World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:6, No:1, 2012

Modeling of Session Initiation Protocol Invite Transaction using Colored Petri Nets

Sabina Baraković, Dragan Jevtić, and Jasmina Baraković Husić

***Abstract*—**Wireless mobile communications have experienced the phenomenal growth through last decades. The advances in wireless mobile technologies have brought about a demand for high quality multimedia applications and services. For such applications and services to work, signaling protocol is required for establishing, maintaining and tearing down multimedia sessions. The Session Initiation Protocol (SIP) is an application layer signaling protocols, based on request/response transaction model. This paper considers SIP INVITE transaction over an unreliable medium, since it has been recently modified in Request for Comments (RFC) 6026. In order to help in assuring that the functional correctness of this modification is achieved, the SIP INVITE transaction is modeled and analyzed using Colored Petri Nets (CPNs). Based on the model analysis, it is concluded that the SIP INVITE transaction is free of livelocks and dead codes, and in the same time it has both desirable and undesirable deadlocks. Therefore, SIP INVITE transaction should be subjected for additional updates in order to eliminate undesirable deadlocks. In order to reduce the cost of implementation and maintenance of SIP, additional remodeling of the SIP INVITE transaction is recommended.

***Keywords*—**Colored Petri Nets, SIP INVITE, state space, dead marking

I. Introduction

WIRELESS mobile communications have experienced the phenomenal growth through last decades [1]. The first decade can be characterized by simple circuit-switched service voice, using Global System for Mobile (GSM) communications or Code Division Multiple Access (CDMA) standard, and rapid adoption of services based on Short Message Service (SMS). While the second decade has been driven by the initial adoption of Internet Protocol (IP)-based packet services, using low-rate General Packet Radio Service (GPRS) or CDMA2000 radio access networks, the next decade of evolution will see rapidly increasing of mobile broadband services, using High Speed Packet Access (HSPA), Worldwide Interoperability for Microwave Access (WiMAX), or Long-Term Evolution (LTE) radio access networks.

These advances in wireless mobile technologies have brought about a demand for high quality multimedia applications and services. The important issue is how the service quality can be maintained at a level similar to which users have come to expect. Different multimedia applications have very diverse Quality of Service (QoS) requirements. In a wireless environment, users are mobile and move between wireless technologies where the available resources are scarce and dynamically change over time. Under these conditions it is difficult to provide any QoS guarantees. The QoS topic therefore remains the most important issue to be dealt with in the design and development of multimedia applications and services.

For such applications and services to work, signaling protocol is required for establishing, maintaining and tearing down multimedia sessions. A number of different communities put forward solutions, each colored by their own priorities and interests. The Internet Engineering Task Force (IETF) offered Session Initiation Protocol (SIP) [2], which is based on request/response transaction model. Each transaction consists of a client request that invokes a particular method on the server and at least one response. Two main SIP transactions are defined in Request for Comments (RFC) 3261, the INVITE transaction for setting up a session, and the non-INVITE transaction for maintaining and tearing down a session.

In this paper, the INVITE transaction is chosen to be considered since it has been recently modified in RFC 6026 [3]. In order to help in assuring that the functional correctness of this modification is achieved, the INVITE transaction is modeled and analyzed using Colored Petri Nets (CPNs). The analysis of performance properties is beyond its scope. Since CPNs are successfully applied as the modeling and analyzing apparatus in many research areas, functional properties of INVITE transaction are investigated using well-developed software tool, the CPN Tools [4]. However, to our best knowledge, only a few papers on analyzing SIP using CPNs have been published [5, 6, 7, 8]. CPNs have been used for verification of the INVITE transaction when the medium is reliable [5] and unreliable [6].

The paper is organized as follows. In Section II basic principles of SIP protocol, and therefore, INVITE client and server transactions are introduced. Section III presents modeling methodologies and tools. CPN model of SIP INVITE transaction over the unreliable transport medium is described in Section IV, while the Section V gives the CPN model analysis. Section VI concludes this paper.

II. Session Initiation Protocol Overview

SIP is an application layer signaling and mobility support protocol that can establish, modify, and terminate multimedia

Sabina Baraković is with the Department for Informatics and Telecommunication Systems, Ministry of Security of Bosnia and Herzegovina, Trg BiH 1, 71000 Sarajevo, Bosnia and Herzegovina (e-mail: barakovic.sabina@gmail.com).

Dragan Jevtić is with the Faculty of Electrical Engineering and Computing, University in Zagreb, Unska 3, 10000 Zagreb, Croatia (e-mail: dragan.jevtic@fer.hr).

Jasmina Baraković Husić is with BH Telecom d.d Sarajevo, Joint Stock Company, Obala Kulina bana 8, 71000 Sarajevo, Bosnia and Herzegovina (e-mail: jasmina.barakovic@bhtelecom.ba).

sessions such as Internet telephony calls [2]. It is not a vertically integrated communications system, but rather a component that can be used with other IETF protocols to build a complete multimedia architecture. Therefore, SIP should be used in conjunction with other protocols, such as the Real-time Transport Protocol (RTP), the Real-time Streaming Protocol (RTSP), the Media Gateway Control Protocol (MEGACO), and the Session Description Protocol (SDP), in order to provide complete services to the users, although its basic functionality and operation does not depend on any of these protocols.

SIP is structured as a four-layer protocol, which means that its behavior is decoupled in terms of a set of fairly independent processing stages with only a loose coupling between each stage (Fig. 1) [9]. The lowest layer of SIP is its syntax and encoding. Its encoding is specified using an augmented Backus-Naur Form (BNF) grammar. The second layer from bottom to top of the structure is the transport layer, and it is contained by all SIP elements. This layer describes how a client sends requests and receives responses, and how a server receives requests and sends responses over the network. The layer above the transport layer is the transaction layer, which handles application-layer retransmissions, matching of responses to requests, and application-layer timeouts when setting up and tearing down a session. On top of transaction layer is a layer called transaction user (TU). The fourth layer creates and cancels SIP transactions, and utilizes services provided by the transaction layer.

Among all SIP layers, the transaction layer is the most important, since SIP is a transaction-oriented protocol that carries out tasks through different transactions. Specifically, a SIP transaction consists of a single SIP request message and any SIP response messages to that request, which include zero or more provisional SIP response messages, and one or more final SIP response messages (Table I). Transactions have a client side and server side. The client side is known as a client transaction and the server side as a server transaction [2]. The client transaction sends the request and the server transaction sends the response. The purpose of the client transaction is to receive a request from the element in which the client is embedded, and reliably deliver the request to a server transaction. The purpose of the server transaction is to receive requests from the transport layer and deliver them to the TU and also, to accept responses from the TU and deliver them to the transport layer for transmission over the network. There are two types of client transactions, depending on the method of the request passed by TU. One that handles INVITE requests is an INVITE client transaction, and another type, which handles all requests except INVITE and ACK, is non-INVITE client transaction. As with the client transactions, we distinguish INVITE and non-INVITE server transactions.

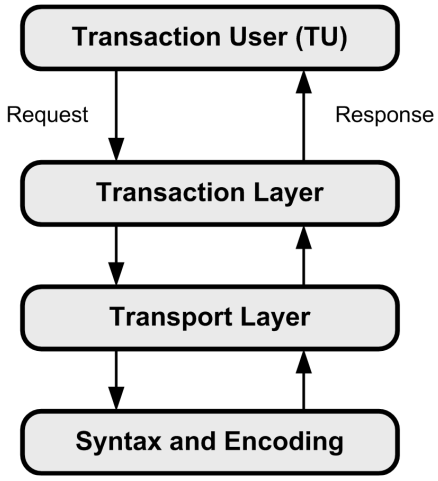

Fig. 1 Layered SIP structure

Legend: SIP (Session Initiation Protocol).

TABLE I
SIP RESPONSE MESSAGES

RESPONSE	FUNCTION
1xx	Provisional – The request was received, but not yet accepted
2xx	Success – The request was received successfully and accepted
3xx	Redirection – A further action is required to complete the request
4xx	Client Error – Bad syntax found in the request
5xx	Server Error – The server failed to answer the request
6xx	Global Failure – No server can answer the request

Legend: SIP (Session Initiation Protocol)

A. INVITE Client Transaction

The INVITE client transaction is defined in [2] using state machines and its modifications are presented in [3]. It is created by TU at the client side. The transaction user creates the INVITE client transaction when it wants to initiate a session. Then it forwards an INVITE request to the transaction. As shown on Fig. 2(a) the INVITE client transaction can enter five different states: (1) *Calling*, (2) *Proceeding*, (3) *Accepted*, (4), *Completed*, and (5) *Terminated*.

The initial *Calling* state is entered when the INVITE client transaction is created. The transaction delivers the INVITE request generated by its TU to the transport layer for transmitting to the server side, and starts Timer B (it controls transaction timeouts). Timer A is started only if an unreliable transport is being used (it controls request retransmissions). When the INVITE client transaction is in the initial *Calling* state, it can cause the occurrence of six events. Firstly, Timer A may fire, forcing the transaction to reset the timer and retransmit the INVITE request. Then, Timer B can fire, thereby causing the transaction to change its initial state to the *Terminated* state. When transport layer fails to send an INVITE request over the network, i.e. transport error occurs, the transaction enters the *Terminated* state and informs its TU. In case of receiving a provisional response (1xx), the transaction forwards the response to its TU and enters the

Proceeding state [6]. When a final success response (2xx) is received, i.e. the INVITE request is accepted by the server, the transaction informs its TU about the response and enters the *Accepted* state. On the other hand, when a final non-success response (300-699) is received, i.e. the INVITE request is received, but not accepted by the server, the transaction forwards the response to its TU, creates an ACK, gives the ACK to the transport layer, and finally enters the *Completed* state.

When the transaction is in the *Proceeding* state it can receive any number of provisional responses (1xx), inform its TU about the response and remain in the *Proceeding* state. Also, it can receive a final success response (2xx), forward it to its TU and enter *Accepted* state, or receive a final non-success response (300-699), pass through the previously mentioned procedure and enter the *Completed* state.

The purpose of the *Accepted* state, which presents the correction of INVITE client transaction according to RFC 6026, is to allow the client transaction to continue to exist to receive and pass to its TU any retransmissions of the 2xx response. When this state is entered, Timer M must be started. This timer reflects the amount of time that the TU will wait for retransmissions of the 2xx responses [3]. When Timer M fires, transaction enters the *Terminated* state.

On the other hand, the purpose of the *Completed* state is to soak up 300-699 responses retransmitted by the server. When the transaction enters this state, Timer D must be started. This timer reflects the wait time for 300-699 response retransmissions. Before Timer D expires (the client transaction enters the *Terminated* state), if any 300-699 response is received, the transaction creates and sends an ACK and remains in the *Completed* state. Also, in case of a transport error while the transport layer is sending an ACK, the transaction enters the *Terminated* state.

Finally, the instant the client transaction enters the *Terminated* state, it must be destroyed in order to guarantee correct operation [2].

B. INVITE Server Transaction

The INVITE server transaction is defined in [2] using state machines, and its modifications are presented in [3]. It is created by TU on the server side when it receives an INVITE request. The INVITE server transaction generates a Trying (100) response unless it knows that the TU will generate a provisional or final response within 200 ms. This provisional response is needed to quench request retransmissions rapidly in order to avoid network congestion. As shown on Fig. 2(b) the INVITE server transaction can enter five different states: (1) *Proceeding*, (2) *Accepted*, (3) *Completed*, (4) *Confirmed*, and (5) *Terminated*.

Initially, the INVITE server transaction enters the *Proceeding* state when it is created. While in the *Proceeding* state, several different events can occur. The transaction can forward any provisional responses (101-199) generated by its TU to the transport layer and remain in the *Proceeding* state. Additionally, the server transaction remains in the *Proceeding* state if it receives an INVITE request retransmitted by the client transaction. In that case, the server transaction retransmits the provisional response that it previously received from its TU. When transport layer fails to send a response, i.e. transport error occurs, the server transaction remains in the *Proceeding* state and informs its TU. When the TU on the server side forwards a final success response (2xx) to the server transaction, the transaction delivers this response to the transport layer for transmission and enters the *Accepted* state. Retransmissions of the 2xx response are handled by TU, not by the server transaction. On the other hand, when the TU on the server side forwards a final non-success response (300-699) to the server transaction, the response is delivered to the transport layer for transmission and the server transaction enters the *Completed* state.

The purpose of the *Accepted* state, which presents the modification of INVITE server transaction according to RFC 6026, is to absorb retransmissions of an accepted INVITE request. Any such retransmissions are absorbed entirely within the server transaction. While in this state, if TU forwards a 2xx response, the server transaction must deliver the response to the transport layer for transmission. Any ACKs received from the network while in the *Accepted* state are forwarded directly to the TU and not absorbed. Timer L is started when the *Accepted* state is entered. This timer reflects the wait time for retransmissions of 2xx responses [3]. When Timer L fires, transaction enters the *Terminated* state.

Once the transaction enters the *Completed* state, Timer H is started. This timer determines when the server transaction abandons retransmitting the response [2], and when it expires the server transaction enters the *Terminated* state. Also, if the transport media is unreliable, Timer G is started in order to control the time for each retransmission of the 300-699 response it previously received from its TU while in the *Proceeding* state. While in this state, if an INVITE request retransmission is received, the transaction delivers the response to the transport layer for retransmission. Otherwise, if an ACK is received, the transaction enters the *Confirmed* state.

The purpose of the *Confirmed* state is to absorb any additional ACK messages that arrive, triggered from retransmissions of the final response. When this state is entered, Timer I is started. Once timer I fires, the server transaction enters the *Terminated* state [2].

The INVITE server transaction must not discard transaction state based only on encountering a non-recoverable transport error when sending a response. Instead, the associated INVITE server transaction state machine remains in its current state. This allows retransmissions of the INVITE to be absorbed instead of being processed as a new request and presents additional modification [3].

Finally, once the server transaction enters the *Terminated* state, it is destroyed by the TU immediately [2].

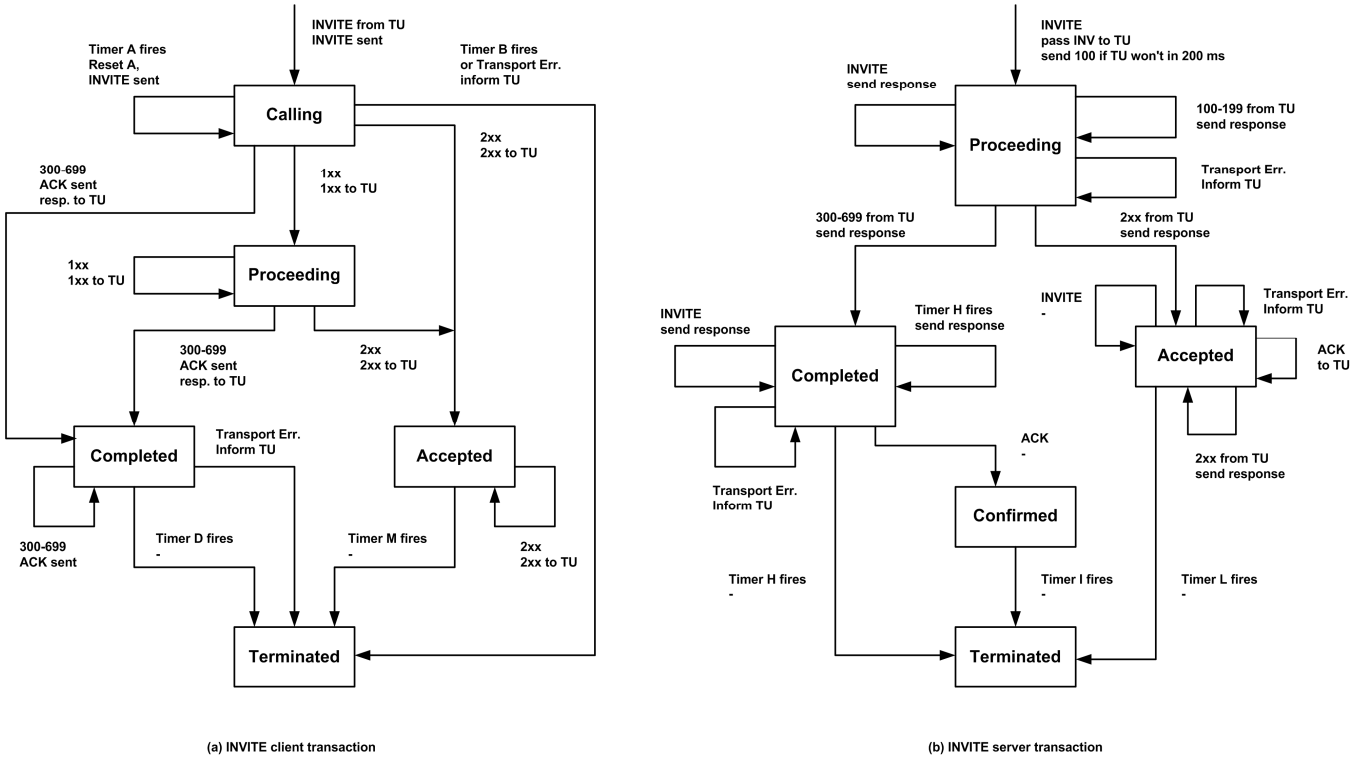

Fig. 2 State machines defining SIP INVITE transaction according to RFC 6026

Legend: ACK (Acknowledgment), RFC (Request for Comments), SIP (Session Initiation Protocol), TU (Transaction User).

III. Modeling Methodologies and Tools

Modeling always precedes system implementation, because it provides visualization of an entire system, assessment to different options, and communication with designs more clearly before taking on the risks of actual construction. As a protocol, SIP can be modeled in two ways. The first method models the protocol itself, and thereby, focuses on call flowing, states while running and timer mechanisms. The second method puts a protocol into a network environment and tests the whole network, interactions between different protocols and evaluates performances [8].

Since this paper aims to assure the functional correctness of the modified SIP INVITE transaction, i.e. the protocol itself, Colored Petri Nets are chosen as a suitable modeling methodology and CPN Tools [4] as an adequate supporting tool suite.

A. Petri Nets

Petri Nets are presented by Carl Adam Petri during his Ph.D. thesis in 1962 [10]. A Petri Net is a graphical and mathematical tool to verify systems and protocols. In a mathematical form Petri Net is like algebra and logic subject, while in graphical form it is like flowchart and network diagram. A formal definition of Petri Net is:

Definition 1:

In a formal way, A Petri Net is a tuple [11]:

$$PN = (P, T, A, N). \quad (1)$$

In the tuple,

1. P is a finite set of Places.
2. T is a finite set of Transitions.
3. A is a finite set of Arcs such that:
$$P \cap T = P \cap A = T \cap A = \varnothing. \quad (2)$$
4. N is a set of Token.

There are two forms of Petri Nets: ordinary Petri Nets and high level Petri Nets.

B. Colored Petri Nets

Colored Petri Nets belong to high level Petri Nets, i.e. they combine the graphical components of ordinary Petri Nets with the strengths of a high level programming language, making them suitable for modeling complex systems, such as distributed and concurrent processes with both synchronous and asynchronous communication. A formal definition of Colored Petri Net (CPN) is:

Definition 2:

In a formal way, A CPN is a tuple [11]:

$$CPN = (\Sigma, P, T, A, N, C, G, E, I). \quad (3)$$

In the tuple,

1. $\sum$ is a finite set of non-empty types, also called colored sets.
2. P is a finite set of Places.
3. T is a finite set of Transitions.
4. A is a finite set of Arcs such that:
$$P \cap T = P \cap A = T \cap A = \varnothing. \quad (4)$$
5. N is a node function. It is defined from A into "colored

over arcs" $P \times T \cup T \cap P$.

6. C is a color function. It is defined from P into $\sum$"token".
7. G is a guard function. It is defined from T into expressions such that: "Boolean function with probability"

$$\forall t \in T : [Type(G(t)) = B \wedge Type(Var(G(t))) \subseteq \Sigma]. \quad (5)$$

8. E is an arc expression function. It is defined from A into expressions such that: i.e. (check $k = n$)

$$\forall a \in A : [Type(E(a)) = C(p)_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma] \quad (6)$$

, where P is the place of $N(a)$.

9. I is an initialization function. It is defined from P into closed expression such that

$$\forall p \in P : [Type(I(p)) = C(p)_{MS}]. \quad (7)$$

C. CPN Tools

CPN Tools is a tool suite for editing, simulation, state space analysis, and performance analysis of CPN models [12]. A CPN model of a system is an executable model representing the states of the system and the events that can cause the system to change state. In other words, CPN model is a resulting model of combining Petri Nets, which provide the graphical notation and basic primitives, and previously mentioned high level programming language Standard ML, which provides the primitives for the definition of data types, describing data manipulation, and for creating compact models.

Regarding the graphical notation, the states of the system are presented with nodes called places (ellipses or circles), while the events are presented with the nodes called transitions (rectangular boxes). In order to constitute a net structure, places and transitions must be connected with a number of directed arcs. The CPN model contains textual inscriptions next to the places, transitions and arcs. The inscriptions are written in CPN ML programming language which is an extension of the Standard ML language. Each place is marked with one or more tokens, which have a data value attached to it. This data value is called token color. When the system performs an action, appropriate transition has to fire. When firing, the transition removes tokens from its input places (the places that have an arc directed from place to transition) and adds those tokens to its output places (the places that have an arc directed from transition to place). The colors of the tokens that are removed and added are determined on arc inscription basis.

The CPN Tools is used because it is possible to perform investigation of modeled system design and behavior on a simple manner. User interaction with this tool is based on direct manipulation of the graphical representation of the CPN model using interaction techniques, while the functionality of the tool can be extended with user-defined Standard ML functions [12].

IV. CPN Model Of SIP INVITE Transaction

A. Modeling Assumptions

Since the modeling of SIP INVITE transaction is not a straight translation from state machines to CPN model, the following assumptions must be made:

- State machine for the INVITE client transaction given on Fig. 2 shows that the transaction receives an INVITE from TU and sends it to the transport layer before it enters the *Calling* state. Since the transaction cannot enter any state before it is created, it is assumed that the transaction can receive an INVITE from TU and send it to the transport layer only when it has been created and is in the *Calling* state.
- State machine for the INVITE server transaction given on Fig. 2 shows that the transaction receives an INVITE request, forwards it to TU, and must generate and send 100 Trying response within 200 ms if TU does not before it enters the *Proceeding* state. It is assumed that the server transaction does not know will TU generate a response within 200 ms, and therefore, a new state, called the *TryProceeding*, must be denoted. The server transaction enters this state immediately after it is created, and it can only send 100 Trying response while in this state.
- When an unreliable medium is used as in this paper, messages may be reordered, and thus, 1xx responses may arrive at the client side after 2xx and 3xx responses. It is assumed that when this situation occurs, the client transaction stays in the correspondent state.
- Modeling assumptions related to timers A, B, D, G, H and I and relations between them are the same as in [6].
- Unreliable transport medium is modeled as in [6].

While the previous assumptions are the same as in [6], the following two are new and in accordance with the modifications of the INVITE transaction made in [3]:

- According to the state machines shown on Fig. 2, when the server transaction is in the *Accepted* state, Timer L can fire. However, since the purpose of the *Accepted* state is to absorb any retransmissions of the INVITE requests, it is assumed that the Timer L can fire only if there are no additional INVITE requests on transport layer.
- Additionally, according to the state machines shown on Fig. 2, when the client transaction is in the *Accepted* state, Timer M can fire. Also, since the purpose of the *Accepted* state is to absorb any retransmissions of 2xx responses, it is assumed that the Timer M can fire only if there are no additional 2xx or r1xx responses on the transport layer.

B. CPN Model of the SIP INVITE Client Transaction

Before describing the CPN model of the modified SIP INVITE client transaction shown on the Fig. 3, it is necessary to point out that the model presents the update of the SIP INVITE client transaction model in [6] in accordance with modifications of the SIP INVITE transaction in given in [3].

The INVITE client transaction is modeled with places named Client and No.INVITESent, together with the transitions connected to them. The place Client is typed with color set STATECLIENT and its initial marking is callingC (Table II). It

models the states of the INVITE client transaction. The place No.INVITESent is typed with color set INT. It is used to count the number of INVITE requests that have been transmitted and retransmitted.

There are six transitions connected to the place Client: Send Request, Receive Response, Timer A or B, Timer D, Timer M and Client Transport Error. The transition Send Request models how the transaction passes the original INVITE request to the transport layer for transmission. This transition is enabled only when the Client is marked with callingC and no INVITE request has been sent. When this transition fires, the place Client remains marked with callingC.

The transition Receive Response models how the client transaction receives responses and sends ACKs to the transport layer. It is enabled only when a response is received and the place Client is not marked with terminatedC. When r100 or r101 response is received, the place Client changes its marking to proceedingC and no ACK is sent to the transport layer. Else, if r2xx response is received, the place Client changes its marking to acceptedC and, also, no ACK is sent to the transport layer. Otherwise, if r3xx response is received, the place Client changes its marking to completedC and an ACK is sent to the transport layer. The arc from the transition Receive Response to the place Client models the assumption made on reordering medium.

The transition Timer A or B models Timer A and Timer B. It is enabled when the place Client is marked with callingC and an initial INVITE request is sent (No.INVITESent contains number equal or greater then 1). The initial marking of the place No.INVITESent is 0. When Send Request or Timer A or B fires, the integer value in No.INVITESent is incremented by 1. According to the assumptions made in [6], Timer B can't fire until Timer A fires 6 times, i.e. the INVITE request is sent 7 times. Therefore, when an integer value in the place No.INVITESent is less than 7, Timer A can fire and the marking of the place Client isn't changed, but the INVITE request is sent to the transport layer. Else, Timer B occurs and the marking of the place Client is changed to terminatedC, but no INVITE request is passed to the transport layer.

The transition Timer D is enabled only when the place Client is marked with completedC. The occurrence of this transition changes the marking of the place Client to terminatedC.

The transition Timer M is enabled only when the place Client is marked with acceptedC and there are no r2xx or r1xx responses left in the place Response. The second condition for occurrence is modeled using the anti-place of restricted model of the SIP INVITE transaction, which counts the number of responses in the place Response. The restriction is modeled as in [6]. These restrictions are made to avoid state space explosion and losing generality.

The Client Transport Error transition is enabled when the Client is marked with callingC or completedC. Its occurrence changes the Client's marking to terminatedC. When an error occurs on the transport layer, the INVITE or ACK that has been passed to the transport layer, based on the marking of the place Client, are destroyed, and therefore, not sent to the server side.

C. CPN Model of the SIP INVITE Server Transaction

The INVITE server transaction is modeled with places Server and No.r3xx, together with transitions connected to them. The place Server is typed by color STATESERVER and its initial marking is Idle. It models the states of the INVITE server transaction. The place No.r3xx is typed with color set INT. It is used to count the number of r3xx retransmitted responses when Timer G fires.

There are six transitions connected to the place Server: Receive Request, Send Response, Timer G or H, Timer I, Timer L, and Server Transport Error. The transition Receive Request models how the server transaction receives the INVITE or ACK requests. It is enabled when the place Server isn't marked with terminatedS or TryProceeding. When this transition occurs upon receiving an INVITE request and when the place Server is marked with Idle, the place Server changes it's marking to TryProceeding. In this case, the Receive Request models the operation of the TU instead of the server transaction of receiving an INVITE request from the client side. Otherwise, when the place Server is marked with proceedingS, acceptedS or completedS, and receives an INVITE request retransmitted by the client, the occurrence of the transition Receive Request results in sending the r101, r2xx or r3xx, respectively. Also, another situation when this transition models the operation of the TU is when it receives a retransmitted INVITE request while the place Server is marked with acceptedS. When this occurs, the transition Receive Request puts r2xx in the place Response, leaving the place Server marked with acceptedS. Additionally, while the place Server is marked with completedS, if an ACK is received, the occurrence of Receive Request changes the Server marking to confirmedS. In any other situation, the transition Receive Request doesn't change the marking of the place Server.

The transition Send Response models how the server transaction sends the response. It is enabled when the place Server is marked with TryProceeding and proceedingS. When this transition occurs, r101, r2xx or r3xx response is put into the place Response, which is presented with the inscription of the arc from the Send Response to the place Response. The marking of the place Server is then changed to proceedingS, acceptedS or completedS, respectively.

The transaction Timer G or H models Timer G and Timer H. It is enabled when the place Server is marked with completedS. According to the assumptions made in [6], Timer H can't fire until Timer G fires 10 times, i.e. the r3xx response is sent 10 times. Therefore, when an integer value in the place No.r3xx is less than 10, Timer G can fire and the marking of the place Server isn't changed, but the r3xx response is sent to the transport layer. Else,

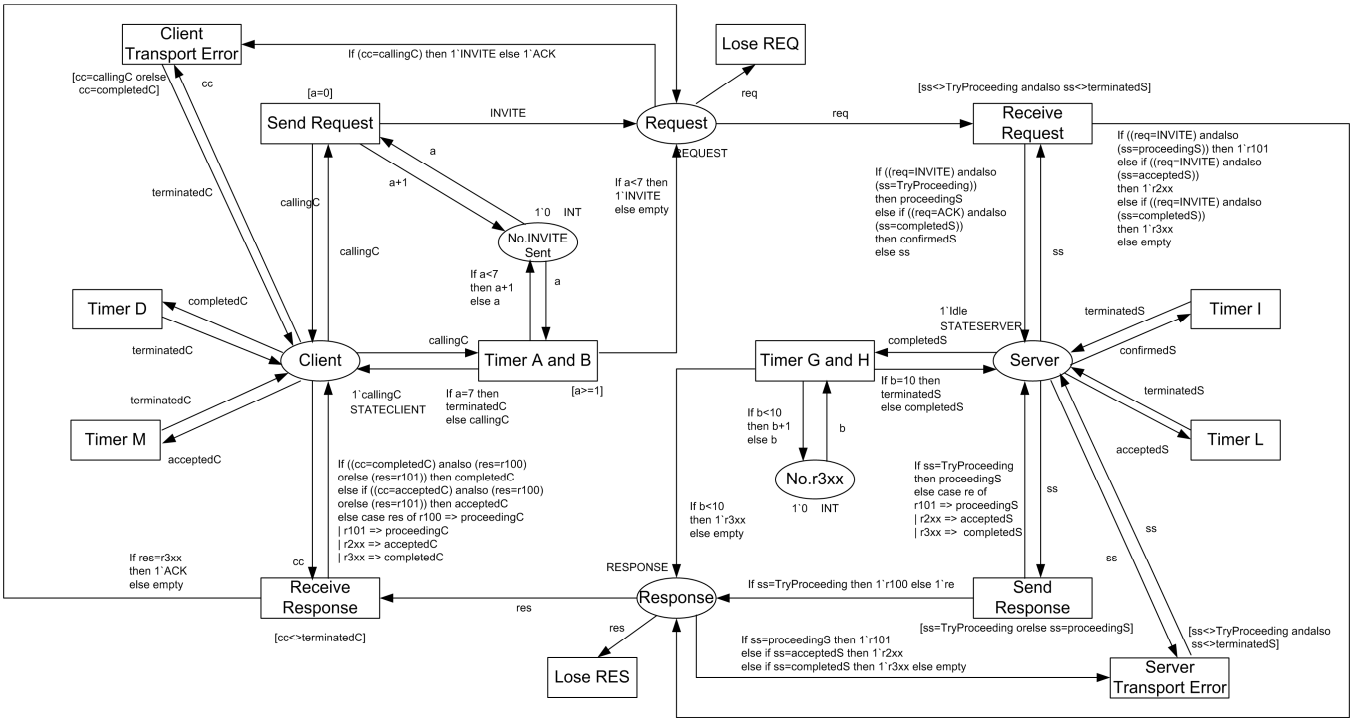

Fig. 3 CPN model of modified SIP INVITE transaction

Legend: CPN (Colored Petri Net), SIP (Session Initiation Protocol)

TABLE II
DECLARATIONS OF CPN MODEL

```
colset STATECLIENT = with callingC | proceedingC | acceptedC |
completedC | terminatedC;
colset STATESERVER = with Idle | TryProceeding | proceedingS |
acceptedS | completedS | confirmedS | terminatedS;
colset REQUEST = with INVITE | ACK;
colset RESPONSE = with r100 | r101 | r2xx | r3xx;
colset Response = subset RESPONSE with [r101 | r2xx | r3xx];
colset INT = int with 0..10;
var cc: STATECLIENT;
var ss: STATESERVER;
var req: REQUEST;
var res: RESPONSE;
var re: Response;
var a,b: INT;
```

Legend: CPN (Colored Petri Net)

Timer H occurs and the marking of the place Server is changed to terminatedS, but no r3xx response is passed to the transport layer.

The transition Timer I is enabled only when the place Server is marked with the confirmedS. The occurrence of this transition changes the marking of the place Server to terminatedS.

The transition Timer L is enabled only when the place Server is marked with accepted S and there are no INVITE requests left in the place Request. The second condition for occurrence is modeled using the anti-place of restricted model of SIP INVITE transaction, which counts the number of requests in the place Request. The restriction is modeled as in [6]. As previously mentioned, these restrictions are made to avoid state space explosion and losing generality.

The Server Transport Error transition is enabled when the Server is marked with proceedings, acceptedS or completedS. Its occurrence does not change the Server marking to terminated, according to [3]. When an error occurs on the transport layer, the response that has just been put into Response is removed, and therefore, not sent to the server side.

V. CPN MODEL ANALYSIS

The analysis of the CPN model of the SIP INVITE transaction includes investigation of properties such as absence of deadlocks, absence of livelocks, and absence of dead codes. A deadlock is an undesirable terminal state of a system [6]. In terms of CPN model of the system, deadlocks appear as dead markings in the state space of the model. A marking of a CPN model is dead if no transitions are enabled in it [6]. An undesirable terminal state for the SIP INVITE transaction must have either a client or the server transaction in a state which differs from *Terminated* state. However, besides the undesirable, there is desirable terminal state for the SIP INVITE transaction. This state must have both, the client and the server transactions in *Terminated* state, and ideally no messages left on the transport layer, i.e. in places Request and Response. A livelock is a cycle of the state space that once entered can never be left, and within which no progress is made in respect to the purpose of the system [6]. In terms of CPN model of the system, the absence of livelocks is given with the equal number of nodes and arcs in Strongly Connected Components (SCC) graph and in the state space. The absence of dead codes presents absence of actions that are

specified, but never executed. In terms of CPN model of a system a dead code is shown as a dead transition of the model.

In order to investigate previously mentioned properties, the state space analysis method of CPNs [12] with the support of the CPN Tools [4] is used. This paper analyses the CPN model which is restricted using the same principles as in [6]. Also, it is assumed that the unreliable medium may only reorder messages and no messages are lost, which correspondents to analysis of the restricted model without transitions Lose REQ and Lose RES. Analyzing the restricted model with these transitions is not performed, because behavior of the medium may mask the problems of transaction itself.

The state space report generated by CPN Tools shows that full state space has 145240 nodes and 410455 arcs (Table III). The number of nodes and arcs contained in the SCC graph is the same as in the state space, which implies that the SIP INVITE transaction has no livelocks. Also, the report shows no dead transitions, which implies that the SIP INVITE transaction has no dead codes. However, the state space report shows there are 18914 dead markings. Only 0.66 % of all dead markings represent the undesirable terminal states, i.e. the client or the server transaction is not in *Terminated* state. Among those 125 undesirable terminal states, 124 of them are the states in which the client is in the *Proceeding* state, and the server is in the *Terminated* state. This deadlock is caused by transport error at the server side, and expiration of Timer L and Timer H. It is undesirable because when the server transaction is destroyed, no responses can be received by the client transaction, thus it cannot exit the *Proceeding* state. Only one dead marking represents the situation in which the client transaction is in *Terminated* state, and the server transaction is in *Idle* state, which is caused by the transport error at the client side. Among all dead markings, 99.33 % represent the desirable terminal states, i.e. the client and the server transactions are in *Terminated* state. Among 18789 desirable terminal states, only 1.78 % is ideally terminal states with no messages left in the communication channel. This small amount is the consequence of transport errors on both sides, and situations when final non-success responses are sent.

TABLE III
STATE SPACE REPORT FOR THE RESTRICTED CPN MODEL OF SIP INVITE TRANSACTION

Occurrence Graph	Nodes: 145240 Arcs: 410455 Secs:1859
SCC Graph	Nodes: 145240 Arcs: 410455 Secs:12
Dead Markings	18914 [99992,99989,99987,99985, 99982,...]
Dead Transitions Instances	None

Legend: CPN (Colored Petri Net), SCC (Strongly Connected Components), SIP (Session Initiation Protocol).

VI. CONCLUSION AND FUTURE WORK

In this paper, SIP INVITE transaction over an unreliable medium is modeled and analyzed, since it has been recently modified in RFC 6026. Colored Petri Nets are chosen as an appropriate modeling methodology. After creating a CPN model of the SIP INVITE transaction, the analysis is carried using a restricted CPN model which was more suitable for investigating the most scenarios. Based on the model analysis, it is concluded that the SIP INVITE transaction is free of livelocks and dead codes, and in the same time it has both desirable and undesirable deadlocks. Therefore, SIP INVITE transaction should be subjected for additional updates in order to eliminate undesirable deadlocks.

Modeling and analyzing SIP specification using formal methods can help in assuring correctness, unambiguity, and clarity of the SIP protocol. Since a well-defined and verified protocol specification can reduce the cost for its implementation and maintenance, modeling and analysis are important steps of the protocol development life-cycle from the point view of protocol engineering. Therefore, the need for additional SIP protocol verification is identified. Finally, the intention is to remodel future modifications of the SIP INVITE transaction and perform verification using Timed CPNs.

ACKNOWLEDGMENT

This work was carried out within research project 036-0362027-1640 "Knowledge-based network and service management", supported by the Ministry of Science, Education and Sports of the Republic of Croatia.

REFERENCES

[1] M. Grayson, K. Shatzkamer, and K. Wierenga, *Building the Mobile Internet*. Cisco Press, 2011.
[2] J. Rosenberg, et al., "SIP: Session Initiation Protocol," Technical Report RFC 3261, Internet Engineering Task Force (IETF), June 2002.
[3] R. Sparks and T. Zourzouvillys, "Correct Transaction Handling for 2xx Responses to Session Initiation Protocol INVITE Request," Technical Report RFC 6026, Internet Engineering Task Force (IETF), Sep. 2010.
[4] Home Page of the CPN Tools, http://cpntools.org/. Accessed on Sep 20th, 2011.
[5] L. G. Ding and L. Liu, "Modelling and Analysis of the INVITE Transaction of the Session Initiation Protocol Using Coloured Petri Nets," *Lecture Notes in Computer Science*, Springer, 2008, pp. 132-151.
[6] L. Liu, "Verification of the SIP Transaction Using Coloured Petri Nets," in *Proc. of the 32nd Australasian Computer Science Conference*, Wellington, New Zealand, Jan. 2009.
[7] S. Kızmaz and M. Kırcı, "Verification of Session Initiation Protocol Using Timed Colored Petri Net," *International Journal of Communication, Network and System Sciences*, vol. 4, pp. 170-179, Mar. 2011.
[8] Y. Ding, G. Su, and H. Wan, "SIP Modeling and Simulation," *SIP Handbook: Services, Technologies, and Security of Session Initiation Protocol*, CRC Press, Taylor & Francis Group, USA, 2009, pp. 373-396.
[9] J. I. Agbinya, *IP Communications and Services for NGN*. Aurebach Publications, CRC Press, Taylor & Francis Group, USA, 2010.
[10] V. Gehlot and C. Nigro, "An Introduction to Systems Modeling and Simulation with Colored Petri Nets," in *Proc. of Winter Simulation Conference*, Baltimore, Maryland, USA, Dec. 2010, pp. 104-118.
[11] Y. Xu and X. Xie, "Modeling and Analysis of Security Protocols Using Colored Petri Nets," *Journal of Computers*, vol. 6, no. 1, pp. 19-27, Jan. 2011.
[12] K. Jensen, L. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3, pp. 213-254, 2007.