Final Report: Model Checking the SIP Protocol with Spin

Ying Wang & Fabian Marquart

February 4, 2018

1 Introduction

This project's goal is to gain practical experience with the state-of-the-art model checker SPIN in modeling and verifying concurrent systems. The system to be modeled is the Session Initiation Protocol (SIP), a communication protocol for establishing, maintaining, and terminating multimedia sessions such as those used by Voice over Internet applications. Spin and its input language Promela are used to (1) model certain aspects of the SIP protocol, to (2) specify high level properties that the model should satisfy, and to (3) verify whether the specified properties are indeed satisfied [3].

2 The SIP protocol

The Session Initiation Protocol (SIP) is a signaling, presence, and instant messaging protocol developed to set up, modify, and tear down multimedia sessions; request and deliver presence; and send and receive instant messages [2, p. 1].

2.1 A Simple Session Establishment Example

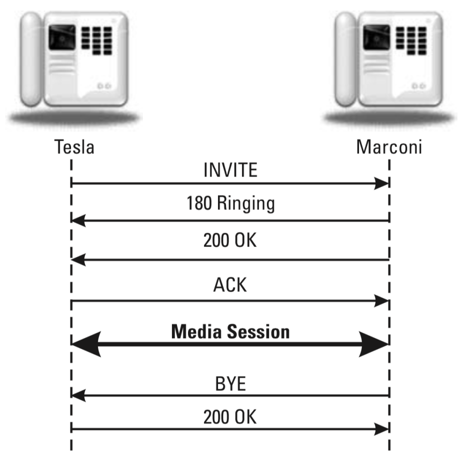

Figure 1: A simple SIP session establishment example, adapted from [2, p. 24].

Session establishment in SIP starts with an `invite` message by the client, called Tesla in Figure 1. The server answers with a `ringing`, sometimes called `trying` message. When ready, it sends an `ok` message to the client. Finally, the client answers with an aknowledgement message `ack`, and the multimedia session begins.

Now, when both agents are finished sending their audio or video data, the server closes the session with a `bye` message, which the client answers with another `ok`.

2.2 SIP Call with a Proxy Server

Most of the time, the agents using SIP to call each other do not know each other's IP address, for reasons such as dynamic reassignment of IP addresses. For this purpose, a proxy server that sits between client and server forwards the messages between both, as shown in Figure 2.

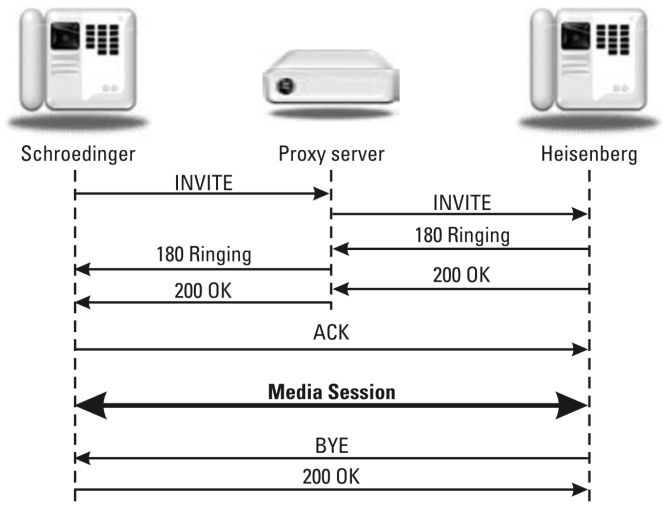

Figure 2: SIP call example with a proxy server, adapted from [2, p. 32].

3 Model Checking with SPIN and Promela

4 Linear Temporal Logic

[1]

5 Requirements

5.1 Model

The model should contain at least two user agents and two proxies. All scenarios related to setting up and terminating sessions should be modeled. User agents should be able to behave as client and as server.

- Client behavior
  - sending the requests INVITE, ACK and CANCEL for setting up sessions
  - sending the request BYE for terminating sessions
- Server behavior Sending corresponding responses to requests of client including:
  - accepting a request

- rejecting a request
- server error to fulfill a request

The behavior of the transaction user layer must be modeled, including client transactions and server transactions. Any user agent should be able to have multiple sessions at a time. Proxies may be stateless. The optional aspects of the SIP protocol include (1) registration, (2) dialogs (3) querying, (4) modifying sessions, (5) stateful proxies, (6) transport layer behavior, (7) forking and merging requests, among others irrelevant to the above mandatory aspects to be modeled.

5.2 Properties

High level properties that the model should satisfy are suggested. These should include at least two safety properties and two liveness properties. The model is simulated using Spin to check whether the properties specified are violated.

5.3 Verification

The suggested properties are expressed in LTL formulae. Spin is used to model check these properties. In case a property is violated, the obtained counterexample is analyzed and the cause of the property violation is identified.

6 The Model

The model consists of two proctypes: an agent and a proxy. The agent has an `idle` state, where it can choose between client and server behavior.

6.1 Implementation

6.2 Properties

1. Agents can have two sessions at the same time. One session is initiated by the agent itself, another session is initiated by the other agent.

   $\varphi_1 = \Diamond(A.mediaSessionClient \wedge A.mediaSessionServer)$

2. Eventually agent A is able to send another invite to agent B after sending an invite to agent B.

   $\varphi_2 = \Box(A.inviting \rightarrow \Diamond(invitesent_A = 0 \wedge A.idle) \wedge B.inviting \rightarrow \Diamond(invitesent_B = 0 \wedge B.idle))$

3. If the agent cancel its invitation, it is unable to initial a media session corresponding to this invitation.

   $\varphi_3 = \Box((\neg cancelsent_A = 1 \wedge A.mediaSessionClient) \wedge \neg(cancelsent_A = 1 \wedge A.mediaSessionClient))$

4. If agent A send an invitation to agent B, it will unable to send another invitation to B until the invitation either leads to a media session or it is canceled.

   $\varphi_4 = \Box(A.inviting \rightarrow \Diamond(A.inviting \wedge (\neg A.inviting \,\mathsf{U}\, A.idle))) \wedge \Box(B.inviting \rightarrow \Diamond(B.inviting \wedge (\neg B.inviting \,\mathsf{U}\, B.idle)))$

6.3 Verification

7 Conclusion