Tag be stored in association's TCB. Thus line 4 declares state variables in TCB, SV_in_TCB, as a record of retransmission counter (RCNT - line 1) and two products of verification tags: (my verification tag, peer's verification tag) and (Local Tag, Peer's Tag).

After association establishment, a SCTP node in our model has TCB defined as a product of state after the association established (TCB_EXIST_STATE- line 8) and state variables SV_in_TCB.

COOKIE_ECHOED_CB (Line 7) is defined as a product of SV_in_TCB and COOKIE (line 4 of Fig. 7). Line 2 declares COOKIE_WAIT_CB as a record of retransmission counter (RCNT), my verification tag (VTAG) and a pair of VTAGs (Local Tag, Peer's Tag). Line 15 defines user commands which are ASSOCIATE, SHUTDOWN and ABORT.

4.3 CPN Model of SCTP Normal Procedures

This subsection illustrates two CPN pages of typical scenarios of SCTP connection management: association establishment and graceful shutdown. Beginners who have just studied this protocol can easily understand these two CPN pages because they are similar to state diagrams (Fig. 2) and the typical message sequence chart in Fig. 3.

Normal Establishment Page The normal establishment procedures described in section 5.1 of RFC 4960 are modelled in the Normal'Establish page shown in Fig. 9. Five transitions represent a sequence of actions directly mapping from the message sequence chart in Fig. 3 (a). In addition to sending outgoing chunks and changing states, SCTP nodes validate VTAG fields and populates the values of Tie-Tags into the cookie and TCB. The guard on transition Rcv_Init requires the VTAG of the Init chunk to be equal to zero. According to last paragraph of section 5.2.2 page 67 of RFC 4960 [4], the value of Tie-Tags in CLOSED, COOKIE-WAIT and SHUTDOWN-ACK-SENT shall be set to zeros. Thus the Tie-Tags in the InitAck chunk and in TCB of COOK-WAIT state are set to zeros.

Normal Shutdown Page Figure 10 shows a CPN page of the normal graceful shutdown procedures described in section 9.2 of RFC 4960. Despite the normal shutdown procedure shown in Fig. 3 (b) comprising only SHUTDOWN primitives and three-way handshakes, the Normal'Shut_Down page contains 9 transitions. Unlike association establishment, section 9.2 of RFC 4960 does not separate the normal closing down and unexpected closing down procedures. Hence, in addition to the typical actions of SHUTDOWN primitives and three-way handshakes, this page includes

1) Clearing outstanding data when receiving Data chunks in SHUTDOWN-PENDING and SHUTDOWN-RECEIVED;

2) Retransmitting Shutdown chunks when receiving Data chunks in the SHUTDOWN-SENT state;

3) Receiving Shutdown chunks in the SHUTDOWN-RECEIVED state.

4.4 CPN Models of Unexpected Procedures

Handling unexpected receptions of SCTP control chunks is modelled by three CPN pages: Unexpected'Int_IntAck, Unexpected'CookieEcho_CookieAck, and Unexpected'Shutdown. These are illustrated in this section.

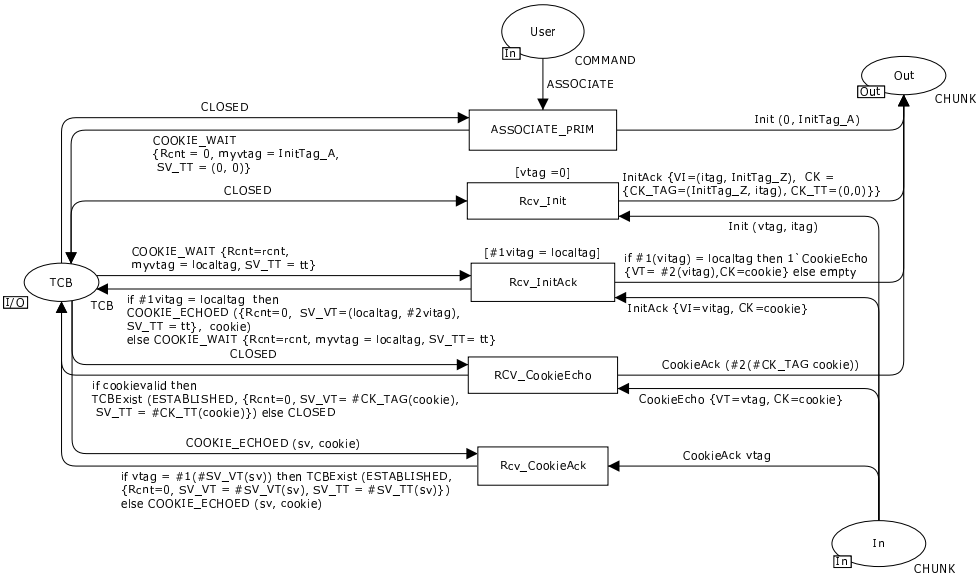

Fig. 9. The Normal'Establish page.

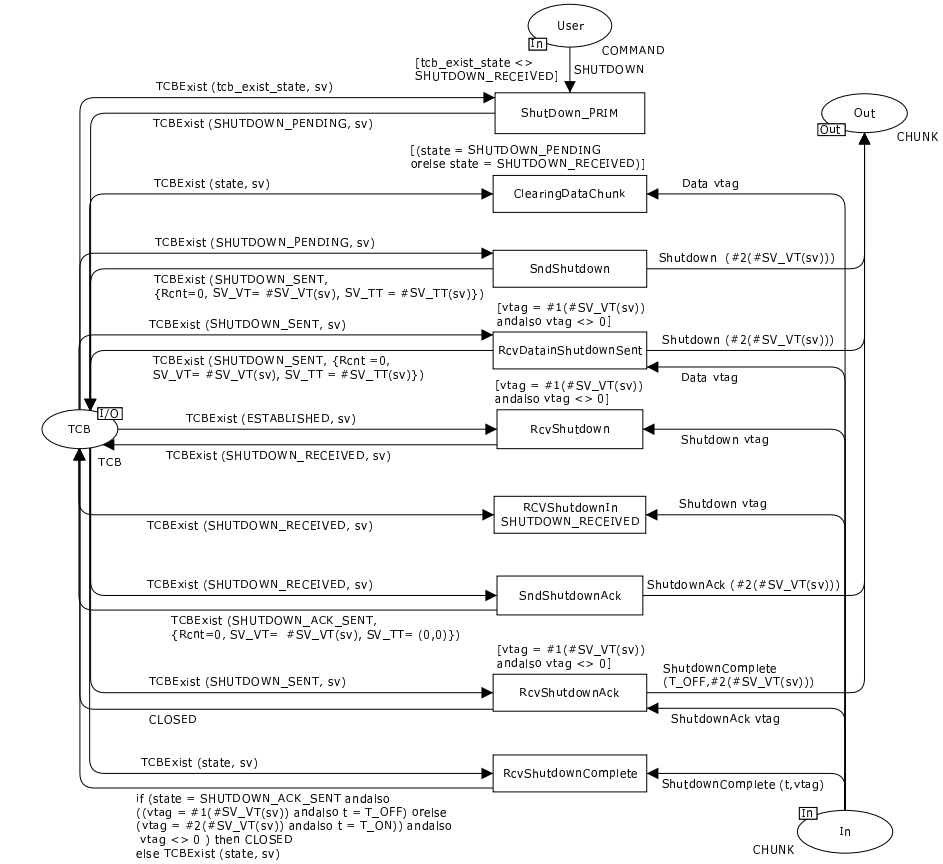

Fig. 10. The Normal'Shut_Down page.

Unexpected Init and InitAck Page Figure 11 shows the CPN page dealing with the unexpected events of receiving Init and InitAck chunks in states other than CLOSED. Transitions RcvInit_CK_WAIT and RcvInit_CK_ECHOED model the actions according to section 5.2.1 of RFC 4960 [4] when an endpoint receives an Init chunk in the COOKIE-WAIT or COOKIE-ECHOED state. The difference between these actions is that the Tie-Tags from the COOKIE-WAIT state are set to zeros but from COOKIE-ECHOED, they are set to the current verification tags. Transitions Rcv_InitOtherThan models the action according to section 5.2.2 of RFC 4960 when the endpoints receive unexpected Init chunks in states other than CLOSED, COOKIE-WAIT and COOKIE-ECHOED. The action is similar to that of transition RcvInit_CK_ECHOED but the "my verification tag" in the cookie and Initial Tag in the InitAck chunk are set to a new value instead of the old value of the Initial tag (InitTag_Z). Transition RcvInit_in_SHUTDOWN_ACK_SENT models the action according to the sixth paragraph of section 9.2 of RFC 4960. After receiving an Init chunk in SHUTDOWN-ACK-SENT, the SCTP node discards the Init chunk but retransmits a ShutdownAck chunk. Transition Rcv_InitAck models the action according to section 5.2.3 of RFC 4960. The SCTP node silently discards any unexpected InitAck chunks if receiving them in states other than COOKIE-WAIT

Unexpected CookieEcho and CookieAck Page Figure 12 shows the CPN page dealing with the unexpected events of receiving CookieEcho chunks in states other than CLOSED; and receiving CookieAck in states other than COOKIE-ECHOED. When receiving CookieAck in states other than COOKIE-ECHOED, (transition Rcv_CookieAck), the SCTP node silently discards the

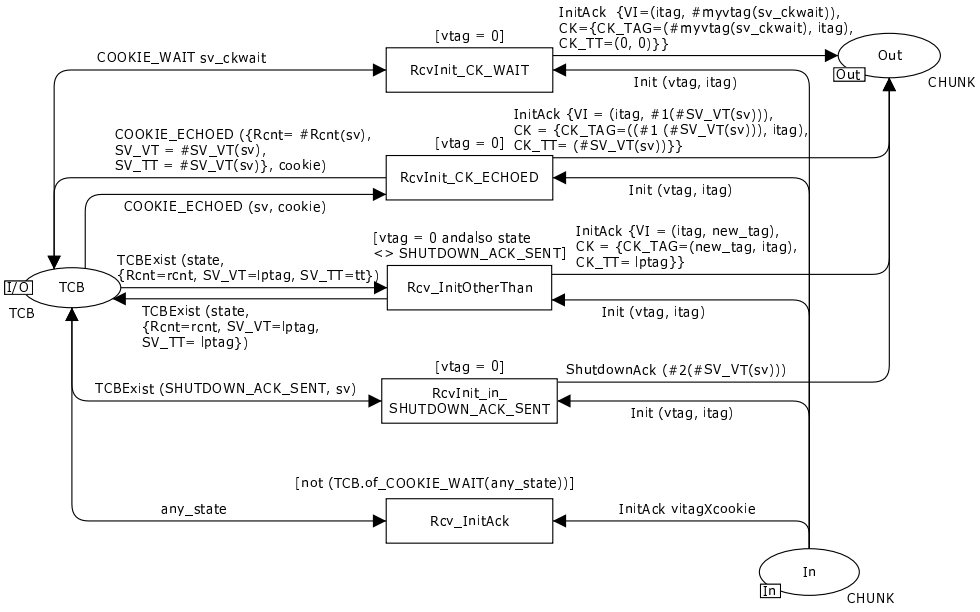

Fig. 11. The Unexpected'Init_InitAck page.

CookieAck chunk. Four substitution transitions, Restart, Simultaneous_Open, Delayed_Cookie and Tags_match, model the actions described in section 5.2.4 of RFC 4960. While we modeled this CPN page, we found an error[2] in Table 2 of section 5.2.4 of RFC 4960. The column headers of the Table, "Local Tag" should be "Local verification tag in the received cookie" and "Peer's tag" should be "Peer's verification tag in the received cookie".

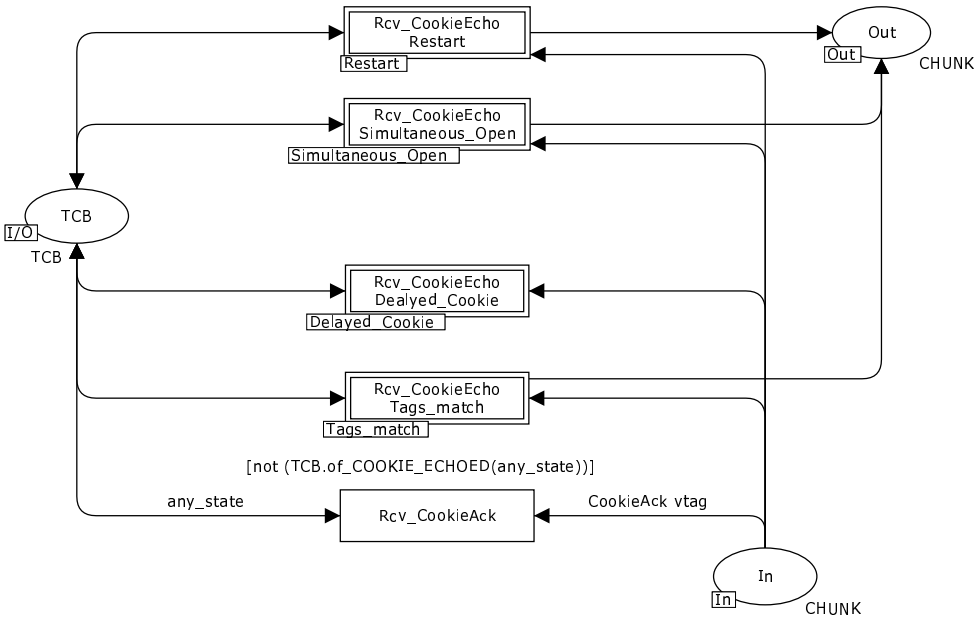

Fig. 12. The Unexpected'CookieEcho_CookieAck page.

Unexpected Shutdown Page This CPN page represents the events when a SCTP endpoint unexpectedly receives shutdown control chunks (Shutdown, ShutdownAck and ShutdownComplete). The states in the establishment phase, COOKIE-WAIT and COOKIE-ECHOED, should not receive the shutdown control chunks. SHUTDOWN-SENT should not receive the Shutdown chunk. SHUTDOWN-ACK-SENT should not receive the ShutdownAck chunk. The reception of shutdown control chunks in CLOSED is modelled by the first transition. The second, third and fourth transitions model the receptions of Shutdown, ShutdownAck and ShutdownComplete respectively in the COOKIE-WAIT and COOKIE-ECHOED states. When a node receives ShutdownAck in either COOKIE-WAIT or COOKIE-ECHOED, the node replies with Shutdown-Complete. The T-Flag is set and the VTAG of the outgoing packet is set equal to the VTAG of the incoming packet.

The fifth transition models the reception of a Shutdown chunk in the SHUTDOWN-SENT state. The node replies with ShutdownAck and enters the SHUTDOWN-ACK-SENT state. The sixth transition represents the reception

[2] See Transport Area Discussion Archive
http://www.ietf.org/mail-archive/web/tsvwg/current/msg08603.html.

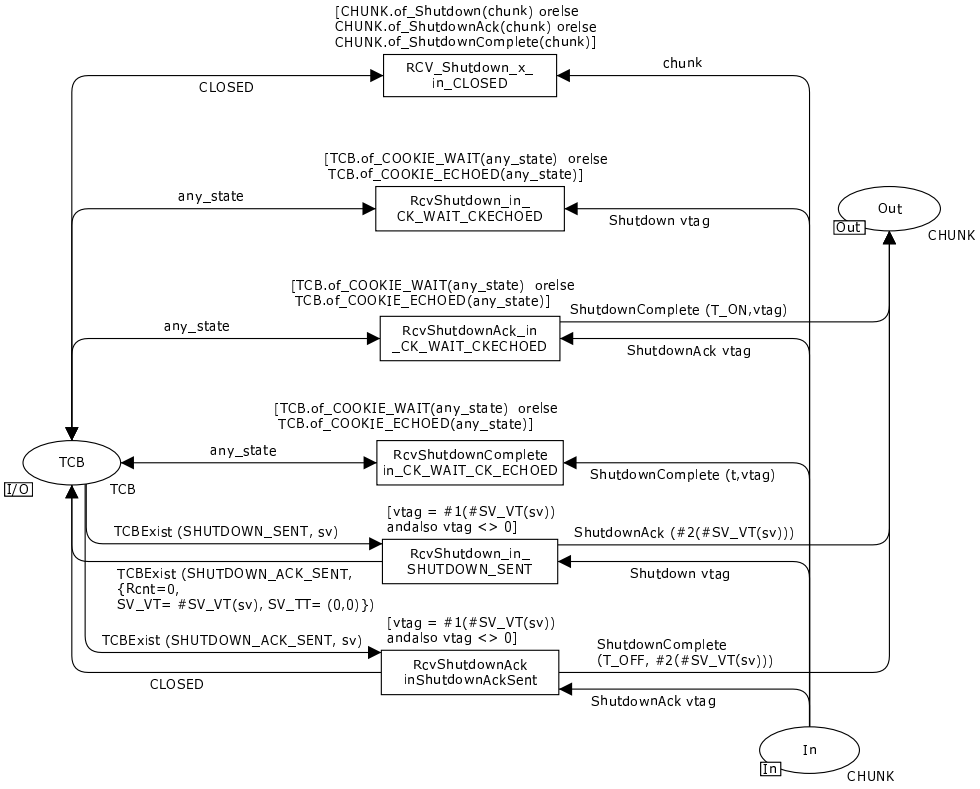

Fig. 13. The Unexpected'Shutdown page.

of a ShutdownAck chunk in the SHUTDOWN-ACK-SENT state. The node replies with ShutdownComplete and goes to CLOSED.

5 Analysis of DCCP-CPN Connection Management Model

5.1 Initial configuration

We analyse our SCTP connection management model using CPN Tools version 2.2.0 on an AMD Athlon 1.79 GHz computer with 2GB RAM. The SCTP-CPN model is initialised by distributing initial tokens to places User_A and User_Z. TCB_A and TCB_Z of the model to create the initial marking. Table 2 shows the initial values of the user commands and TCB. All channel places are empty. All initial tags used during association establishment are randomly generated.

We analyse five cases. Case A is association establishment. Both node A and Z are initially in CLOSED. The user of node A issues an "ASSOCIATE" command to start an association. Case B is simultaneous establishment, where both sides attempt to initiate the association at about the same time. Case C is graceful shutdown. Both sides are in ESTABLISHED, and the user at node A issues a SHUTDOWN command. Case D is simultaneous graceful shutdown when both sides try to close the association at about the same time. Case E is ungraceful abort. Both sides are in ESTABLISHED, and the user at node A

Case	Initial Markings in place			
	User_A	User_Z	TCB_A	TCB_Z
A	1‘ASSOCIATE		CLOSED	CLOSED
B	1‘ASSOCIATE	1‘ASSOCIATE	CLOSED	CLOSED
C	1‘SHUTDOWN		TCBExist (ESTABLISHED,	TCBExist (ESTABLISHED,
D	1‘SHUTDOWN	1‘SHUTDOWN	{Rcnt=0,SV_VT=(2,3)	{Rcnt=0, SV_VT=(3,2)
E	1‘ABORT		SV_TT=(2,3)})	SV_TT=(3,2)})

Table 2. Initial Configurations.

issues an ABORT command. The initial markings for these cases are shown in Table 2.

5.2 Analysis Results

The analysis results of our SCTP Connection Management CPN model using the initial configurations in Table 2 are shown in Table 3. The 4-tuple in the first column is the maximum retransmissions allowed for Init, CookieEcho, Shutdown and ShutdownAck respectively. An “x” indicates that the retransmission of those chunk types does not occur in that configuration. The state space tool (in CPN Tools) provides the number of nodes, arcs and terminal markings. In all cases (A to E) in Table 2 the number of nodes and arcs in the Strongly Connected Component (SCC) Graph are the same as the number of nodes and arcs in the state space. Thus no livelocks are found.

We classify the terminal markings into four categories based on the SCTP endpoint states:

TYPE-I CL-CL: both sides terminate in CLOSED.

TYPE-II EST-EST: both sides terminate in ESTABLISHED.

TYPE-III CL-EST: node A terminates in CLOSED but node Z in ESTABLISHED.

TYPE-IV EST-CL: node A terminates in ESTABLISHED but node Z in CLOSED.

Cases C to E in Table 2 are when the association is terminated. All terminal markings of cases C to E are TYPE-I (CLOSED-CLOSED) which is desirable, and no other deadlocks are found. Case D has three terminal markings because there is the possibility of one user command token (SHUTDOWN) remaining in either place User_A or User_Z.

Case A and B are when SCTP’s nodes attempt to establish an association. TYPE-I terminal marking (CLOSED-CLOSED) is a desirable terminal marking when the association can not be established thus both sides go to CLOSED state (No connection). TYPE-III and TYPE-IV terminal markings occur when one side is in ESTABLISHED while the other is in CLOSED. This can happen when the maximum number of retransmissions of the CookieEcho chunk is reached and the node enters CLOSED before CookieAck arrives. An example of this scenario is shown in Fig. 14. Although TYPE-III and TYPE-IV are unwanted, they are not harmful. This is because when the peer is considered unreachable, SCTP will report the failure to its user so that the user may decide to re-initiate

Case	Nodes	Arcs	Time	Terminal Markings			
			(sec)	(I)CL-CL	(II)EST-EST	(III)CL-EST	(IV)EST-CL
A-(0,0,x,x)	12	13	0	1	1	1	0
A-(1,0,x,x)	38	58	0	1	2	2	0
A-(0,1,x,x)	20	26	0	1	1	1	0
A-(1,1,x,x)	78	147	0	1	2	2	0
A-(2,2,x,x)	345	905	1	1	2	2	0
A-(3,3,x,x)	1,124	3,535	1	1	2	2	0
B-(0,0,x,x)	380	756	0	1	16(11)	4	4
B-(1,0,x,x)	8,206	25,778	41	1	36(20)	7	7
B-(0,1,x,x)	1,087	2,498	1	1	16(11)	4	4
B-(1,1,x,x)	31,295	113,302	579	1	36 (20)	7	7
C-(x,x,0,0)	14	20	0	1	0	0	0
C-(x,x,1,0)	28	50	0	1	0	0	0
C-(x,x,0,1)	22	36	0	1	0	0	0
C-(x,x,1,1)	45	87	0	1	0	0	0
D-(x,x,0,0)	106	234	0	3	0	0	0
D-(x,x,1,0)	415	1,178	0	3	0	0	0
D-(x,x,0,1)	253	598	0	3	0	0	0
D-(x,x,1,1)	1,125	3,604	1	3	0	0	0
D-(x,x,2,2)	6,024	22,294	16	3	0	0	0
E-(x,x,x,x)	3	2	0	1	0	0	0

Table 3. State space analysis results.

the ASSOCIATE command. Thus the association can be restored as described in Fig. 5 of RFC 4960.

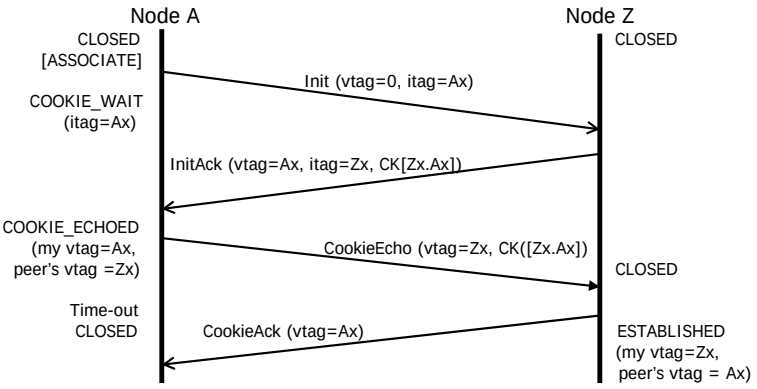

Fig. 14. A scenario leads to a terminal marking TYPE III (half open state).

TYPE-II terminal markings should be desirable when both sides successfully establish the association. However when we check the verification tags stored in the TCBs, some terminal markings of TYPE-II are undesirable. #1SV_VT in TCB_A must equal #2SV_VT in TCB_Z and vice versa, otherwise all received data packets will be discarded. In column TYPE-II, the number in parenthesis is the number of TYPE-II terminal markings in which verification tags between both TCBs match each other. For example, in case B(0,0,x,x), eleven terminal markings of TYPE-II have verification tags matched to each other.

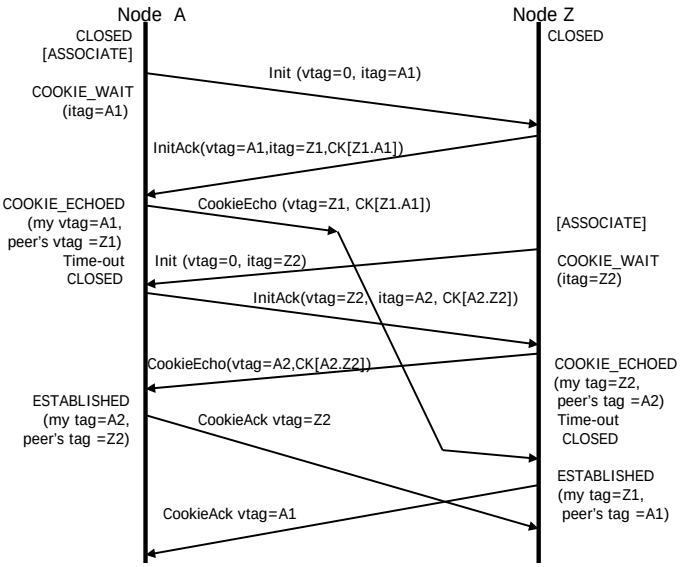

Fig. 15. A scenario when both sides reach ESTABLISHED with mismatched verification tags.

According our investigation, one cause of this problem is shown in Fig. 15. Node A starts initiating the association. The setup sequence proceeds according to the typical scenario but the CookieEcho chunk from node A is delayed. While waiting for CookieAck, node A reaches the maximum number of retransmission of the CookieEcho chunk, thus node A goes to CLOSED. Meanwhile after replying InitAck, node Z initiates the association establishment (simultaneous establishment) using a different set of verification tags. Similar to node A, while waiting for CookieAck, node Z retransmits the maximum number of retransmission of CookieEcho chunks, and goes to CLOSED. After both sides are in CLOSED, CookieEcho's arrive at both sides. Both sides process the cookies and authenticate the State Cookie if it is the one that it has just generated. Both sides create their TCBs from the received cookies and go to ESTABLISHED with mismatched verification tags. Notice that this problem does not relate to Tie-Tags mechanism or section 5.2 of RFC 4960.

6 Conclusions and future work

This paper has presented a Coloured Petri Nets model and analysis of SCTP connection management. Our CPN model is based on the recent RFC 4960 rather than the obsolete RFC 2960. Besides the typical association establishment, graceful shutdown and abort, our CPN model includes the procedures for handling the reception of unexpected control chunks. In particular we attempt to model the use of Tie-Tags from RFC 4960 that differs significantly from the description in RFC 2960.

We build the procedure-based CPN model of SCTP connection management directly from RFC 4960. It took about two months (part time) to study the SCTP procedures, create the model and debug the model. The most critical problem of this project is to understand how to use Tie-Tags. When SCTP receives a CookieEcho chunk in the states other than CLOSED (section 5.2.4

of RFC 4960), Tie-Tags are used to identify complex scenarios such as an association restart and simultaneous establishment. Although we found an error in section 5.2.4 of RFC 4960 while we developed the SCTP-CPN model, to gain an insight into these complex scenarios requires more exhaustive analysis and more time.

Our initial state space analysis shows that the shutdown procedures have no deadlocks but the establishment procedures have undesired deadlocks. The undesired deadlocks are half open states, where one SCTP node is in CLOSED while the other is in ESTABLISHED. This deadlock could be easily solved by restarting the association. When the maximum number of retransmissions has been reached, SCTP must report to its user. Then the user can restart the association.

The second problem seems more severe because both sides are in ESTABLISHED with mismatched verification tags stored in their TCB. As far as we aware there is no existing discussion of this problem. The solution to the second problem seems to involve cookie authentication, which needs further investigation. In future, we are interested in modelling security attacks against SCTP as well as multi-homing.

Acknowledgments The author are thankful to the anonymous reviewers and also to Professor Jonathan Billington and Dr. Guy Gallasch. Their constructive feedback has helped to improve the quality of this paper.

References

1. J. Billington, G. E. Gallasch, and B. Han. A Coloured Petri Net Approach to Protocol Verification. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 210–290. Springer, Heidelberg, 2004.
2. J. Billington and S. Vanit-Anunchai. Coloured Petri Nets Modelling of an Evolving Internet Standard: the Datagram Congestion Control Protocol . *Fundamenta Informaticae*, In Press, 2008.
3. M. Martins M. G. Modelagem e Análise Formal de algumas Funcionalidades de um Protocolo de Transporte Atrvés das Redes de Petri. Master's thesis, Instituto Nacional de Telecomunicações (INATEL) , Santa Rita do Sapucaí, Brazil, December 2003.
4. R. Stewart Ed. Stream Control Transmission Protocol (SCTP), RFC4960. Available via http://www.rfc-editor.org/rfc/rfc4960.txt, September 2007.
5. R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, M. Tuexen. Stream Control Transmission Protocol (SCTP) Specification Errata and Issues, RFC4460. Available via http://www.rfc-editor.org/rfc/rfc4460.txt, September 2007.
6. R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang and V. Paxson. Stream Control Transmission Protocol (SCTP), RFC2960. Available via http://www.rfc-editor.org/rfc/rfc2960.txt, October 2000.
7. S. Vanit-Anunchai and J. Billington. Modelling the Datagram Congestion Control Protocol's Connection Management and Synchronisation Procedures. In J. Kleijn and A. Yakovlev, editors, *Proceedings of the 28th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN'07)*, volume 4546 of *Lecture Notes in Computer Science*, pages 423–444, Siedlce, Poland, 25-29 June 2007. Springer, Heidelberg.
8. S. Vanit-Anunchai, J. Billington, and G.E. Gallasch. A Combined Protocol Channel Model and its Application to the Datagram Congestion Control Protocol. In D. Moldt N. Sidorova and H. Rolke, editors, *Proceeding of the International Workshop on Petri Nets and Distributed Systems (PNDS08)*, pages 32–46, Xian, China,, 23-24 June 2008.