DD2452 Formal Methods
Mobile IP Assignment

Spring 2009

Due: 5 March 2009

1 Introduction

It is often important to verify properties of a system based on a specification before the system is actually implemented. This can be done with an abstract model that captures the relevant parts of the specification, and *excludes* those parts that are not relevant for verifying the desired properties. The verification can be done in various ways. In this course a modeling language, PROMELA, and a tool, SPIN, have been presented. You should use these to verify some properties of the *Mobile-IP protocol*. The corresponding IETF Proposed Standard Protocol can be downloaded from `ftp://ftp.rfc-editor.org/in-notes/rfc3220.txt`. You should be acquainted with PROMELA, SPIN and XSPIN before you start on this assignment.

2 Requirements

The assignment should be done in groups of one or two students. Each group should present their solution at a work-station and at the same time submit one *report* documenting their model and results.

The report should contain:

- Name and e-mail address of each of the participants in the group.
- A clarification of the initial PROMELA model, i.e. what abstractions have been made, which parts of the specification are included in the model and which are not, together with motivations why. Explain also how these decisions have influenced the validation results. (Approx 1/2 page)
- Answers and results of the tasks. You should also include the parameter settings of the SPIN validations. *Note that the explanations are as important as the results of running the tool.* (Approx 2-5 pages)

3 Getting Started

1. Browse through the description of the protocol, so that you get a rough understanding of it. Start by reading the Protocol Overview at section 1.7, refer to the previous pages for the terminology used. When reading the rest of the protocol description, pay special attention to sections 3.3 and 3.4.

2. Walk through the draft PROMELA model that is enclosed in the assignment and try to understand how it works. Identify what parts of the specification is expressed in the PROMELA model.

4 Tasks

As a preparatory task, draw (some sort of) state graphs depicting the behaviours of mobile nodes and network agents.

1. (a) Specify the following properties in LTL. Motivate your formalizations.
   i. Always, the mobile node eventually talks
   ii. Always, the mobile node eventually talks or moves
   iii. Always, the mobile node eventually talks or moves or timesout

   (b) Check the above properties. Explain your results. Here and in the following tasks, in the case a property does not hold, describe an *error scenario* based on the *counterexample* provided by SPIN.

2. (a) Extend the PROMELA model so that a second mobile node (with the second agent as home agent) is running.

   (b) Check property 1-a-iii and explain your result.

   (c) Check property 1-a-iii with *weak fairness* and explain your result.

3. Comment out the second mobile node, so that only one mobile node is running in the PROMELA model.

   (a) Extend home agents, so that when granting a request to their mobile node, they store (remember) at what care-of agent the mobile node is registered.

   (b) Specify the following property in LTL. Motivate your formalization.
   - Always, when the mobile node talks, it is registered correctly.
     - Explanation: "Registered correctly" means that the mobile node is registered (according to the home agent) with the care of agent she "thinks" she is.
     - Hint: "Registered correctly" can be defined as an agreement between a variable in the mobile node and a variable in its home agent.

(c) Check property 3-b and explain your results.

4. (a) Add an *identification number* field to messages (see the protocol specification!).
   - Hint: Choose a suitably small range of possible identification numbers. Be careful with the range you choose, and how you treat the maximal value in this range!

   (b) Extend mobile nodes so that they disregard messages with an old identification number.

   (c) Check property 3-b and explain your result.

   (d) Extend agents so that they disregard messages with an old identification number.

   (e) Check property 3-b and explain your result.

5. Extend the PROMELA model again so that 2 mobile nodes are running again.

   (a) Check property 3-b and explain your result.

   (b) Add a *mobile node ID* field to messages. (Different mobile nodes should have distinct mobile node ID:s) Extend mobile nodes so that they disregard messages with the wrong mobile node ID, i.e. messages not intended for them.

   (c) Check property 3-b and explain your result.

5 Hints

- If verification is slow, try verification options that yield a faster verification; Weak fairness slows verification down, while Partial order reduction and compression speeds it up.

- When you are making extensions to the model, it is convenient to enclose your new constructions in # *if* .. # *endif* in order to do verification on different versions of the model. These compiler directives serve to include or exclude parts of the model in a particular validation or simulation.

- When writing properties in LTL, a useful predefined function of SPIN is `procname[pid]@label` which returns true at a state only if the next statement that can be executed in the process with instantiation number `pid` is the statement that comes after the label `label` in `procname`. (It is an error if the process referred to with `pid` is not an instantiation of `procname`).

  To illustrate how this function can be used in expressing properties we give the following example.

  Property: The mobile node will eventually move.

  Expression of Property in LTL:

  #`define moves MobileNode[3]@Movement`

♢ `moves`

Note that `pid` is the unique process instantiation number of a process at each running. The `pid`s start with 0 and are assigned in order of creation. The single process of type `MobileNode` in the draft model is the forth created process, hence its `pid` is 3.

- When writing properties in LTL one can refer to the current value of a local variable in a similar way. The function `procname[pid]:var` refers to the current value of the local variable `var` in the process with instantiation number `pid`. (Again, it is an error if the process referred to with `pid` is not an instantiation of `procname`). We illustrate this with an example.

  Property: Eventually the value of `currentLink` in the mobile node is equal to the value of `link` in the first agent.

  Expression of Property in LTL:

  `#define areEqual MobileNode[3]:currentLink == Agent[1]:link`

  ♢ `areEqual`