An LTL Specification and Verification of a Mobile Teleconferencing System

Yassine ELGHAYAM*, Mohammed OUZZIF** and Mohammed ERRADI*
*Laboratoire Alkhawarizmi de Génie Informatique, Université Med-V Souissi
ENSIAS, Rabat, Morocco
**École supérieure de Technologie Casablanca, Maroc
yassine.elgh@gmail.com; ouzzif@est-uh2c.ac.ma; erradi@ensias.ma

Abstract

Portable computing devices with wireless communication interfaces such as PDAs or smart phones led to the emerging of applications and services that support communication and collaboration among mobile users. In this paper we focus on the collaborative applications that allow a group of users, geographically distributed, to work together. In particular a teleconferencing system is an example of a collaborative application where remote users may interact to accomplish a common goal. Therefore, the teleconference owner may initiate a session, which is then joined and left dynamically by the involved group members. The teleconferencing collaborative system over a mobile network provides tools for communication between group members while allowing them to share information and collaborate with each other. Collaboration in a mobile condition poses new challenges, such as host mobility, limited device resources, and intermittent connectivity. The group members are frequently opposed to disconnections during the collaboration session. A user might become temporarily unavailable even though he or she is still engaged in the collaboration session.

The contribution of this paper is two fold. First we describe a formal specification of a teleconferencing system over a mobile network. This specification allowed us to define a set of important properties of the teleconferencing system dynamic behaviour. Second we formalize these properties using LTL logic while distinguishing between the required properties for the floor control and those related to the mobility requirements.

KEYWORDS: Mobile Teleconferencing; Control floor ; LTL properties ; Specification ; Verification.

1. INTRODUCTION

Multimedia Teleconferencing systems (MMTS) allow a group of users, geographically distributed, to work together. The remote users of an MMTS may interact share the available resources such as audio, video, whiteboard, etc. Access to these shared resources must undergo a policy of control and management to achieve a good and profitable working group. In the literature of collaborative systems, policies for the control and management of shared resources adopt the concept of "Floor control". The floor means a temporary permission associated with a shared resource and is dynamically given to group members to ensure a mutually exclusive use of these resources [1]. Today, the use of mobile devices (PDA, Laptop, Notebook, etc.) has become increasingly feasible. At the same time, the emergence of mobile networks (WiFi, Bluetooth, etc.) is also taxable in our daily lives. This advanced technology can open new forms of collaboration between mobile users. Such systems allow a group of mobile users to collaborate without being constrained by the geographical location [2, 3].

But this new environment poses new challenges related to the peculiarities of the terminals and the networks in use. The mobile terminals are characterized by a limited lifetime and mobile access networks are considered unstable because of the limited and variant bandwidth. In addition, the user mobility is another factor that can influence on the continuity of the collaboration.

As a result, mobile users are frequently faced with frequent disconnections during their collaboration. The presence of a user becomes intermittent while it is still engaged in a collaborative session.
An LTL Specification and Verification of a Mobile Teleconferencing System

The collaboration systems on a mobile environment must incorporate new mechanisms for the control and management of working together to adapt to the environment requirements. The detection of the state of connectivity (connected, partially connected, disconnected) is a crucial phase for managing collaborative users disconnected [4]. The floor management and control policy should be made from this state of connectivity.

The contribution of this paper is two fold. First we describe a formal specification of a teleconferencing system over a mobile network. This specification allowed us to define a set of important properties of the teleconferencing system dynamic behaviour. Second we formalize these properties using LTL logic while distinguishing between the required properties for the floor control and those related to the mobility requirements.

The work presented in this paper is an extension of our previous work [5] by considering the mobility constraints during a collaborative session.

The paper is organised as follows: Section 2 presents the architecture of a mobile teleconferencing management system. Section 3 gives a formal description of a floor control protocol for managing the access the available shared resources while emphasizing on mobility constraints. Section 4 presents the properties to be satisfied by the teleconferencing system including its mobility constraints. Then an LTL specification of these properties is presented. Before concluding, section 5 describes the verification process using Promela/Spin tool.

2. THE ARCHITECTURE OF A MOBILE TELECONFERENCE MANAGEMENT SYSTEM

The architecture of a mobile teleconferencing management system consists in a server side and a client side (figure1). The sever side is composed of two servers: the conference server, and the coordination and floor control server. The first one provides conference services which consist in storing and maintaining dynamically the conference information concerning a collaborating group using the teleconferencing system. The coordination and floor control server convey the floor control messages among the client entities and the floor server in order to guarantees mutual exclusive access to the shared resources.

The customer (Client) side consists in different remote conference actors. The first actor is the conference owner who creates and initiates the conference. The second actor is the conference chair who is in charge of managing the shared resources using a floor control mechanism. The third actor is a simple participant who joins the conference and asks resources needed for his participation to the teleconference. The client entities (participants and chair) may use mobile devices (laptop, PDA, mobile phone, etc.) to perform their tasks.

![Conference Management Architecture](image)

**FIGURE 1**: Conference Management Architecture

3. FORMAL DESCRIPTION OF FLOOR CONTROL PROTOCOL

3.1 Mobile connexion

Resource sharing between mobile participants must be organized according to a well-defined policy. A Simple Conference control Protocol SCCP, was proposed by the IETF community [6]. The SCCP document defines the
services required for simple conference control of tightly coupled conference. SCCP provides a set of services control conference namely: management of the set of members participating in the conference; management of the set of media/application sessions that constitute the conference (figure 2); and the floor control.

The floor control depends considerably on the state of connectivity. Therefore a connectivity detector must be committed to the SCCP entity.

The main conference management services are described as follows:

- **SCCP_Create_Conference**: allows to create a new conference by a conference owner (the responsible of a conference);
- **SCCP_Join**: allows to dynamically join a conference. The conference context is updated appropriately.
- **SCCP_Leave**: allows to dynamically leaving a conference without disturbance. The conference context is updated appropriately.
- **SCCP_Invite**: SCCP users may invite other users to participate in the current SCCP conference.
- **SCCP_Accept**: expresses that the invited participant accepts an invitation;
- **SCCP_Remove**: permits to a conference owner, in the case of a "conducted" conference, to remove a selected participant.
- **SCCP_Terminate**: in the case of a "conducted" conference, the conference owner may terminate a running conference.

The main Floor control services are depicted in the following:

- **SCCP_Give_Floor**: grants an exclusive floor to another participant;
- **SCCP_Grab_Floor**: allocates a floor for exclusive use by the requesting participant;
- **SCCP_Release_Floor**: releases a previously allocated floor;
- **SCCP_Ask_Floor**: asks the floor holder to grant an exclusive floor to the requesting SCCP entity;
- **SCCP_Reject_Floor**: allows the floor chair to reject a floor request;
- **SCCP_Ask_Floor_Pending**: notifies the participant that his floor request is pending and will be processed later.

In [4], the mobile user connectivity state fluctuates between the following states: Connected_State, Partially_Connected state, and Disconnected state. In the first state, the mobile device is similar to a fixed device connexion (good and stable connexion). In the second state, the mobile device is weakly connected to the network. In the third state, the mobile device is isolated from the network.

Figure 3 depicts the behaviour of the connectivity detector component. This behaviour consists in notifying the SCCP entity of the different connectivity state. For example, when the connectivity detector percept a bandwidth degradation or a low battery level, it transits from the "connected" state to the "partially connected" state and notify this change to the conference controller.
the local SCCP entity by the “partially disconnection” message. This entity will then notify the floor server and the conference server while the connection is partly maintained.

![Diagram of Connectivity Detector Behaviour](image)

**FIGURE 3: Connectivity Detector Behaviour**

### 3.2 Mobile floor description

In [7] the authors present the SCCP protocol service primitives and the requirements of the floor control protocol. The aim of our work is to specify formally the behaviour of this protocol taking in to account the mobile constraints of participants and to check temporal properties related to this protocol. The floor control policy adopted in this work is a chaired control policy [8] in which one person is designated as the chair person or the moderator of a floor by the conference owner. He gives/grabs the floor conference resources and processes the floor requests of a participant.

In this section we describe the behaviour of the SCCP entity using finite state machine by considering the notification made by the connectivity detector. According to the architecture given in figure 1, we distinguish three kinds of SCCP entity: the conference server SCCP entity located in the conference server, the floor server SCCP entity located in the floor control server and the customer SCCP entity located in the other logical entity of this architecture (owner site, chair site or participant site).

Figure 4 depicts the SCCP entity behaviour related to the session server. It is initially in the ready state and transits to an other state when it receives a SCCP conference service request (create, join, leave or terminate). Once this service is made, it returns to the ready state. The participant des-activation state is reached when a partial disconnection message is received from the detector. A participant may be reactivated to join the active collaborative group.

![Diagram of Session Server Behaviour](image)

**FIGURE 4: Session Server Behaviour**

Figure 5 depicts the floor control server behaviour. It is initiated by the ready state. When it receives a floor control service request (e.g. floor creation request), it transits to a floor control service rendering state (e.g. floor creation state). Once the service is made, it sends a notification (e.g. SCCP_Floor_Created) to the logical entity that has requested the service (e.g. owner) and return to the ready state. The floor server may also be notified by the message related to the partially disconnection materialized by the messages: “Holder_Partial_Disconnection”, “Chair_Partial_Disconnection” and “Participant_Partial_Disconnection”. In the first case, the floor server returns the
floor to the chairman. In the second case, it proceeds to the floor appointing to designate another chair. In the last case, it updates the floor context.

**FIGURE 5:** Floor Server Behaviour

Figure 6 describes the customer SCCP entity. It is initially in the init state. It consists mainly in the following states:
- Participant and holder states related to the simple
- Participant behaviour;
- Owner state related to the owner behaviour;
- Chair state related to the chair behaviour.
- And no active participant

The owner behaviour can be in the waiting state when it is waiting for a particular service from the floor control server or the conference server. The participant behaviour and the chair behaviour transit to the waiting_floor state when they do not hold the floor and are waiting for it. The floor is exchanged between the chairs and the participants. It is the chair which controls the floor holding. It requests periodically for the floor context to the floor server which contains the set of floor requesting participants. Based on this information, it gives the floor to a particular participant which becomes the holder and rejects the floor requests of the other participants. A participant in a holder state may release the floor and return to the participant state. The chair may grab the floor to the holder if he does not release the floor in a particular delay.

In order to consider the mobility constraints, the different actors (chair or simple participant) may transit to the “No Active Participant” state when receiving the partial disconnection message. A “No Active Participant” may return to the “Participant” state and become active in the collaborative group. This may happen when he is notified by the connectivity detector that he is reconnected (reception of “Connection” message).
The mobility constraints impact the complexity of our model. This is due to the integration of the connectivity detection components for each participant and to the addition of new states in the different processes of the model to consider the disconnection states.

4. FLOOR CONTROL PROPERTIES

In this section, we define the set of properties to be satisfied by our teleconferencing system over a mobile network. Then, a formal description of these properties is given using LTL notation.

4.1 Properties

We distinguish two categories of these properties. Properties 1 to 6 are considered to be general properties while properties 7 to 12 are specific to the mobility requirements.

General Properties:

- Property 1: It must be possible to define who is allowed to create, change and remove a floor in a conference. It is assumed that the conference has always these privileges.
- Property 2: It must be possible to use a chair-controlled floor policy in which the floor control server notifies the floor chair and waits for the chair decision. This enables the chair to fully control who has the floor.
- Property 3: Participants must be able to request a floor.
- Property 4: It must be possible to grant a floor to a participant.
- Property 5: A participant must be informed that he has been granted the floor.
- Property 6: The floor chair or moderator must be able to grab the floor from its current holder.
Specific Properties:

- Property 7: A participant must become no active in the cooperative group if it is partially disconnected.
- Property 8: A disconnected participant should be informed that it is disconnected.
- Property 9: A partially disconnected participant should not monopolize the floor.
- Property 10: A disconnected participant should not monopolize the floor.
- Property 11: A partially disconnected participant should belong to the cooperative group as no active participant.
- Property 12: A participant must become active in the cooperative group if it is reconnected.

4.2 LTL Description

The properties mentioned above are described according to the states of automata presented in the previous section. For each property, we give the propositions and temporal operators used in linear logic. Afterwards, we use the notations for the LTL as described in [9]: the operator "always" symbolized by $\Box$ and the operator fatality by $\Diamond$. The propositions uses classical logic operator: $\land$, $\lor$, $\neg$, and $\rightarrow$. The properties fluently used are expressed by $[] (p \Rightarrow q)$ or $[] (p \Rightarrow \Diamond q)$, where $p$ and $q$ are propositions. The first case expresses, the fact that at every time a state of system verifies $p$, it should verify $q$. The second case express, that at every time, the system verifies $p$, in all the future executions, it should verify $q$.

- **Property 1:**
  The used propositions are:
  - Floor_Server_Ready : represents that the floor server is in the ready state ;
  - Floor_Server_Appointing : means that the floor server is in the appointing state ;
  - Owner : means that the costumer automaton is in the owner state ;
  - Owner_Waiting : represents that the owner is in the waiting state.
  The LTL description of this property is :
  $[] (Floor\_Server\_Ready \land Owner \Rightarrow \Diamond (Floor\_Server\_Appointing \land Owner\_Waiting))$

- **Property 2:**
  The used propositions are :
  - Waiting_Floor_Context : means customer automaton is in the floor context waiting state ;
  - Floor_Context_Request_processing : means that the floor server is processing the floor context request ;
  - Waiting_Floor : represents that the customer automaton is waiting the floor.
  The LTL description of this property is :
  $[] (Waiting\_Floor\_Context \land Floor\_Context\_Request\_Processing) \Rightarrow \Diamond (Waiting\_Floor\_and\_Floor\_Server\_Ready)$

- **Property 3:**
  The used propositions are :
  - Participant : means customer automaton is in the participant state ;
  - Participant_Waiting_Floor : means that the customer automaton is related to a participant role and it is waiting floor state
  The LTL description of this property is :
  $[] (Participant \Rightarrow \Diamond (participant\_Waiting\_Floor))$

- **Property 4:**
  The used propositions are :
  - Chair : means customer automaton is in the chair state ;
  - Chair_Waiting_Floor : means that the customer automaton is related to a chairman role and it is waiting floor state
  The LTL description of this property is :
  $[] (Chair \Rightarrow \Diamond (Waiting\_Floor))$
• Property 5:
The used propositions are:
  - Floor_Holder : means that the customer automaton is related to a participant and is in the holder state
The LTL description of this property is:
\[
[] (Participant\_Waiting\_Floor \Rightarrow \Diamond (Floor\_Holder))
\]

• Property 6:
The used propositions are:
  - Chair : means customer automaton is in the chair state ;
  - Chair\_Waiting\_Floor : means that the customer automaton is related to a chairman role and it is waiting floor state
The LTL description of this property is:
\[
[] (Chair\_Waiting\_Floor \Rightarrow \Diamond (Chair))
\]

• Property 7:
To describe this property, we use the following propositions:
  - Partially\_Disconnected : means that the connectivity detector automaton is in the "Partially Disconnected" state.
  - Participant\_Desactivating : means that the automaton related to the conference server is in the "participant desactivating" state.
The LTL description of this property is:
\[
[] (Partially\_Disconnected \Rightarrow \Diamond Participant\_Desactivating)
\]

• Property 8:
To describe this property, we use the following propositions:
  - Disconnected : means that the connectivity detector automaton is in the "Disconnected" state.
  - Participant\_Disconnected : means that the customer SCCP entity automaton is in the "Participant Disconnected" state.
The LTL description of this property is:
\[
[] (Disconnected \Rightarrow \Diamond Participant\_Disconnected)
\]

• Property 9:
To describe this property, we use the following propositions:
  - Partially\_Disconnected : means that the connectivity detector automaton is in the "Partially Connected" state.
  - Floor\_Holder : means that the customer automaton is in the holder state.
The LTL description of this property is:
\[
[] (Partially\_Disconnected \Rightarrow \neg Floor\_Holder)
\]

• Property 10:
To describe this property, we use the following propositions:
  - Disconnected: means that the connectivity detector automaton is in the "Disconnected" state.
  - Floor\_Holder : means that the customer automaton is in the holder state.
The LTL description of this property is:
\[
[] (Disconnected \Rightarrow \neg Floor\_Holder)
\]

• Property 11:
To describe this property, we use the following propositions:
  - Disconnected : means that the connectivity detector automaton is in the "Disconnected" state.
An LTL Specification and Verification of a Mobile Teleconferencing System

- **No_Active_Participant**: means that the customer SCCP entity automaton is in the "No Active Participant" state.

The LTL description of this property is:

\[ \neg (\text{Partially\_Disconnected} \Rightarrow \Box \text{No\_Active\_Participant}) \]

- **Property 12**
  - **Connected**: means that the connectivity detector automaton is in the "Connected" state.
  - **No_Active_Participant**: means that the customer SCCP entity automaton is in the "No Active Participant" state.
  - **Participant**: means that the customer SCCP entity automaton is in the "Participant" state.

\[ \neg (\text{No\_Active\_Participant} \land \text{Disconnected} \land \text{Connected} \Rightarrow \Box \text{Participant}) \]

```proctype Participant_Behaviour(int i)
{
  state P[i] = Participant;
  do ::
    if :: (state P[i] == Participant) ->
      atomic {
        Channel_Partici_Server ! SCCP_Ask_Floor;
        Channel_Partici_Server_indice ! i;
        state P[i] = Waiting_Floor;
      }
    :: (state P[i] == Waiting_Floor) ->
      Channel_Server_Partici[i] ? msg;
      if :: (msg == SCCP_Give_Floor) ->
        state P[i] = Holder;
      :: (msg == SCCP_Reject_Floor) ->
        state P[i] = Participant;
      fi
    ...
  unless {
    Channel_Detector[i] ? msg;
    if :: (msg == SCCP_Partial_Disconnexion) ->
      Dcxn : atomic {
        state[i] = No_Active_Participant;
        Channel_Partici_Server ! SCCP_Partial_Disconnexion;
        Channel_Detector[i] ? i;
        Channel_Partici_Server_indice ! i;
      }
    if :: (state P[i] == No_Active_Participant) ->
      Channel_Det_Connectivity[i] ? msg;
      if :: (msg == Disconnexion) ->
        state P[i] = Disconnected;
      fi
    fi
    :: (state P[i] == Disconnected) ->
      Channel_Cxn ? msg;
      if :: (msg == SCCP_Partial_Disconnexion) ->
        goto Dcxn;
    ...
  }
}
```

**FIGURE 7**: Promela Spécification of "Participant " automate

### 6. VERIFICATION

In this section, we discus the verification of temporal properties using the Promela/Sping tool. This verification imposes the description of the automata in Promela.
6.1 Promela Description
A Promela automaton description [10] consists in a global loop "do .. od" composed of indeterministic choices that are described by the "if .. fi" statement, proposing then several alternatives as shown in Figure 7. This figure shows the specification of a participant behaviour.

The reactive or operational part of it is described by a guard whose evaluation allows (or not) the execution of the action. Each alternative corresponds to a particular state of the automata. Therefore, for the choice 2 for example, when the automaton is at "INIT" state (state == "INIT") and receives a "SCCP_CREATE_CONFERENCE" message from the conference controller, it sends the same message to the conference server and transits to the "WAITING" state.

In order to set up the interaction between processes, we use communication channels that are described as follow. For example, "CHANNEL_RECEPTION": a channel from which the customer automaton receives incoming messages; "CHANNEL_CONFERENCE_SERVER": a channel in which the customer automaton sends messages to conference server.

6.2 Verification with Spin
We start the validation of our Promela Model by checking that there are no deadlocks, no dead code. This means that all the transitions are executed. We also check that there are neither live locks nor end state unless for terminal states that belong to the specification of the processes.

The verification is done using Spin from Bell Labs. It consists in translating the LTL properties to Büchi automaton [11] which are translated themselves into Promela. The verification process uses the model-checking technique based on the accessible states. The obtained diagnosis is the result of synchronous product of the automata.

Figure 8 depicts the spin verification panel for the property 9, in which we have considered ten participants. Table 1 describes the verification spin results:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory usage</td>
<td>2,539</td>
</tr>
<tr>
<td>Total states</td>
<td>169</td>
</tr>
<tr>
<td>Reached states</td>
<td>104</td>
</tr>
<tr>
<td>Time consumed</td>
<td>0,062</td>
</tr>
</tbody>
</table>

**TABLE 1. Verification results**

6. CONCLUSION
In this paper, we have presented a formal specification of the floor control for a teleconferencing system over a mobile network. The specification was defined using a finite state machine model. Then we have described a set of properties which should be satisfied by the underlying teleconferencing system. In particular we have extended our previous work by considering the mobility constraints especially the frequent disconnections within a mobile network.

The suggested specification considers three new states: Connected State, Partly_connected state, and Disconnected state. Also the defined properties were formally described using the LTL notation. The Promela/Spin verification tool was used to first translate the specification automata using Promela language. Then the described properties have been confronted to the specification model by the synchronous product as provided by Spin to proceed for their verification. An implementation of the presented specification has been made for a teleconferencing system prototype over a fixed network. We note that the formalisation of the teleconferencing system was very helpful and make the implementation process easier. We are currently moving this implementation to a WiFi mobile network environment.
An LTL Specification and Verification of a Mobile Teleconferencing System

FIGURE 8. The ninth property Spin verification

REFERENCES


