Generating Interactive Protocol Simulations and Visualizations for Learning Environments

S. Papakosta, C. Burger

University of Stuttgart, Institute of Parallel and Distributed High-Performance Systems (IPVR)
Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany
Email: {papakosi, caburger}@informatik.uni-stuttgart.de

Abstract

This position paper introduces the idea and concept of generating interactive protocol simulations and visualizations for teaching environments in lectures as well as for learning environments over the internet. Since the realisation is based on the SPIN model checker, a short presentation of this verification tool and its language Promela is shortly been given before sketching the generation procedure itself. The architecture presented here is open to further extensions.

1 Introduction

Communication protocols in the area of computer networks and distributed systems tend to be very complex. In security protocols for instance, there are many parameters such as keys and random numbers involved, whose meaning is not offhand to understand. Thus, the teaching procurement in this area is particularly complicated and difficult to undertake. To gain dedicated teaching and learning support we started with developing a Java-based Toolkit called HiSAP (Highly interactive Simulation of Algorithms and Protocols). HiSAP is a framework for the generation of Java-Applets, which perform simulation and visualization of algorithms and protocols. Applets generated with HiSAP (see figure 1) offer a high-level interactivity in order to allow students to experiment with the protocol and try out several feasible run sequences.

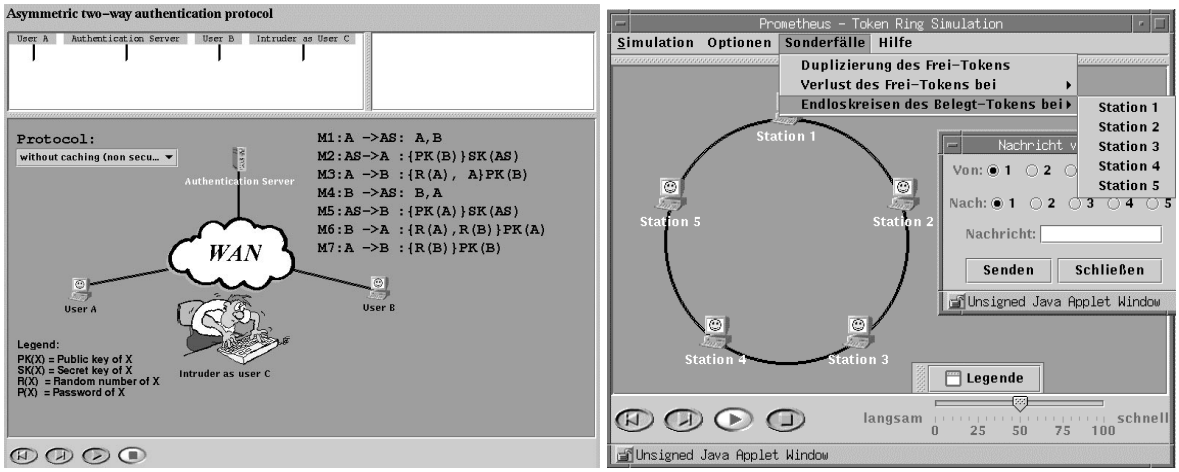

Figure 1: Applets for the two-way-authentication and the token-ring protocol

HiSAP-Applets for some selected protocols have been used successfully during lectures that took place at the university of Stuttgart [BR01]. Although an enhancement on the part of understanding the complexity and dynamics of protocols has been achieved this way, the effort of preparing such an interactive learning module has only slightly been improved by the Toolkit, since essential parts of the Java-Applets (e.g. parts that refer to the protocol logic) still have to be implemented manually.

A further significant requirement in the context of teaching protocols, concerns the possibility to have predefined simulation runs. Especially, when thinking of beginners being confronted the first

time with a protocol, they would probably like to know about those input sequences demonstrating main functions of the protocol. Moreover, when doubting the usefulness of certain parameters, learners should be able to perform modifications and observe the resulting protocol behaviour. But it can be very difficult and protracted for the learner to search for existing conflicts between the protocol specification and the given correctness criteria.

Hence we have the intention of developing a new tool, in such a way as to optimally:

1. provide a learning environment, which offers extensive support in the experimentation with algorithms and protocols within the scope of lectures, exercises and private study (telelearning)
2. reduce the overhead for the preparation of appropriate interactive learning materials

To this end, the tool accepts formal protocol specifications as user-input, and generates automatically a simulation model, a visualization of this model as well as test sequences for main protocol function and error cases. The creation of a library consisting of predefined formal protocol specifications provides widened assistance in the preparation of such learning modules.

We have already started with the development of a prototype, which fulfills the requirements mentioned above, by employing the SPIN model checker. A brief introduction to this verification tool and its language Promela is given in the next section. In section 3 we discuss the architecture and a high-level process flow of our implementation. Section 4 contains conclusions and directions for future work.

2 SPIN and PROMELA

Promela (PROcess MEta LAnguage) is a high-level specification language for modelling interactions in distributed systems, and for expressing logical requirements (correctness criteria) about such interactions [Hol91]. The model checker SPIN (Simple Promela Interpreter, a similar functionality is provided by [CWB]) accepts specifications written in this language, and it can produce automated proofs for each type of property. SPIN either proves that a property is valid in the given system, or it generates a counter-example (so called trail-file).

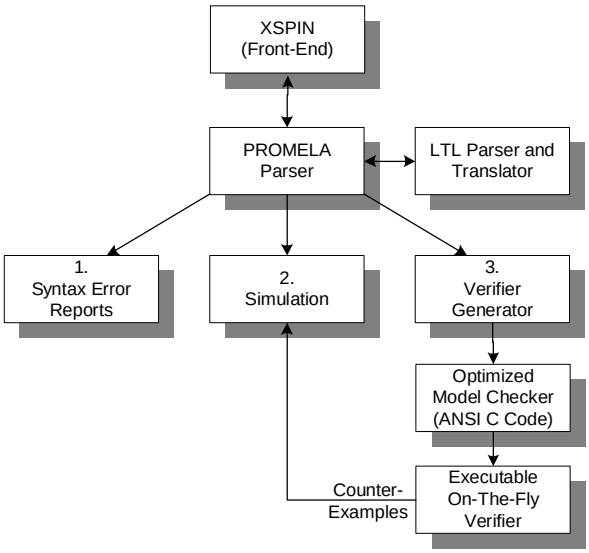

Figure 2: The structure of SPIN simulation and verification

The basic structure of the SPIN model checker is illustrated in fig. 2 [Hol97]. After specifying a high-level model of a distributed algorithm or protocol (typically using SPIN's graphical front-end XSPIN) and fixing syntax errors, interactive simulation can be performed. Then, in a third step, SPIN is used to generate an optimised on-the-fly verification program from the specification. This verifier is compiled and executed. If any counter-examples to the correctness criteria are detected, these can be fed back into the interactive simulator and inspected in detail to establish, and remove their cause.

Although SPIN really provides effective support in verification of distributed process systems, it is not sufficient for teaching purposes and does not fulfill all the requirements we indicated in the introduction part. The main reason why SPIN can not be directly deployed for teaching and learning environments, is the absence of a proper visualization, in a way that user interaction can occur. The only grant of SPIN in this direction is the generation of a standard Postscript file that contains the message sequence chart for a simulation. However, this type of protocol presentation deviates a lot from an interactive visualization and can definitely not be sufficiently adapted for teaching purposes.

3 A tool for interactive protocol simulation and visualization

To cover the requirements of an elaborated teaching and learning environment, our protocol simulation and visualization tool has to operate like a black box (see figure 3). From an abstract, formal protocol specification and the appropriate correctness criteria, a simulation model and a suitable visualization is generated. With such a system, interactive experiments can be performed.

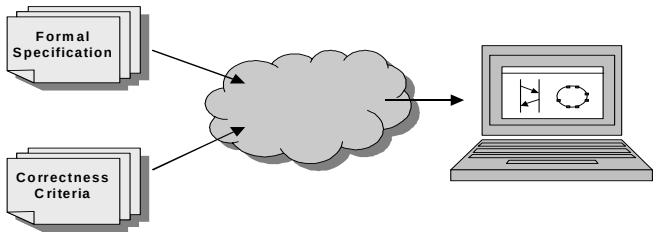

Figure 3: The idea for generating protocol simulations and visualizations

Regarding predefined test sequences to support such experiments, there are two possible cases, which could be differentiated:

1. The given formal protocol specification fulfills the correctness criteria, i.e. the protocol has been correctly specified and provides the desired service. In this case only a demonstration of the correct protocol behaviour within a visualization has to be carried out (e.g. for beginners or a short presentation during a lecture).
2. There are hidden flaws in the protocol specification. These have to be detected and visualized accordingly.

The concept for generating a visualization of protocol test cases is illustrated in fig. 4. The user enters the formal specification of the protocol in a friendly and comfortable user-interface. His input is delivered to the protocol generator. The seven steps of fig. 4 are described in the following:

1. The user-input is translated into a corresponding Promela-Code (e.g. from MSC or SDL to Promela [IRT]).
2. The abstract, formal protocol specification generated this way, is automatically converted into a simulation model, which pertains to a lower abstraction level (cp. fig. 4), due to the fact that it includes more implementation details about the internal sequential computations. Even though SPIN provides an innate simulation model, a new simulation model has to be generated, that applies to Java and is open to further extensions.
3. SPIN runs the executable verification program for the given Promela specification and correctness criteria.
4. As soon as a verification has been performed, SPIN delivers a trace, that is either the run sequence for a correct protocol behaviour, or consists of (one or more) counter-examples, in case of flaws in the specification.
5. This trace has to be converted into a format, that can be used interactively by the user as input for the visualization.
6. The simulation model receives this trace step by step, in terms of single user entries, from the visualization, and uses it as input-parameters for the simulation run. There are two possible ways for the user to trigger the simulation:
   a) following the trace for one predefined path (given by the test case)
   b) selecting one of the available next-step alternatives for each step of the simulation
7. The simulation results are visualised properly.

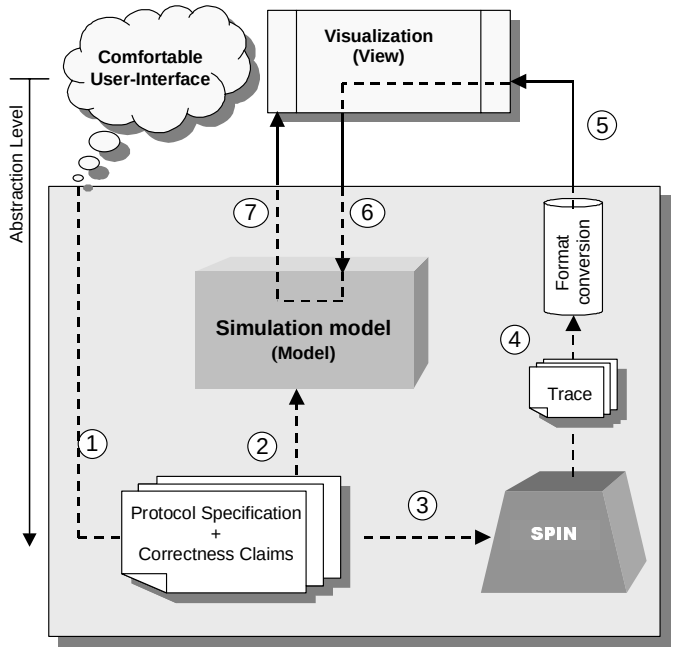

Figure 4: The concept for generating interactive protocol simulation and visualization

4 Conclusion and future work

Up to now, there are a lot of environments to create simulations and animations, like e. g. the Network Simulator [NS] or COVERS [COV], that restrict to performance aspects of communication protocols. In contrast to this, the main focus of our work is on performance of correctness checks for protocols. Thus, we have presented a concept for automatically generating interactive protocol simulations and visualizations including test cases, by using formal protocol specifications and correctness claims. Integrating the SPIN model checker for this realisation has the advantage, that an effective verification process is assured, since SPIN guarantees the detection of all existing errors and flaws of the protocol specification.

The automated generation of protocol simulations and visualizations reduces significally the overhead for preparing learning materials. The user-friendly interface as well as the interactive use of the tool, support and simplify the effort of understanding complicated and dynamic processes such as protocols by distributed systems and computer networks. These protocol visualizations can be further used for the presentation during lectures, private study and teletraining purposes or even for design purposes in software engineering.

Our current work is indirectly insinuated in section 3. According to the second step illustrated in fig.4, we are primarily concerned with the automated translation of formal Promela specifications into Java, i.e. we work on the prototypical combination of the new tool with the existing HiSAP-Toolkit. The generation of a suitable visualization from the simulation model is also in process. The supply of a user-friendly interface for the learners as well as the integration of further aspects (e.g. performance test) are motivating the directions for our future work.

References

[BR01] C. Burger, K. Rothermel, *A framework to support teaching in the area of distributed systems*, acm JERIC 2001.

[COV] http://www.xjtek.com/products/covers/

[CWB] http://www.dcs.ed.ac.uk/home/cwb/

[Hol91] G. J. Holzmann, *Design and Validation of Computer Protocols*, Prentice-Hall International, 1991.

[Hol97] G. J. Holzmann, *The Model Checker SPIN*, IEEE Transactions on Software Engineering, Vol. 23, NO. 5, May 1997.

[IRT] Intermediate Representation Toolkit, http://www-verimag.imag.fr/ DIST_SYS/ IF/

[NS] http://www.isi.edu/nsnam/ns/