# Validating SCTP Simultaneous Open Procedure

Somsak Vanit-Anunchai*

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

**Abstract.** The Stream Control Transmission Protocol (SCTP) is a reliable unicast transport protocol originally specified by the Internet Engineering Task Force (IETF) in RFC 2960. 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 published RFC 4960, which replaces RFC 2960. Despite of being the revised specification, the descriptions of the simultaneous open and the restart procedures are still unclear and difficult to understand. To clarify this informal specification and gain insights, we formally model and analyse the association management using Coloured Petri Nets. In particular this paper focuses on the Tie-Tag operation and the simultaneous open procedure operating over the simplest channels, First In First Out (FIFO) with no loss. Our analysis reveals errors in which both sides are in ESTABLISHED but the verification tags in both Transmission Control Blocks do not match.

**Keywords:** Coloured Petri Nets, Procedure-based, Verification Tags, Tie-Tags, COOKIE ECHO

## 1 Introduction

The Stream Control Transmission Protocol (SCTP), RFC 4960 [8] is a unicast connection oriented transport protocol providing an error-free reliable flow of data between a client and a server. Originally it was designed by the Signalling Transport working group for transporting telephony signalling messages over UDP. Foreseeing its significance and great potential to become a major transport protocol, IETF decided to operate SCTP over IP instead.

After several years of implementation and testing, fifty-two defects in the original specification (RFC 2960) and solutions were discussed in RFC 4460 [9]. The IETF has published a revised version of the SCTP informal specification, RFC4960 [8], in September 2007, and RFC 2960 has become obsolete. Despite many years of implementing and testing SCTP, it is still important to have a proper formal model and to perform formal analysis of SCTP association management, especially when SCTP is designed for reliable data transfer

* Supported by the the Thai Network Information Center Foundation and the Thailand Research Fund Contract no.TRG 5380023.

F. Arbab and M. Sirjani (Eds.): FSEN 2013, LNCS 8161, pp. 233–249, 2013.
DOI: 10.1007/978-3-642-40213-5_15,

such as signalling in Public Switching Telephone Networks. Previously in [10] we focused on modelling the typical procedure of SCTP association management. This paper places emphasis on the exception handling procedure (handling an unexpected packet) which is more complex. Analysis of a formal model in [10] illustrated that SCTP simultaneous open procedure in RFC 4960 could fall into an undesired final state in which both sides are in ESTABLISHED with mismatched VTAGs. However, it is arguable that these errors could happen only if a packet is reordered and delayed too long so that these defects are unlikely to occur. This paper discovers an error when SCTP operates over First in First out (FIFO) channel without losses. Thus, this error is more likely to occur than those errors previously found.

This paper is organised as follows. Section 2 provides an overview of SCTP association set up. Section 3 discusses the related work and contributions. A brief description of the CPN model of SCTP association management 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 Overview of Stream Control Transmission Protocol

### 2.1 SCTP Packet Format

Figure 1(a) shows an SCTP packet comprising a common header and one or more chunks. The SCTP header contains 16 bit source and destination port numbers, a 32 bit verification tag (VTAG) and a 32 bit checksum. The VTAG is used to protect an association from blind attacks. Each end point keeps two values of VTAG: "local VTAG" and "peer's VTAG". "local VTAG" sometimes is called "My VTAG". In general, any received packets containing a VTAG differing from "local VTAG" will be discarded. On the other hand, sent packets will carry a VTAG equal to "peer's VTAG". These tag values are randomly selected at initialization and exchanged between the end points during association set up.

A Chunk is an information unit. According to RFC 4960, 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 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 [8].

[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.

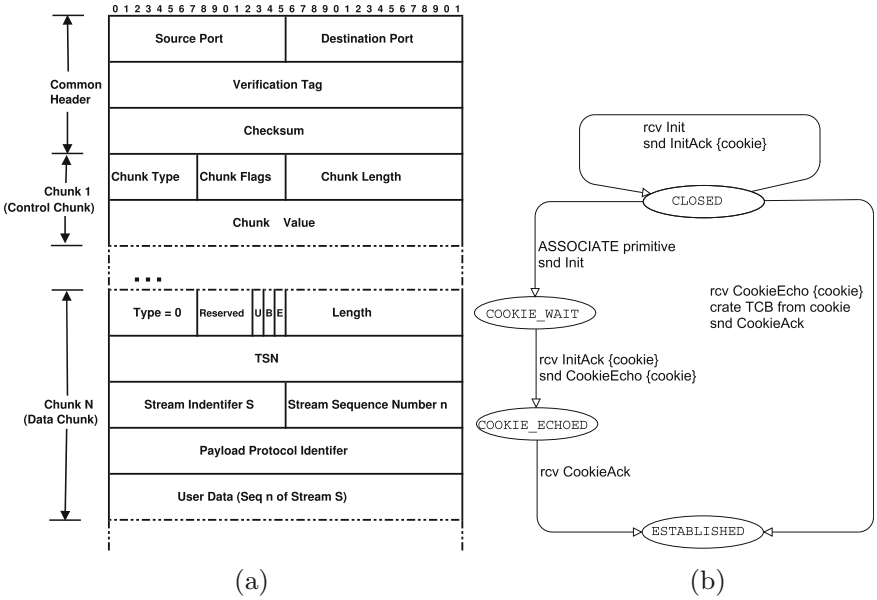


**Fig. 1.** (a) SCTP packet format. (b) SCTP state diagram: association set up (redrawn from [8])

## 2.2 SCTP Association Establishment Procedure

**Normal Association Establishment** Figure 1(b) shows the state diagram when SCTP sets up the association. Figure 2 shows a typical procedure of association establishment. An association between two nodes, A and Z, is initiated by an SCTP user on node "A" issuing an "ASSOCIATE" command. After receiving the "ASSOCIATE" primitive, node A sends an SCTP packet with VTAG equal to zero. This SCTP packet contains only an Init chunk with an initial tag to specify the VTAG of returning 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 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.

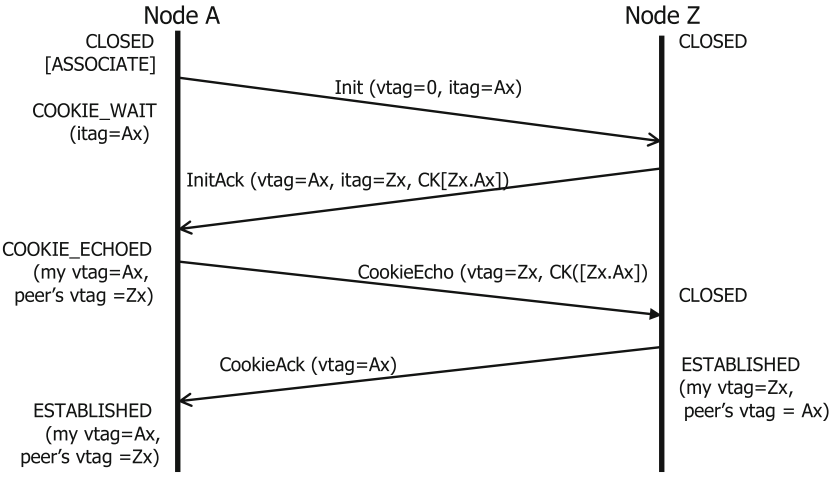


**Fig. 2.** Typical message sequence charts association set up

**Handling Unexpected Init Chunks** The rules for handling duplicate or unexpected Init, InitAck, CookieEcho, and CookieAck chunks are specified in the Section 5.2 of RFC 4960. When SCTP receives an unexpected Init before the association established, SCTP composes a state Cookie using its local VTAG and the initial tag found in the unexpected Init. The Cookie is attached to the outbound InitAck. When an unexpected Init is received after the association established, SCTP composes a state Cookie using a new random number for local VTAG and the initial tag found in the unexpected Init. This implies that the Cookie contains state information of a new connection. When an unexpected Init is received in SHUTDOWN_ACK_SENT, SCTP replies with ShuntDownAck.

**Handling Unexpected CookieEcho Chunks** This procedure specified in RFC 4960 is unclear and subtle. With reference to Table 2 in RFC 4960 (Fig. 3), VTAGs and Tie-Tags are compared to identify which action SCTP shall take. If the conditions in Fig. 3 are not met, SCTP silently discards the received CookieEcho chunk. However Section 5.2 of RFC 2960 and of RFC 4960 define Tie-Tags differently.

According to RFC 2960, Tie-Tags are stored in a state Cookie. They are copies of VTAGs from the existing TCB (local VTAG and peer's VTAG). These Tie-Tags are created when SCTP creates InitAck. Using the Tie-Tags in the restart procedure (Section 5.2.4.1 of RFC 4960), a newly restarting association can be tied to the original association without shutting down and starting a new association. The first two columns of Fig. 3 compare a pair of VTAGs in Cookie with a pair of VTAGs in existing TCB. The third and fourth columns of Fig. 3 compare a pair of Tie-Tags in Cookie with a pair of VTAGs in existing TCB.[2]

[2] [10] uses this comparison which is incorrect.

| Local Tag | Peer's Tag | Local-Tie-Tag | Peer's-Tie-Tag | Action/ Description |
|---|---|---|---|---|
| X | X | M | M | (A) |
| M | X | A | A | (B) |
| M | 0 | A | A | (B) |
| X | M | 0 | 0 | (C) |
| M | M | A | A | (D) |

Table 2: Handling of a COOKIE ECHO when a TCB Exists

Legend:

X - Tag does not match the existing TCB.
M - Tag matches the existing TCB.
0 - No Tie-Tag in cookie (unknown).
A - All cases, i.e., M, X, or 0.

Note: For any case not shown in Table 2, the cookie should be silently discarded.

**Fig. 3.** Table 2 in RFC 4960 from [8]

In order not to reveal the true VTAGs of the existing association, RFC 4960 defines Tie-Tags as two 32-bit random numbers or 64-bit nonce. They are stored in both the state cookie and TCB. When we consider the third and fourth columns of Fig. 3, it is incorrect to compare 64-bit nonce with VTAGs. We should compare a pair of Tie-Tags (64-bit nonce) in Cookie with a pair of Tie-Tags (64-bit nonce) in existing TCB instead.[3]

Each row in Fig. 3 identifies the SCTP's action as follows:

- Action A is the restart scenario when the other side crashes and starts up in CLOSED. SCTP shall continue the association by replacing the existing VTAGs with the ones in Cookie and sending a CookieAck.
- Action B is the simultaneous open scenario when both sides attempt to start an association at about the same time. SCTP shall enter the ESTABLISHED state, update its peers VTAG from Cookie. and then send a CookieAck.
- Action C is when the Cookie is so delayed that SCTP has already sent an Init, received an InitAck and then sent a CookieEcho. The delayed Cookie arrives after the CookieEcho is sent. In this case the delayed Cookie is silently discarded.
- Action D is when both local and peer's VTAG in both Cookie and TCB are matched, SCTP shall enter the ESTABLISHED state and reply with CookieAck.

One subtle ambiguity is the meaning of the zero values in Fig. 3. The values of Tie-Tags are set to zero indicating that no previous TCB existed. Action C

[3] This paper uses this comparison.

requires the conditions that the values of Tie-Tags in the received cookie are zero. The value of peer's tag in TCB can be zero or unknown only when SCTP endpoint is in the COOKIE_WAIT state. It implies that action B in the third row of Fig. 3 occurs when SCTP endpoint is in COOKIE_WAIT.

**Handling Unexpected InitAck and CookieAck Chunks** An unexpected InitAck is simply discarded if SCTP is not in COOKIE_WAIT. SCTP also discards the CookieAck if it is not in COOKIE_ECHO.

## 3 Related Work

### 3.1 Modelling Approach

Coloured Petri Nets [5] are well known for modelling and analysing concurrent and complex systems including validating various transport protocols such as Wireless Application Protocol (WAP) [3], TCP [4], and Datagram Congestion Control Protocol (DCCP) [11]. Our model has been created and maintained using CPN Tools [2] which is a software package for the creation, editing, simulation and state space analysis of CPNs. It supports the hierarchical construction of CPN models [5], using constructs called *substitution transitions*. These transitions hide the details of subnets and allow further nesting of substitution transitions. This technique allows a complex specification to be managed as a series of hierarchically related pages.

According to [1], the hierarchical structure of the CPN model can be classified into three modelling styles: state-based; event-processing and procedure-based. Similar to state tables, the state-based style groups actions in the same state into a CPN page. ITU-T often describes their narrative specification based on the state tables. This approach has the advantage of readability and unspecified actions can be easily discovered during model construction. But its disadvantage comes from redundant specification of the same actions that are common across different states. Hence the event-processing style folds the similar actions across different states into a transition. An example of specification that uses event-processing style is RFC 793 [7] Transmission Control Protocol (TCP). While the event processing style makes the CPN model smaller and easier to maintain, it has some drawbacks with respect to readability. Thus [1] proposed the procedure-based modelling style, which structures the CPN model according to the protocol's functionality. Actually suitability of the modelling style depends on how the narrative specification is written. As long as the model can be read and understood easily alongside the narrative specification, it is a good modelling style. We notice that IETF's transport protocol specifications (TCP, SCTP and DCCP) are more suitable to the procedure-based style. During modelling SCTP association management [10], we discovered that this procedure-based style has two merits. First, using a state-based or event processing style a CPN page contains actions that are scattered in different sections of RFC 4960. Illustrated in [10], with the procedure-based style actions in each CPN page are confined to

only a few sections in RFC 4960. Our SCTP-CPN model in [10] is easy to read alongside RFC 4960. Second, the procedure-based CPN model comprises typical procedures (straight forward) and unexpected procedures (complex). Beginners can pay attention to the typical scenarios before getting into the complex procedures later.

### 3.2 Comparing to the SCTP-CPN Model by Others

Despite a lot of work on SCTP's security, performance and multi-homing, we have found only three works [12,6,10] that use Coloured Petri Nets to model SCTP association management operating over reordering channels. The CPN model in [6] followed the state-based style whereas [12] used the event-processing style similar to [4]. Our CPN model in [10] was the procedure-based style following the approach proposed in [1]. The work in [10] attempted to build the CPN models according to the revised specification, RFC 4960 while [6,12] used RFC 2960 which was obsolete. The CPN models in [6,12] were incomplete because [6] did not include the procedure when SCTP nodes receive duplicated or unexpected messages and [12] assumed no retransmission.

### 3.3 Contributions

The contribution of this paper is three-fold. First, while [10] illustrates a CPN model of typical SCTP's association management, it leaves out the model of handling an unexpected CookieEcho chunk partly because it was not well understood at that time. In this paper, we attempt to finish up the CPN model of handing an unexpected CookieEcho chunk. Second, [10] followed Fig. 5 (a restart example) of RFC 4960 and used the Tie-Tags as "old VTAGs" instead of 64-bit nonce. We compared the pair of Tie-Tags in Cookie with the pair of VTAG in existing TCB. Unfortunately, it turns out that Fig. 5 of RFC 4960 is incorrect[4]. Nevertheless, those errors uncovered in [10] were not related to this mistake. This paper attempts to rectify the mistake in [10]. We use Tie-Tags as 64-bit nonce and compare the pair of Tie-Tags in Cookie with the pair of Tie-Tags in existing TCB. This correction leads us to an undesired terminal marking in which both sides are in ESTABLISHED with a mismatched VTAG. This error scenario happens even when SCTP operates over FIFO channels without loss.

Third, after we have investigated the actions in Fig. 3 using state space analysis, we found two implementation flaws. Firstly, the implementor does not need to check the condition for action C because the Cookie will be discarded anyway if the conditions in Fig. 3 are not met. Secondly, the condition of action B in the third row (Fig. 3) has never been reached because the condition of action B in the second row is always satisfied before reaching the third row. We suggest the implementor checking the condition of the third row before checking the second row.

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

## 4 CPN Model of SCTP Association Management

Space limitation prevents us from including all CPN model pages. This paper focuses on handling an unexpected CookieEcho which is excluded from [10]. However, for sake of completeness, we shall briefly describe the model starting from the top level toward the `Unexpected CookieEcho and CookieAck` Page. The rest of the model and its declarations can be found in [10].

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 five places. Places `User_A` and `User_Z` represent application users represented by `COMMAND`. Places `ITAG_A` and `ITAG_Z` contain 32-bit random values of the initial verification tags. Places `TCB_A` and `TCB_Z` model Transmission Control Block represented by `TCB`. Both end points are connected via two channel places, `CH_A2Z` and `CH_Z2A`. We assume that during association set up and closing down a packet contains only one chunk represented by `CHUNK`. To form a FIFO queue the channel places are represented by a list of `CHUNK` (`L_CHUNK`). 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` in Fig. 4 are linked to the second level page named `SCTP_Procedures` (Fig. 5 (a)). We divide `SCTP_Procedures` into five categories: normal events; unexpected events; retransmission; abort; and checking invalid tags. Substitution transition `UnexpectedEvents` in Fig. 5 (a) links to the CPN page `Unexpected` (Fig. 5 (b)). Handling unexpected receptions of SCTP control chunks is modelled by three CPN pages: `Int_IntAck`, `CookieEcho_CookieAck`, and `Shutdown`.

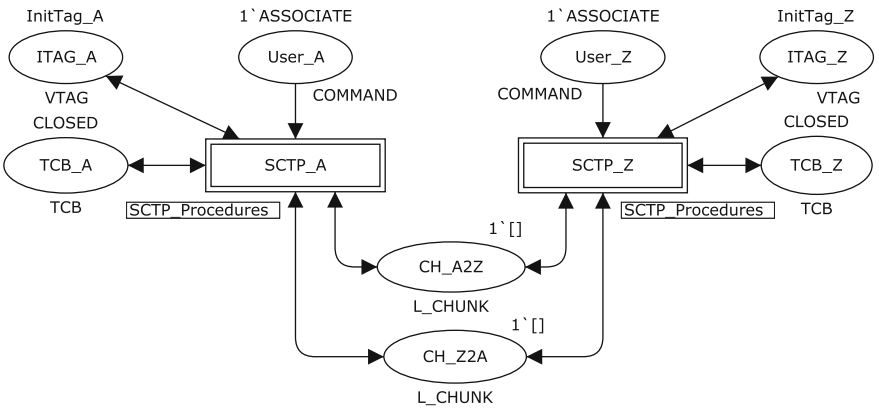


**Fig. 4.** The Top-level CPN page

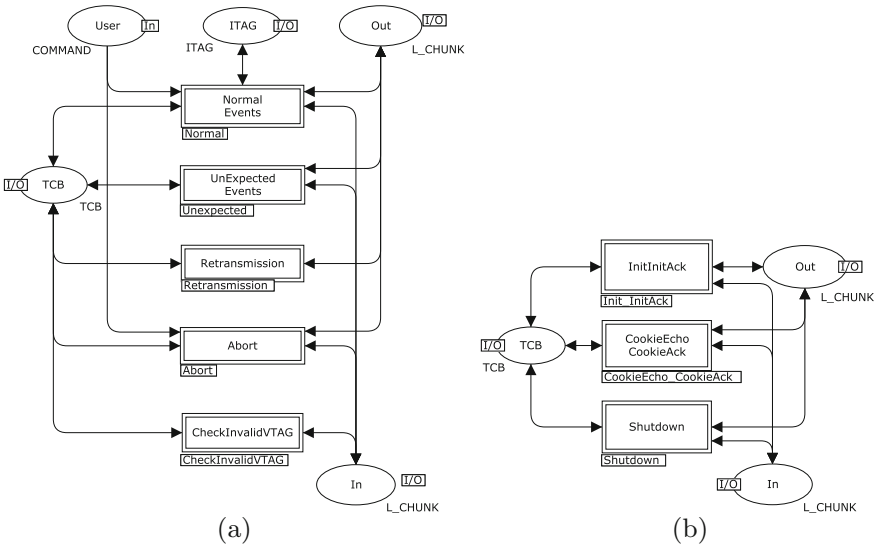


**Fig. 5.** (a) The SCTP_Procedures page. (b) The Unexpected page

### 4.1 Unexpected Init and InitAck Page

Figure 6 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 [8] 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 64-bit nonce. 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, COOKIE_ECHOED and SHUTDOWN_ACK_SENT. The action is similar to that of transition `RcvInit_CK_ECHOED` but the "local VTAG" 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. 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.

### 4.2 Unexpected CookieEcho and CookieAck Page

Substitution transition `CookieEcho_CookieAck` in Fig. 5 (b) links to the CPN page `CookieEcho_CookieAck` (Fig. 7). The first four substitute transitions in

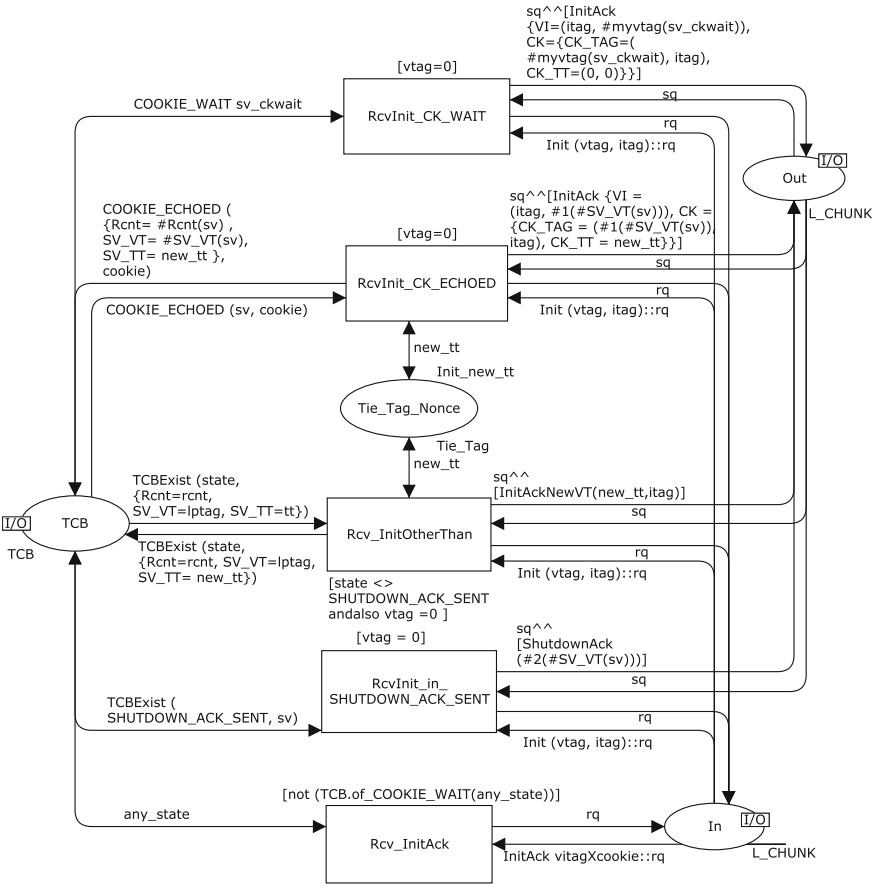


**Fig. 6.** The Unexpected'Init_InitAck page

Fig. 7 represent the actions when SCTP receives an unexpected CookieEcho chunk. The last transition models when SCTP receives CookieAck in states other than COOKIE_ECHOED.

**The Restart page** Substitute transition `Restart` in Fig. 7 is linked to the `Restart` page shown in Fig. 8. This page models action A of Fig. 3. SCTP replaces its VTAGs with the VTAGs in the received Cookie and replies with CookieAck. If SCTP is in SHUTDOWN_ACK_SENT, it retransmits ShutdownAck.

**The Simultaneous_Open page** Substitute transition `Simultaneous_Open` in Fig. 7 is linked to the `Simultaneous_Open` page shown in Fig. 9. This page models action B of Fig. 3. This page includes the actions when SCTP receives

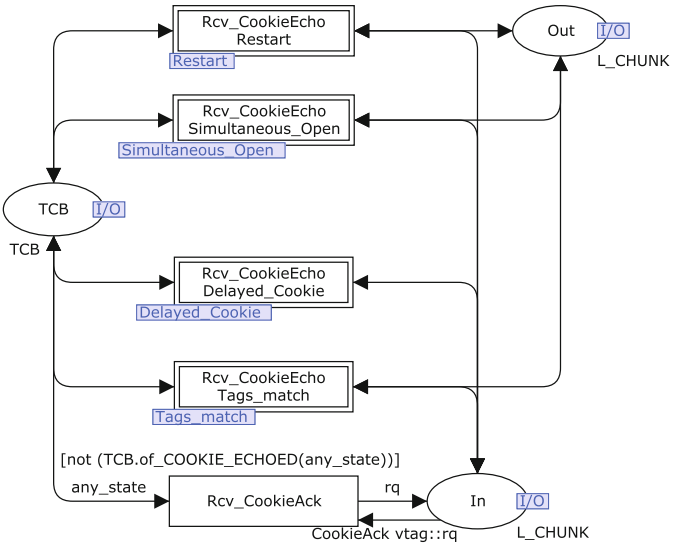


**Fig. 7.** The Unexpected'CookieEcho_CookieAck page

an unexpect CookieEcho in COOKIE_WAIT. It also include the actions when the conditions in Fig. 3 are not met (Case E).

**The Delayed_Cookie page** Substitute transition `Delayed_Cookie` in Fig. 7 is linked to the `Delayed_Cookie` page shown in Fig. 10. This page models action C of Fig. 3. SCTP silently discarded the delayed Cookie.

**The Tags_match page** Substitute transition `Tags_match` in Fig. 7 is linked to the `Tags_match` page shown in Fig. 11. This page models action D of Fig. 3. SCTP replies with CookieAck and enters ESTABLISHED.

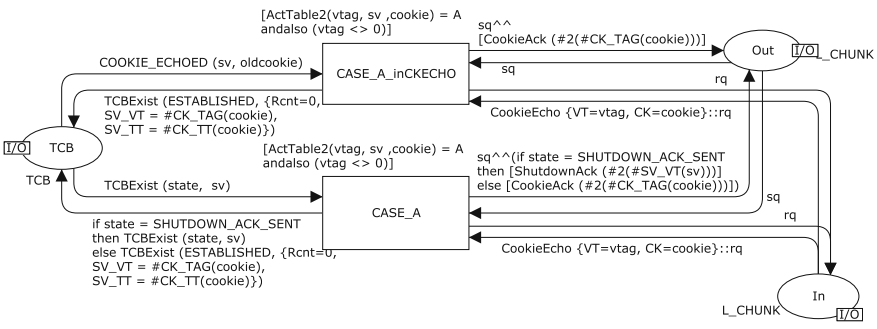


**Fig. 8.** The Restart page

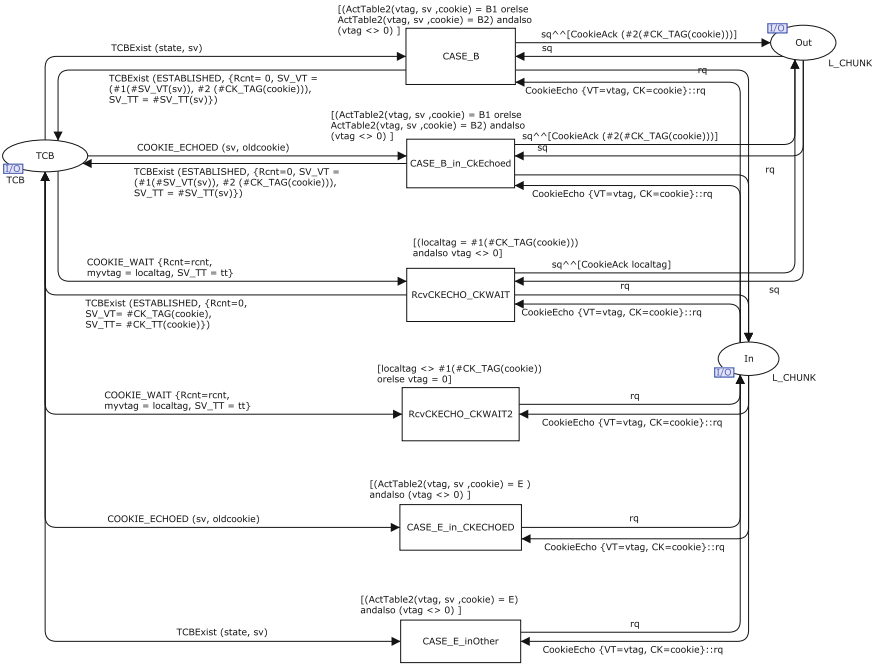


**Fig. 9.** The Simultaneous_Open page

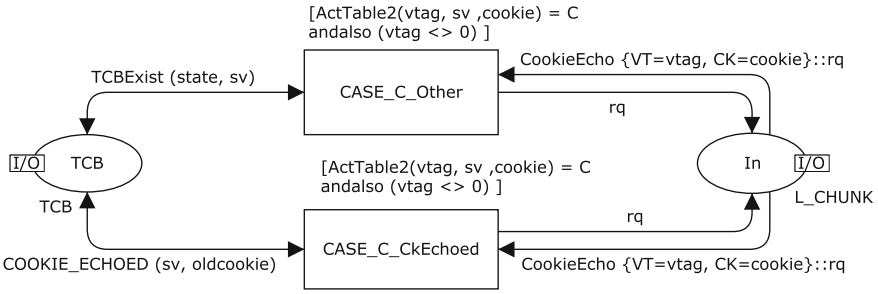


**Fig. 10.** The Delayed_Cookie page

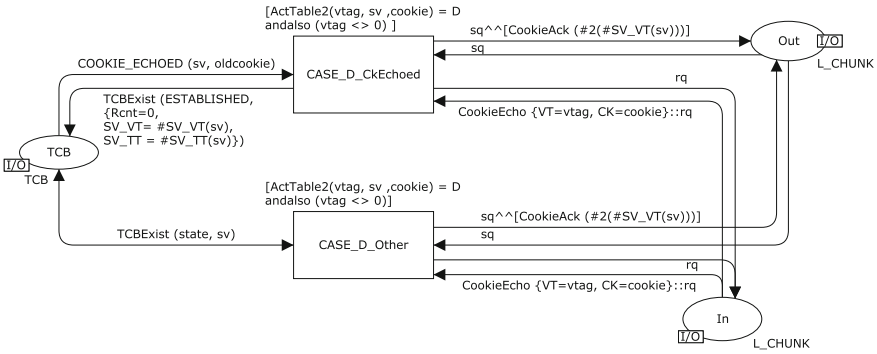


**Fig. 11.** The Tags_Matched page

## 5 Analysis of SCTP-CPN Association Management Model

### 5.1 Initial configuration

We analyse our SCTP association management model using CPN Tools version 3.2.2 on an Intel(R)Core i5 2.67 GHz computer with 4GB RAM. The SCTP-CPN model is initialised by distributing initial tokens to the places shown in Fig. 4. No packet is in both channel places. Places ITAG_A and ITAG_Z store initial verification tags which are randomly generated. Place Tie_Tag_Nonce in Fig. 6 contains a pair of 32-bit random numbers for Tie-Tags.

### 5.2 Analysis Results

The analysis results of our SCTP simultaneous open CPN model are shown in Table 1. The 2-tuple in the first column is the maximum retransmissions allowed for Init and CookieEcho. The state space tool in CPN Tools provides the number of nodes, arcs and terminal markings. In all cases 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 terminal marking (CLOSED-CLOSED) is a desirable terminal marking when the association cannot be established thus both sides go to CLOSED state (No connection). TYPE-III and TYPE-IV terminal markings[5] 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. 12. Although TYPE-III and TYPE-IV are unwanted, they are not harmful. This is because SCTP in CLOSED will report

[5] TYPE-III: node A terminates in CLOSED but node Z in ESTABLISHED.
TYPE-IV: node A terminates in ESTABLISHED but node Z in CLOSED.

**Table 1.** State space analysis results

| Case | Nodes | Arcs | Time | Terminal Markings | | | |
|---|---|---|---|---|---|---|---|
| | | | (sec) | (I)CL-CL | (II)EST-EST | (III)CL-EST | (IV)EST-CL |
| (0,0) | 278 | 444 | - | 1 | 15(0) | 8 | 8 |
| (0,1) | 663 | 1,158 | - | 1 | 19(2) | 9 | 9 |
| (0,2) | 1,353 | 2,554 | 00:00:02 | 1 | 21(2) | 10 | 10 |
| (0,3) | 2,458 | 4,892 | 00:00:04 | 1 | 23(2) | 11 | 11 |
| (0,4) | 4,098 | 8,458 | 00:00:06 | 1 | 25(2) | 12 | 12 |
| (0,5) | 6,405 | 13,564 | 00:00:15 | 1 | 27(2) | 13 | 13 |
| (0,6) | 9,523 | 20,548 | 00:00:25 | 1 | 29(2) | 14 | 14 |
| (1,0) | 10,242 | 18,820 | 00:00:28 | 1 | 74(0) | 37 | 37 |
| (0,7) | 13,608 | 29,774 | 00:00:22 | 1 | 31(2) | 15 | 15 |
| (0,8) | 18,828 | 41,632 | 00:00:40 | 1 | 33(2) | 16 | 16 |
| (1,1) | 27,433 | 54,104 | 00:02:51 | 1 | 102(8) | 47 | 47 |
| (1,2) | 65,589 | 135,134 | 00:17:13 | 1 | 122(8) | 57 | 57 |
| (1,3) | 139,919 | 296,178 | 01:29:02 | 1 | 142(8) | 67 | 67 |

the failure to its user so that the user may decide to re-initiate the ASSOCIATE command. Thus the association can be restored as described in Fig. 5 of RFC 4960 (a restart example).

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. "peer's VTAG" of node A must equal "local VTAG" of node Z and vice versa, otherwise 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 do not match each other.

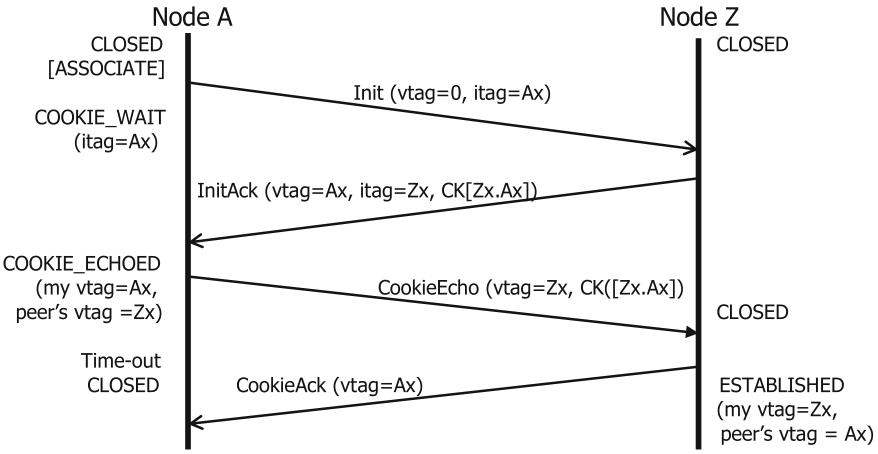


**Fig. 12.** A scenario leads to a terminal marking TYPE III (half open state)

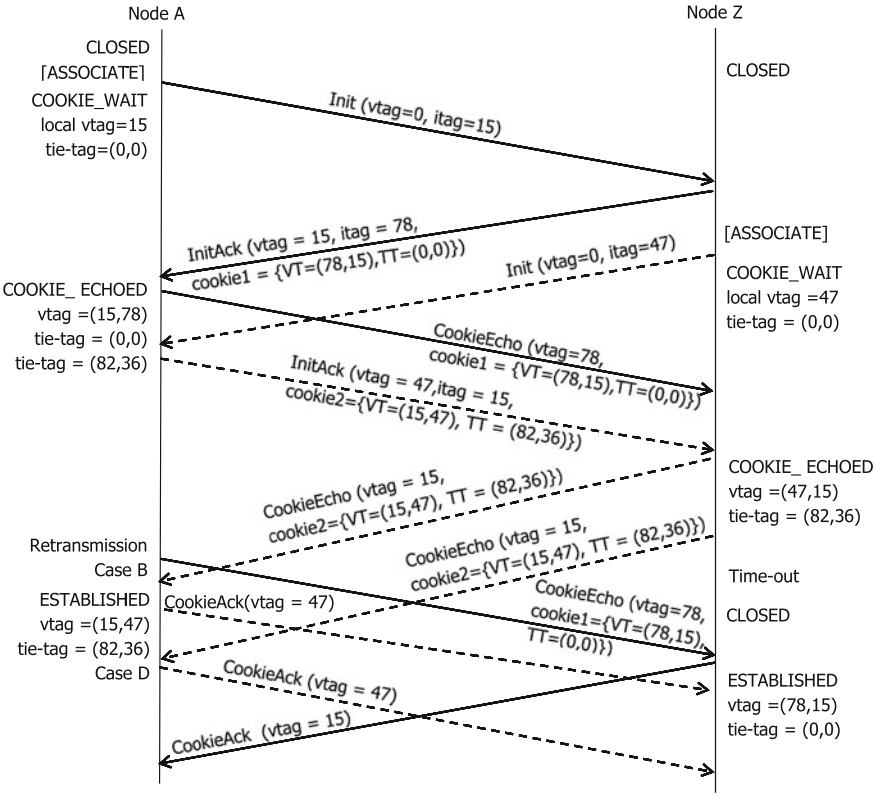


**Fig. 13.** A scenario leading to an undesired terminal marking TYPE II with mismatched VTAGs

Figure 13 shows a message sequence diagram leading to an undesired deadlock for case (0,1). Node A starts initiating the first connection (solid line). After replying with InitAck (itag=78), node Z initiates the second connection (dot line) using a different initial tag (itag=47). Node Z in COOKIE_WAIT keeps discarding the returned CookieEcho of the first connection (solid line) because the conditions in Fig. 3 are not met. After receiving InitAck of the second connection (dot line) and replying with CookieEcho, node Z stays in COOKIE_ECHOED state. Owing to the condition of action C in Fig. 3, node Z in COOKIE_ECHOED keeps discarding the returned CookieEcho of the first connection (solid line). When the timer expires or an intermittent fault occurs, node Z enters the CLOSED state. After node Z in CLOSED receives the CookieEcho of the first connection (solid line), it restores ESTABLISHED state from the received cookie. After node A in COOKIE_ECHOED, receives the CookieEcho of the second connection (dot line), it enters the ESTABLISHED state (*Action B*). Note that in Fig. 13 the CookieEcho of the first connection (solid line) is always sent by node A and the CookieEcho of the second connection (dot

line) is always sent by node Z. After both sides are in ESTABLISHED, "peer's VTAG" of node A (47) is not equal to "local VTAG" of node Z (78). Node A can receive data but will not receive any acknowledgement from node Z. Node Z cannot receive any data from node A. Node Z cannot get into the restart procedure immediately because it is not in CLOSED yet. Node Z and node A have to wait for time-out before closing down the association.

## 6 Conclusions and Future Work

This paper has presented a Coloured Petri Nets model and analysis of SCTP simultaneous open procedure. While constructing the SCTP-CPN model, we identify the incorrect description of the *Tie-Tags* in RFC 4960. Our rigorous analysis shows that SCTP simultaneous open procedure, operating over FIFO channels with no loss, could fail into an undesired deadlock. Both sides in ESTABLISHED could have mismatched verification tags in their TCBs. When the server is located behind middle-boxes such as fire wall or Network Address Translators (NAT), the transport protocols (UDP, TCP, DCCP and SCTP) normally use simultaneous open procedures. Nowadays NATs are widely deployed so that these defects in simultaneous open procedures should not be overlooked.

Formal analysing connection management of the other transport protocols: WAP [3], TCP [4] and DCCP [11], reveal no error when the protocols operate over FIFO channels with no loss. Usually errors could appear when the protocols operate over reordering and/or lossy channels. But the deadlock shown in Fig. 13 does not require any reordered or irregularly delayed packets. Although the odds of this particular scenario is low, the number of terminal markings Type II in Table 1 suggests that depending on the number of retransmitted Init Chunks, there are a large number of possible scenarios leading to the similar deadlock.

SCTP includes various capabilities, such as the restart and multi-homing procedures, aiming for high reliability or fault tolerance applications. When SCTP nodes enter the deadlock state, they have to wait for time-out before closing down the association. This delay degrades SCTP performance. As far as we are aware, this kind of errors has not been raised before. Given the above reasons and the enormous number of potential SCTP connections in the Internet, we consider that this problem could be a serious threat to SCTP applications especially when the high reliability is required.

In future, we are interested in modelling SCTP operating via Network Address Translators (NAT) with multi-homing functions.

**Acknowledgements.** This work is supported by Research Grant from the Thai Network Information Center Foundation and the Thailand Research Fund. The author is thankful to Jonathan Billington, Guy Gallasch and the anonymous reviewers. Their constructive feedback has helped the author improve the quality of this paper.

## References


1. Billington, J., Vanit-Anunchai, S.: Coloured Petri Net Modelling of an Evolving Internet Standard: the Datagram Congestion Control Protocol. Fundamenta Informaticae 88(3), 357–385 (2008)
2. CPN Tools home page, http://wiki.daimi.au.dk/cpntools-help/_home.wiki
3. Gordon, S.: Verification of the WAP Transaction Layer uisng Coloured Petri Nets. PhD thesis, Institute for Telecommunications Research and Computer Systems Engineering Centre, School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia (November 2001)
4. Han, B.: Formal Specification of the TCP Service and Verification of TCP Connection Management. PhD thesis, Computer Systems Engineering Centre, School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia (December 2004)
5. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, Heidelberg (2009)
6. Martins, M.G.M.: 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) (Only available in Portuguese)
7. Postel, J.: Transmission Control Protocol (TCP), RFC793 (September 1981), http://www.rfc-editor.org/rfc/rfc793.txt
8. Stewart, R. (ed.): Stream Control Transmission Protocol (SCTP), RFC4960 (September 2007)
9. Stewart, R., Arias-Rodriguez, I., Poon, K., Caro, A., Tuexen, M.: Stream Control Transmission Protocol (SCTP) Specification Errata and, Issues, RFC4460 (September 2007)
10. Vanit-Anunchai, S.: Toward Formal Modelling and Analysis of SCTP Connection Management. In: The 9th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Department of Computer Science, University of Aarhus, October 20–23, pp. 163–182 (2008)
11. Vanit-Anunchai, S.: An Investigation of the Datagram Congestion Control Protocol's Connection Management and Synchronisation Procedures. PhD thesis, Computer Systems Engineering Centre, School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia (November 2007)
12. Wang, J., Zhang, S., Chen, F.: Modelling and Verification of SCTP Association Management Based on Coloured Petri Nets. In: 2008 ISECS International Colloquium on Computing, Communication, Control, and Management, Guangzhou, China, August 3–4, pp. 379–383. IEEE Computer Society (2008)