Validating DCCP Simultaneous Feature Negotiation Procedure

Somsak Vanit-Anunchai

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

Abstract. This paper investigates the feature negotiation procedure of the Datagram Congestion Control Protocol (DCCP) in RFC 4340 using Coloured Petri Nets (CPNs). After obtaining a formal executable CPN model of DCCP feature negotiation, we analyse it using state space analysis. The experimental result reveals that simultaneous negotiation could be broken on even a simple lossless FIFO channel. In the undesired terminal states, the confirmed feature values of Client and Server do not match.

Keywords: Datagram Congestion Control Protocol, Feature Negotiation, Coloured Petri Nets, State Space Analysis

1 Introduction

In 2006, the Internet Engineering Task Force (IETF) published a set of standards for a transport protocol, the Datagram Congestion Control Protocol (DCCP) [15] comprising RFC 4336 [6]; RFC 4340 [16]; RFC 4341 [7] and RFC 4342 [9]. RFC 4336 discusses problems and disadvantage of existing transport protocols and the motive for designing a new transport protocol for unreliable datagrams. RFC 4340 specifies reliable connection management procedures; reliable negotiation of options; acknowledgement and optional mechanisms used by congestion control mechanisms. RFC 4340 also provides the extension for modular congestion control, called *Congestion Control Identification* (CCID) but the congestion control mechanisms themselves are specified in other RFCs. Currently there are three published standards, RFC 4341, CCID2: TCP-like congestion control [7], RFC 4342, CCID3: TCP-Friendly Rate Control [9] and CCID4: RFC 5622 TCP-Friendly Rate Control for Small Packets [8].

1.1 Motivation

Unlike TCP, DCCP does not impose flow control on data transfer. But state information such as the sequence number sent and received is still required in order to trace packet loss which is crucial for congestion control. From the sequence number variables, a sequence number validity window is set up [16] to defend

against attacks from hackers. Thus *connection management procedures* specified in RFC4340 are used to set up and clear the state information. Apart from the reliable connection management, both sides must choose congestion control mechanisms and agree upon the same CCID. This requires a reliable negotiation procedure called *Feature Negotiation* which is also specified in RFC4340. If both sides are not aware of reaching an agreement with different CCIDs, the situation will be very harmful and currently there is no recovery mechanisms. Hence it is vital to verify that the DCCP feature negotiation procedure works correctly. In this paper we use Coloured Petri Nets (CPNs) [12] to formally model and analyse DCCP feature negotiation procedures.

1.2 Related Work

Formal methods [1] are techniques based on mathematically defined syntax and semantics for the specification, development and verification of software and hardware systems. They remove ambiguities and are indispensable for checking correctness of high-integrity systems. Coloured Petri Net (CPN) [14] is a formal method which is widely used [2,3,5,13,17] to model and analyse concurrent and complex system. An important advantage of CPNs is its graphical notation with the abstract data types providing a high level of user friendliness. CPNs were used to verify industrial scale protocols such as the Wireless Application Protocol (WAP) [10], the Internet Open Trading Protocol (IOTP) [18], TCP [11] and DCCP [21]. [21] studied DCCP connection management operating over reordering channels with no loss using Coloured Petri Nets. [23] extended the work in [21] by including DCCP simultaneous open procedure (RFC 5596) and Network Address Translators (NAT) in the model. However regarding DCCP feature negotiation procedure, there are very few articles [19,20] investigating it. As far as we are aware of, DCCP feature negotiation has not been formally modelled and analysed before.

1.3 Contribution

The contribution of this paper is three fold. Firstly, as far as we are aware of this paper presents the first formal executable model of DCCP feature negotiation. Secondly the formal analysis helps us identify an error in the specification. Thirdly, investigating the state space analysis provides us insight what causes the error.

This paper is organised as follows. Section 2 provides an overview of the protocol and packet format. Section 3 briefly describes DCCP feature negotiation procedure. The description of the CPN model of DCCP feature negotiation is described in section 4, which starts with modelling assumptions and specification interpretation. Section 5 discusses analysis result and insight. Section 6 presents the conclusion of this paper and future work.

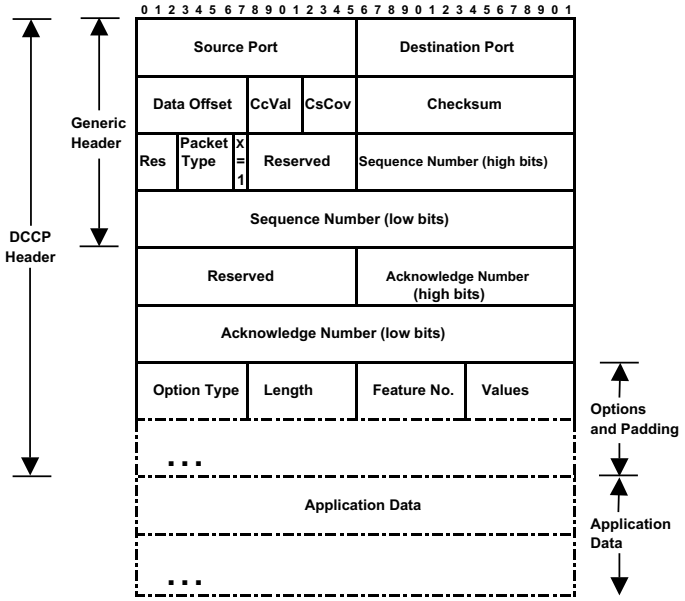

Fig. 1. DCCP packet format.

2 DCCP Overview

The Internet protocol architecture is organized into five layers known as the TCP/IP reference model. While TCP is a transport protocol that provide the reliable delivery of a byte stream, DCCP is a transport protocol for the timely but unreliable delivery of datagrams. DCCP can be viewed as an upgraded version of UDP equipped with new facilities for connection management; acknowledgement; feature negotiation and congestion control.

2.1 DCCP Packet Format

DCCPs exchange packets over the Internet Protocol between a client and a server. The protocol uses 11 packets to setup and release connections and transfer data. RFC 4340 [16] defines a DCCP packet as a sequence of 32 bit words comprising a DCCP Header and Application Data area as shown in Fig. 1. The header comprises a generic header (applicable to all packets), followed by an acknowledgement number (if any) and then the options field. The length of the Option and Application Data fields can vary.

The DCCP header contains 16 bit source and destination port numbers, and a 16 bit checksum. An 8 bit data offset indicates the length in 32-bit words from the beginning of the Header to the beginning of the Application data. CCVal, a 4 bit field, is a value used by the congestion control mechanisms [9]. Checksum

Coverage (CsCov), also a 4 bit field, specifies the part of the packet being protected by the 16 bit checksum. The four bit Packet Type field specifies the name of the packet: Request, Response, Data, DataAck, Ack, CloseReq, Close, Reset, Sync, SyncAck and Listen. Request and Data packets do not include acknowledgement numbers. The sequence numbers of Data packets and the sequence numbers and acknowledgement numbers of Ack and DataAck packets can be reduced to 24-bit short sequence numbers when setting the Extend Sequence Number (X) field to 0.

The Options field contains state information or commands for applications to negotiate various features such as the Congestion Control Identifier (CCID) and the width of the Sequence Number validity window [16].

2.2 Options Fields

The options field is a multiple of 32-bit words which may contain more than one option. Because each option consists of a multiple of 8 bits, the field may need to be padded to the word boundary. Options are classified into two groups: single byte and multi-byte. A single byte option has a value from 0 to 31 which represents an option type. An Option type is a 8-bit integer which represents the meaning of the option, such as 1 meaning mandatory, 2 meaning slow receiver [16]. The format of a multi-byte option is shown in Fig. 1. The first byte is an option type. The second byte is the length in bytes of each option including the option type field, the length and data of the option. The data comprises a number of features, the format of which will be explained in section 3.

3 Feature Negotiation Procedure

DCCP allows both the client and server to change their parameters called *features* using feature negotiation procedures. The negotiation can happen at any time but typically during connection establishment. Each entity can initiate the negotiation of two kinds of features: *local features* (L)-the initiator's features and *remote features* (R)-the other side's features. Four particular options are dedicated to feature negotiations; Change L, Confirm L, Change R and Confirm R. The option types have values of 32 to 35 respectively. The format of Confirm or Change Options including feature numbers and feature values are shown in

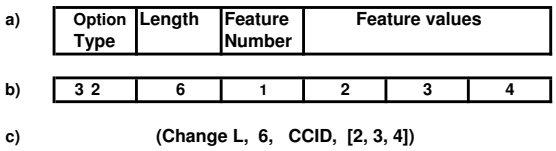

Fig. 2. Option format in DCCP header and an example of a Change L option.

Fig 2 a). Fig 2 b) shows six of 8-bit values representing a Change L option when negotiating CCID. The meaning of each 8-bit values is shown in Fig 2 c).

The feature number identifies the feature. For instance, 1 refers to CCID and 2 means short sequence numbers are allowed. The complete list of features is given in [16]. To reach agreement on a feature value, a reconciliation rule known to both sides is required. Currently RFC 4340 defines two reconciliation rules: server priority and non-negotiable.

1. The server priority rule: This rule is applied when the feature value is a fixed-length byte string. During negotiation DCCP entity keeps an ordered preference list of the feature values. The initiator sends a Change option containing its preference list. The receiver responds with the Confirm option containing an agreed value followed by its preference list. Thus the agree value will appear twice in the Confirm option. The agreed value is defined as the first element in the server's list that matches any element in the client's list. If there is no match, the agreed value remains the existing feature value.

For example, the client sends *32,6,1,2,3,4* corresponding to Change L(32), length(6), CCID(1), the client's preference list(2,3,4). This means the client proposes to change its CCID and the preferred CCIDs are CCID#2, CCID#3 and CCID#4 respectively. The server responds *35,7,1,3,3,4,2* corresponding to Confirm R(35), length(7), CCID (1), agreed value (3) and the server's preference list (3,4,2). According to the Client's and Server's preference lists in this example, the client must use CCID#3.

2. Non-negotiable rule: The Change and Confirm options under this rule contain only one feature value which is a byte string. After receiving the Change L from the feature local, the feature remote must accept the valid value and reply with Confirm R containing this value. If the received feature value is invalid, the feature remote must send an empty Confirm R. This non-negotiable rule must not be used with Change R and Confirm L options.

For example the client sends *32,9,3,0,0,0,0,4,0* corresponding to Change L(32), length(9), Sequence number window (3), value of window size(1024). The server replies with *35,9,3,0,0,0,0,4,0*.

3.1 Finite State Machines

The feature negotiation procedures are represented by state diagrams. Figure 3 shows the state diagram for *feature local*. It comprises three states: STABLE; CHANGING; and UNSTABLE. The entity in the STABLE state always knows its feature value and expects the other end agrees with the same value. When the local receives Change R, it calculates a new agreed value and replies Confirm L. On the other hand the Confirm R received will be discarded.

After the entity in STABLE sends the first Change L command, it enters the CHANGING state and goes back to the STABLE state upon receiving a Confirm R or a empty Confirm R. When the local in CHANGING does not get reply from the other side, it keeps retransmit the Change L option.

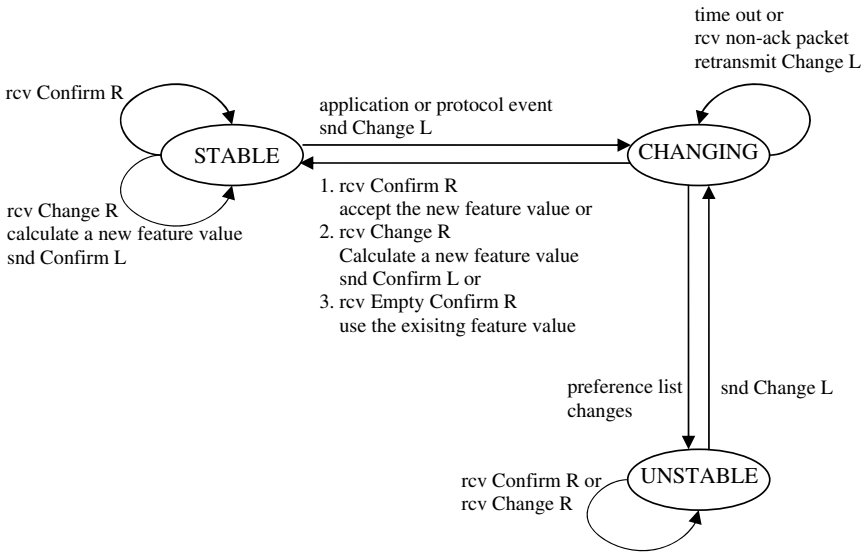

Fig. 3. DCCP feature negotiation state diagram - redrawn from [16].

When the preference list is changed by its user while the entity is in the CHANGING state, it enters the UNSTABLE state. Here it ignores the on-going negotiation but starts a new negotiation by sending a Change command with the new preference list before going back to the CHANGING state.

The state diagram for *feature remote* can be obtained by interchanging *L*s and *R*s in Fig. 3. Thus each entity consists of three state machines working together: connection management, feature local and feature remote. It is possible that one side initiates Change L while the other side initiates Change R of the same feature. According to Fig. 3 when the local in CHANGING receives Change R, it computes a new agree value and replies Confirm L. This situation is called *simultaneous negotiation*. The specification also allows the preferences to be changed at any time.

3.2 Important Rules of Feature Negotiation

Although the feature negotiation procedures explained in the previous section sound simple, the real situation could be very complex when packets are re-ordered and lost. Moreover the negotiation for the same feature could be simultaneously initiated by both sides and the preference lists can be changed at any time. To cope with this, the RFC specifies some rules intended to provide reliable signalling so that both sides reach agreement with the same feature value.

Non-reordered Change and Confirm Options The RFC specifies that the Change and Confirm options in packets that do not arrive in strictly increasing order must be ignored. According to the related pseudo code and algorithms, the strictly increasing order rule is only enforced for packets that contain the Change and Confirm options. An ordered packet with the Change and/or Confirm options may have a sequence number less than GSR if the later packets do not contain any Change or Confirm options.

In order to check the order of arrival, the RFC specifies another two variables: Feature Greatest Sequence Number Received (FGSR) and Feature Greatest Sequence Number Sent (FGSS). If the received packet's sequence number is less than or equal to FGSR, Change or Confirm options received must be ignored. If the acknowledgement Number is less than FGSS or the packet contains no acknowledgement, the Confirm option received must be ignored.

Because DCCP-Data with short sequence numbers is vulnerable to be attacked, any option attached to DCCP-Data that might cause the connection to be reset shall be ignore. *Thus both Change and Confirm options received in DCCP-Data must be ignored in all circumstances.* A sequence number valid packet received that contains non-reordered Change or Confirm options updates FGSR while FGSS is updated when the entity sends a Change Option during a transition from STABLE or UNSTABLE to CHANGING.

Retransmission Because the reordered options are ignored or the packet can be lost, Change options must be retransmitted when the sender does not receive a non-reordered Confirm option within a specific period. The Confirm option must be generated only when a non-reordered Change option is received. Retransmission of options may be achieved by either generating a new packet (DCCP-Ack or DCCP-Sync) or by including the appropriate option field in a packet that is about to be transmitted. Retransmission continues until a non-reordered Confirm option is received or the connection is closed down.

4 CPN Model of DCCP Feature Negotiation (DCCP-FN)

4.1 Modelling Assumptions and Specification Interpretation

We make the following assumptions regarding DCCP feature negotiation when creating our model.

1. This paper assumes the medium to be First-in First-out (FIFO) channels with no loss. There are three reasons supporting this assumption. Firstly, according to RFC 4340 the Change and Confirm Options must arrive in strictly increasing order otherwise it will be ignored. This requirement implies that actually DCCP feature negotiation protocol operates over FIFO channels. Secondly, reordered and/or lossy channels can mask out inherent errors such as unspecified receptions which could appear when protocol operates over FIFO channels with no loss. Thus protocol validation shall be started from operating over the

FIFO channels with no loss. Thirdly, the assumption of FIFO channel makes the model simpler. We can abstract away irrelevant details such as sequence number, acknowledgement number, state variables FGSS and FGSR.

2. Although we agree with [20] that the feature negotiation is not independent of the protocol state machine. To reduce the complexity of our CPN model, we assume that the feature negotiation is independent of the protocol state machine. Without loss of generality, instead of modelling three FSMs (connection management, feature local and feature remote) at each side, only one FSM (Fig. 3) (either the feature local's or the feature remote's FSM) is required. In particular we assign the feature local's FSM to Client and the feature remote's FSM to Server. This assumption makes the CPN model readable and easy to understand.

3. A DCCP packet is modelled by an option type and a list of feature values (preference list). Other fields such as packet type and sequence-acknowledgement numbers are omitted because they do not affect the operation of the feature negotiation.

4. RFC4340 allows many options to be sent in one packet and many features to be negotiated at the same time. Following an incremental approach [3], as a first step we consider the negotiation of *Congestion Control Identification* (CCID) that uses the server-priority reconciliation rule because the ability to negotiate the suitable congestion control mechanism is the main objective of DCCP.

5. Our model does not include the mandatory options, invalid options and unknown feature numbers.

6. RFC 4340 specifies that the preference list can be changed at any time. It is unclear what should be happened if the preference list is changed while the endpoint in STABLE. However according to [19], the endpoint can remain in STABLE if it changes the preference list without changing the preferred value. Thus we assume that the endpoint remains in STABLE after it changes the preference list. However we investigate the scenario when the endpoint changes the preference list without changing the preferred value.

4.2 Model Structure

Our model structure is inspired by [21, 22] who model and analyse DCCP connection management. However [21, 22] do not include the feature negotiation procedure. Our DCCP feature negotiation model comprises three hierarchical levels as shown in Fig. 4. The first level page is `Main_FN`. This page calls the second level pages named `FN_Local` and `FN_Remote`. The third level has six pages. Each one is named by a DCCP feature negotiation state. Figure 5 shows Global Declaration which defines the data associated with the model. The CPN diagram in the first level page (Fig 6) comprises two substitution transitions (represented

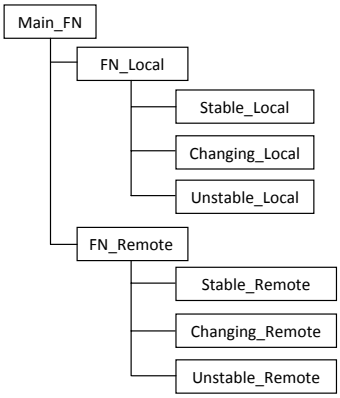

Fig. 4. The DCCP-FN hierarchy page.

```
(* Feature Negotiations *)
colset E = with e;
colset CCID = int with 2..255;
colset Confirmed_Value = CCID;
colset Preference_List = list CCID;
colset Option_Type = with ChangeL | ConfirmL
                   | ChangeR | ConfirmR;
colset Option_Field = product Option_Type
                    * Preference_List;
colset List_Option_Field = list Option_Field;
colset FN_State = with STABLE | CHANGING | UNSTABLE;
colset FN_CB = product FN_State * Confirmed_Value
                                * Preference_List;
```

Fig. 5. Global Declaration.

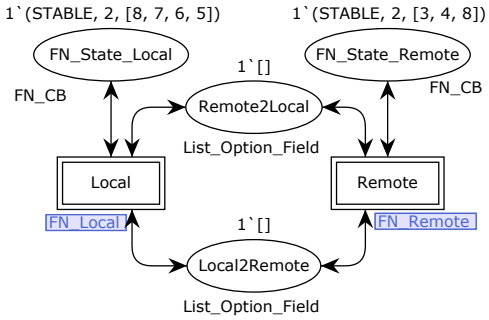

Fig. 6. The Main_FN overview page.

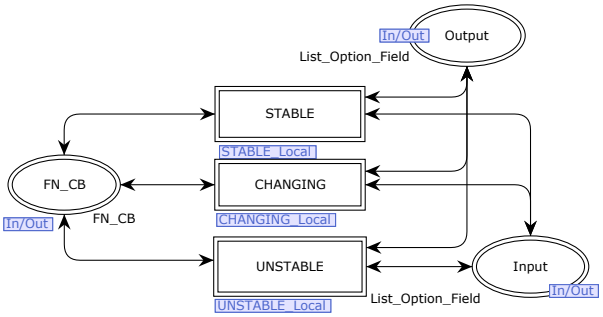

Fig. 7. The FN_Local page.

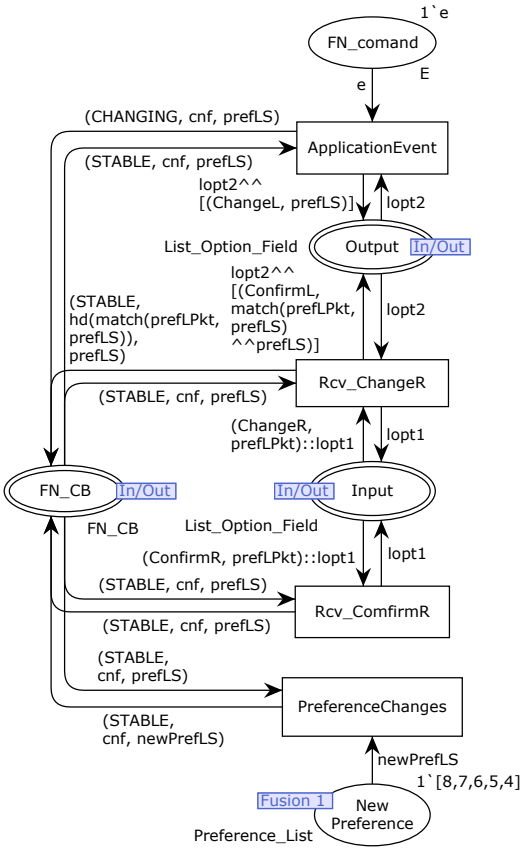

Fig. 8. The STABLE_Local page.

by double-line rectangles), four places (represented by ellipses) and arcs connecting between places and transitions. The substitution transition on the left models the Client (`Local`) and another on the right models the Server (`Remote`). Both communicate via two places named `Remote2Local` and `Local2Remote` in the middle of Fig. 6. Each place models a unidirectional and First-in First-out channel typed by `List_Option_Field`. `List_Option_Field` is a list of product sets named `Option_Field` defined in Fig. 5. `Option_Field` comprises `Option_Type` and `Preference_List` sets also defined in Fig. 5. Through these places, tokens (which are values taken from the type of the place) are transferred between `Local` and `Remote`.

Places `FN_State_Local` and `FN_State_Remote`, typed by `FN_CB`, model the states of the feature negotiation procedure. The `FN_CB` is defined as a product comprising colour sets `FN_State`, `Confirmed_Value` and `Preference_List`.

The substitution transitions `Local` and `Remote` in Fig. 6 are linked to the second level pages named `FN_Local` (Fig. 7) and `FN_Remote`. Each of the second level CPN diagrams comprises further three substitute transitions, named by the feature negotiation states (Fig. 7) and linked to the CPN diagrams in the third level. Because these CPN diagrams of `FN_Remote` are very similar to those of `FN_Local`, this paper illustrates only the CPN diagrams of `FN_Local`.

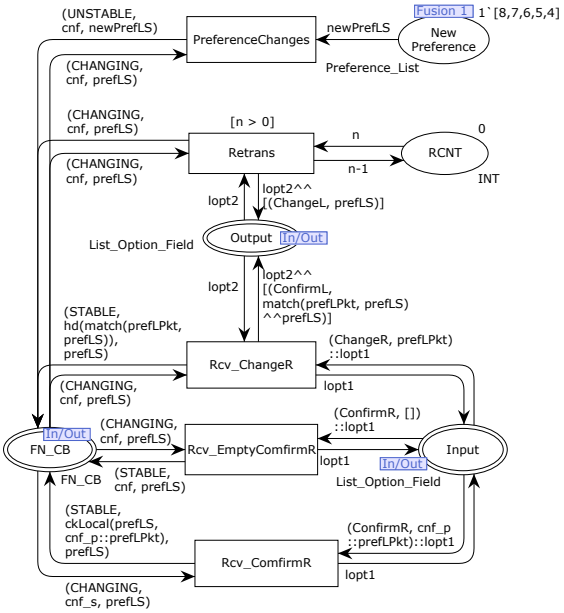

Fig. 9. The CHANGING_Local page.

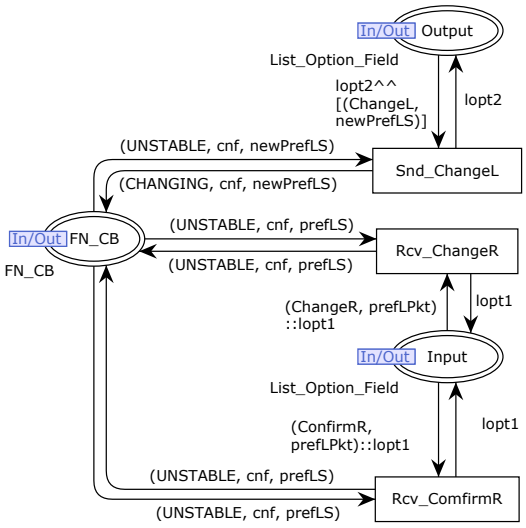

Fig. 10. The UNSTABLE_Local page.

Figure 8 captures behaviour when Local is in STABLE State. We allow DCCP entity in STABLE change its preference and remain in the STABLE state. Figure 9 and 10 model the procedures to be followed by Local when it is in CHANGING and UNSTABLE respectively.

5 Analysis of DCCP-FN CPN Model

5.1 Initial Configurations

Our DCCP feature negotiation model is analysed using CPN Tools [4,14] version 4.0 on an Intel i5-4300U 1.90GHZ with 4 GB RAM. To analyse a particular scenario, the CPN model needs to be initialised by distributing initial tokens to places `FN_State_Local` and `FN_State_Remote` (Fig. 6); places `FN_Command` and `NewPreference` in `Stable_Local` (Fig. 8) as well as places `FN_Command` and `NewPreference` in `Stable_Remote`. The channel places `Remote2Local` and `Local2Remote` initially contain an empty list. The presence of tokens 1‘e in place `FN_Command` allows the entity to start the feature negotiation procedure. The analysis in this paper assumes no retransmission.

We choose to model and analyse the negotiation of the feature CCID. This feature uses the reconciliation rule: server priority. The default feature value is 2 which represents TCP-like congestion control. Although currently the standard specifies only CCID2 (RFC4341), CCID3 (RFC4342) and CCID4 (RFC5622), we make up CCID numbers in each preference list for the purpose of validating the feature negotiation procedure. Table 1 shows the values in preference lists we

Table 1. An agreed feature value before and after preference lists have been changed.

			Client (Local)	
			before	after
			[8,7,6,5]	[8,7,6,5,4]
Server (Remote)	before	[3,4,8]	8	4
	after	[4,5]	5	4

Table 2. Initial configurations of twelve possible scenarios.

	FN_Command		Change of Preference List	
Case	Local	Remote	Local	Remote
1	1'e	empty	disable	disable
2	empty	1'e	disable	disable
3	1'e	1'e	disable	disable
4	1'e	empty	enable	disable
5	empty	1'e	enable	disable
6	1'e	1'e	enable	disable
7	1'e	empty	disable	enable
8	empty	1'e	disable	enable
9	1'e	1'e	disable	enable
10	1'e	empty	enable	enable
11	empty	1'e	enable	enable
12	1'e	1'e	enable	enable

used in our experiment before and after the preference has been changed. The resolved values before and after the preference changed under the server-priority reconciliation rule are shown in Table 1 as well. According to [19] the endpoint can remain in the STABLE state if it changes the preference list without changing the preferred value. Therefore at Client (Local) we keep the old preference list but adding the new feature value (4) at the end of the list.

Table 2 shows the initial configurations of twelve possible scenarios. They are classified according to which sides are allowed to initiate the negotiation and which sides change their preference lists. Our CPN model allows simultaneous negotiation and both sides can change their preference lists (Case 12).

5.2 Analysis Result

The analysis results of DCCP feature negotiation CPN model using the initial configurations described in the previous subsection are shown in Table 3. The total number of states, arcs in each case are shown in the second and third columns. Column 4, 5 and 6 show the terminal markings of each scenario. All terminal markings have both sides in STABLE and no packets left in the channels and hence there is no unspecified reception. The terminal markings are classified into 3 types. Type-I is the desired terminal state where both Client and Server reach the same feature value. Type-II is the undesired terminal state where both

Table 3. Analysis results of the CPN model

Case	nodes	arcs	Terminal Markings		
			Type I	Type II	Type III
(1)	(2)	(3)	(4)	(5)	(6)
1	4	3	1	0	0
2	4	3	1	0	0
3	20	26	1	0	0
4	19	22	2	1	0
5	10	11	2	0	0
6	106	169	2	1	0
7	10	11	2	0	0
8	19	22	2	1	0
9	106	169	2	1	0
10	50	77	3	1	0
11	52	78	3	2	0
12	553	1043	3	3	1

sides reach the different feature values but an endpoint knows that the agreed value is wrong. Type-III is also the the undesired terminal state where both sides reach the different feature values and both endpoints do not know that their feature values do not match.

5.3 Discussion

Figure 11 shows a scenario leading to a Type-II terminal state. Referring Fig. 11, after sending the first Change L Option, Client changes its preference list in UNSTABLE and sends the second Change L. When receiving the Confirm R of the first Change L in the CHANGING state, Client enters the STABLE state and then ignores the Confirm R of the second Change L. The agreed feature value in the first Confirm R is outdated and different from the feature value in Server. However when comparing the preference list in the first Confirm R option with the preference list in Client's state information, Client is able to know that the agreed value is wrong. Obviously in this case the Client should resend the Change option or reset the connection.

Figure 12 illustrates a scenario leading to an undesired Type-III terminal state. This is the center of our attention in this paper. This scenario could happen when both sides initiate the negotiation simultaneously and both sides change their preference list (Case 12). We notice that all Confirm options in Fig. 12 are discarded. It becomes one way communication with no acknowledgement. Figure 12 can be viewed as three attempts of negotiation. Two attempts are initiated simultaneously from both sides. This could be happened during DCCP simultaneous open procedure. Preference list changed in CHANGING

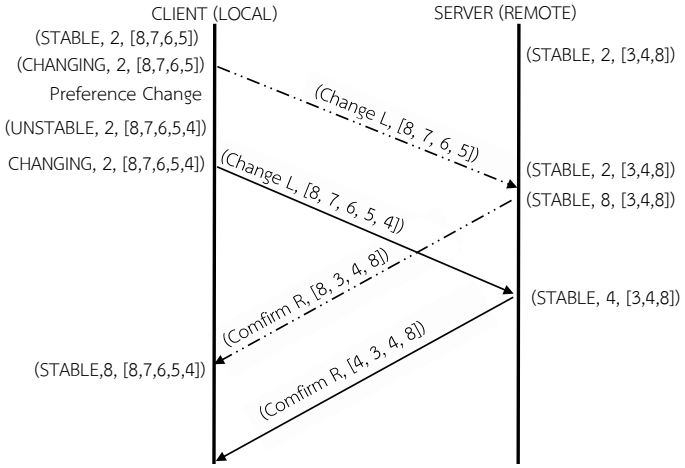

Fig. 11. A scenario leading to an undesired terminal marking Type-II.

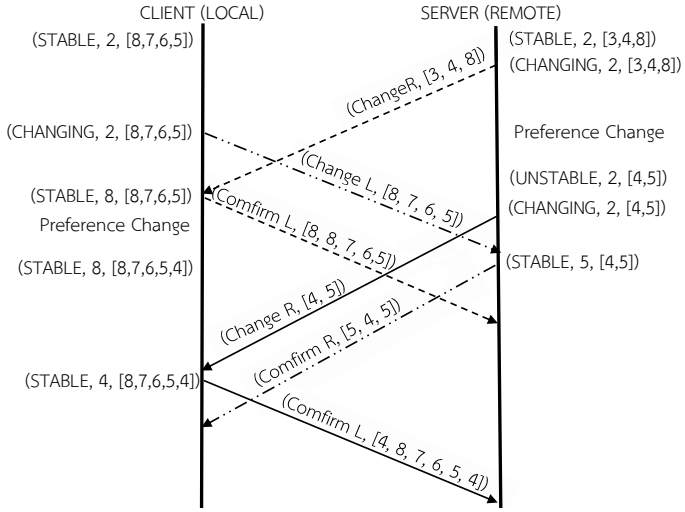

Fig. 12. A scenario leading to an undesired terminal marking Type-III.

state causes the third attempt of negotiation. All three calls do not receive any reply. The root of the problem is that the new preference list from the other side cannot pass through.

Type-III terminal state is worse than type-II because both entities are not aware that their agreed feature values are different. In our opinion the main objective of the DCCP feature negotiation protocol is to exchange the preference lists. After the preference list of the other side is known, the agreed feature value can be correctly computed. We suggest a solution when the preference list is

changed (either major or minor change), the endpoint shall send Change option to inform the other side. If the preference list is changed in the STABLE state, the endpoint shall send Change option and enter CHANGING state. Another solution could be that the endpoint does not discard Confirm option in STABLE state.

6 Conclusion and Future Work

This paper presents Coloured Petri Net model and analysis of DCCP feature negotiation procedure operating over FIFO with no loss channels. The analysis result shows that the protocol could fail to an undesired state (Type-III) where the feature values of both sides do not match and both sides are not aware of the mismatch.

Usually when the protocol operates over reordering and/or lossy channels, it is possible that the protocol could fail due to the channel imperfection. However if the protocol operates over the ideal channels (FIFO with no loss), the error indicates the flaw in the protocol itself.

The terminal state (Type-III) occurs when both sides change their preference lists during the simultaneous feature negotiation. Although the odds of this scenario is low, given the large number of potential connection in the Internet, we consider that this defect could be a serious threat.

The model development begins with a lot of assumptions. In the future we would like to relax these assumptions and refine the model. In particular we are interested to include connection management procedures together with Network Address Translators (NATs) into the model.

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

References

1. F. Babich and L. Deotto. Formal Methods for the Specification and Analysis of Communication Protocols . *IEEE Communications Surveys*, 4(1):2–20, Third Quarter 2002.
2. J. Billington, M. Diaz, and G. Rozenberg (Eds.). *Application of Petri Nets to Communication Networks*, volume 1605 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 1999.
3. 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.
4. CPN Tools home page. http://cpntools.org.

5. J. C. A. Figueiredo and L.M. Kristensen. Using Coloured Petri Nets to Investigate Behavioural and Performance Issues of TCP Protocols. In *Second Workshop and Tutorial on Practical Use of Coloured Petri Nets and Design/CPN*, DAIMI PB-541, pages 21–40. Department of Computer Science, University of Aarhus, 11-15 October 1999.
6. S. Floyd, M. Handley, and E. Kohler. Problem Statement for the Datagram Congestion Control Protocol (DCCP), RFC 4336. Available via http://www.rfc-editor.org/rfc/rfc4336.txt, March 2006.
7. S. Floyd and E. Kohler. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control, RFC 4341. Available via http://www.rfc-editor.org/rfc/rfc4341.txt, March 2006.
8. S. Floyd and E. Kohler. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 4: TCP-Friendly Rate Control for Small Packets (TFRC-SP), RFC 5622. Available via http://www.rfc-editor.org/rfc/rfc5622.txt, August 2009.
9. S. Floyd, E. Kohler, and J. Padhye. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC), RFC 4342. Available via http://www.rfc-editor.org/rfc/rfc4342.txt, March 2006.
10. S. Gordon. *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.
11. B. Han. *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.
12. K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts.* Monographs in Theoretical Computer Science. Springer, Heidelberg, 2nd edition, 1997.
13. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 3, Practical Use.* Monographs in Theoretical Computer Science. Springer, Heidelberg, 1997.
14. K. Jensen and L.M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems.* Springer, Heidelberg, 2009.
15. E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion Control Without Reliability. In *Proceedings of the 2006 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'06)*, pages 27–38, Pisa, Italy, 11-15 September 2006.
16. E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol, RFC 4340. Available via http://www.rfc-editor.org/rfc/rfc4340.txt, March 2006.
17. L. M. Kristensen, J.B. Jørgensen, and K. Jensen. Application of Coloured Petri Nets in System Development. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets - Advanced in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 626–685. Springer, Heidelberg, 2004.
18. C. Ouyang. *Formal Specification and Verification of the Internet Open Trading Protocol using Coloured Petri Nets.* PhD thesis, Computer Systems Engineering Centre, School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia, June 2004.
19. University of Aberdeen, Electronics Research Group, School of Engineering. Background on Feature Negotiation. Available via http://www.erg.abdn.ac.uk /users/gerrit/dccp/notes/feature_negotiation/background.html.

20. University of Aberdeen, Electronics Research Group, School of Engineering. Why feature negotiation and protocol state machine are not independent. Available via http://www.erg.abdn.ac.uk/users/gerrit/dccp/notes/feature_negotiation/dependencies.html.
21. S. Vanit-Anunchai. *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.
22. S. Vanit-Anunchai, J. Billington, and T. Kongprakaiwoot. Discovering Chatter and Incompleteness in the Datagram Congestion Control Protocol. In F. Wang, editor, *Proceedings of the 25th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2005)*, volume 3731 of *Lecture Notes in Computer Science*, pages 143–158, Taipei, Taiwan, 2-5 October 2005. Springer, Heidelberg.
23. Somsak Vanit-Anunchai. Analysis of Two-Layer Protocols: DCCP Simultaneous-Open and Hole Punching Procedures. In Christine Choppy and Jun Sun, editors, *1st French Singaporean Workshop on Formal Methods and Applications (FSFMA 2013)*, volume 31 of *OpenAccess Series in Informatics (OASIcs)*, pages 3–17, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.