Broadband-ISDN functional model simulator with IN capabilities

V. Dobrota, C. Miron, A. Vlaicu
Technical University of Cluj-Napoca, 26 Baritiu Street, 3400 Cluj-Napoca, Romania

Abstract: *This paper presents a functional model simulator for broadband integrated services digital networks, based on requirements which fit within the objectives of RACE-MAGIC project. The first software simulator, written in Erlang, is an ISUP (ISDN User Part) state machine implementation, designed for measuring the performances of interworking of signalling between B-ISDN and the existing N-ISDN. The second software simulator represents a proposal for a MAGIC Functional Model implementation, using Erlang facilities for a rapid prototyping of a robust system. The future global telecommunications network should be able to understand all of terminals requirements, to negotiate the services and the resources to be offered and finally to have an adaptive behaviour.*

1 INTRODUCTION

One of the most important European collaborative projects for telecommunications within RACE *(Research and Development in Advanced Communications Technologies in Europe)* is MAGIC *(Multiservice Applications Governing Integrated Control)*. Its approaches were to investigate advanced Broadband-ISDN signalling protocols and it has defined several functional models to test the capability to handle the complex multimedia multiparty services. Within the very first stages of the project it has been necessary to prototype the ISUP *(ISDN User Part)* state machines in *Erlang*, in order to test the interworking of B-ISDN and the existing Narrowband-ISDN signalling systems. Some of the advantages by using this new symbolic concurrent programming language *Erlang* (realised by the Computer Science Laboratory of Ellemtel Utvecklings AB, Sweden) are the rapid prototyping (comparing to imperative languages) and the building of a robust concurrent system [1]. The first aim of this paper is to discuss the concepts of the software simulator for ISUP protocol, as it has been presented in [5], but with a new implementation, based on inter-processes communication by Pid (process identifier) instead of using inter-processes communication by port/socket. Why did we consider this new version more interesting and more useful than the previous one? The answer is: because of *the efficiency and the capability to be generalised for more than two state machine configurations*, for instance in a completely different application with six state machines and other building blocks, modelling a signalling system. Even the method is similar to that applied for the ISUP state machine implementation, the rules within this new software simulator are compatible to the MAGIC standard functional model, which has three layers (Call Control, Resource Control and Bearer Control). The idea was to define a state machine for each layer, for both source

and recipient side (this means a six state machine configuration). The new concepts (from MAGIC point of view) as timers (attached to each state machine) and links (between state machines) are compatible to requirements of measuring the performances of signalling protocols. *The paper is focused on the resource allocation strategy to be adopted, the reaction time and the probability to support a call because of the incompatibility, the absence of the user or not enough resources.* It becomes possible to appreciate if the "amount" of artificial intelligence the network is provided with is fully enough to have an adaptive behaviour. This means that all of terminals requirements must be understood, the services and the resources offered have to be subjects of negotiation between users and the network.

2 SOFTWARE SIMULATOR CONCEPTS APPLIED TO AN ISUP STATE MACHINE IMPLEMENTATION

The first aim of this paper is to present a method of understanding the mechanism of a client/server architecture and the inter-processes communication using a Pid. This is a new version of the simulator designed in [5] and presented at SEHE'94 It is expected to improve the ability to create a rapid prototype, being an ISUP state machine implementation. The technical requirements are very important in designing of this tool but they can not be covered by this paper (more details could be found in [3],[4],[5]). For this application it has been necessary to create the following building blocks (each of them has one or more *Erlang* processes attached): *N-ISDN Network Application, Interworking Unit* and *Communications Link* [4]. In order to simplify the implementation it is possible to have two configurations: *single ISUP state machine (Figure 1)* and *two ISUP state machines (Figure 2).*The ISUP simulation makes use of the following main component blocks: **CC** *(Call Control)*: This is an entity that simulates the call control processes of the signalling applications. It is realised as an user interface displaying and allowing the user to input signalling primitives. **SPRC** *(Signalling Procedure Control)*: This is an entity that simulates the signalling procedure control processes. **TIMER** *(Timing Control)*: This is an entity that offers four independent timers (T1,T5,T7,T9) for every active circuit. It receives timer_start and timer_stop commands and sends alarms when the time expired. **MASTER & CPCI/CPCO** *(Master & Call Processing Control Incoming, Call Processing Control Outgoing)*: This is an entity that monitors all the sockets of the ISUP state machine and also implements all the functions of the CPC machine for both incoming and outgoing calls. It is not realised as a user interface but it provides complete details about the exchange of information through the sockets and about every internal transition of the current state machine. **SPRC1-SPRC2 LINK MASTER**: This is an entity that essentially simulates the N-ISDN link between two ISUP state machines. It includes the function of SPRC for both machines and also monitors the exchange of information

from one ISUP machine to another. All these blocks communicate by sockets: socket_CC, socket_SPRC, socket_TIMER, socket_SPRC1, socket_SPRC2.

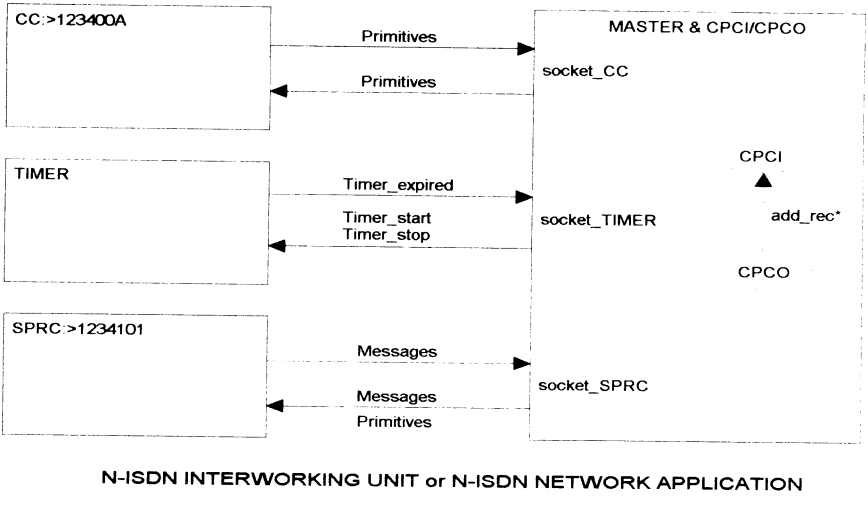

Figure 1. Single ISUP state machine configuration

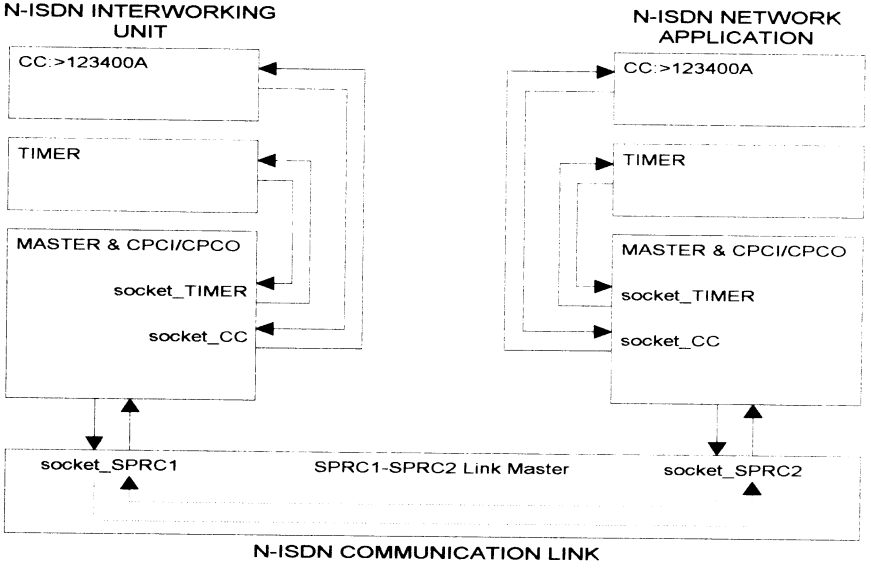

Figure 2. Two ISUP state machine configuration

The software consists of the following source files: **v20.erl**: This is MASTER&CPCI/CPCO for ISUP state machine implementation. **v22.erl**: This provides the same function as with v20.erl but is suitable for exchange control call, with priority for the outgoing calls. **v23.erl**: This is ISUP emulation test

program for CC, starts a server to CC. **v25.erl**: This is ISUP emulation test program for SPRC, starts a server to SPRC. **v27.erl**: This is SPRC1-SPRC2 LINK MASTER which effectively realises a link between two ISUP state machines. **vt6.erl**: This is the TIMER for ISUP state machine implementation and it offers four independent timers (T1,T3,T7,T9) for every active circuit. In fact this is the second step in the action of testing a communications link between two ISUP state machines because it supposes that previous test (single ISUP state machine) was successful for both terminal and interworking unit. The operator has access to the call control interfaces of both ends of the link only. Compare to the procedure applied for the version using an inter-processes communication based on port/socket [5], within this version using an inter-processes communication based on Pid, the initialisation is much more simple. The opening of the seven windows for every item involved will be done automatically by a module called *sim*, without any requirements addressed to the user of the simulator about internal details of the machine. In fact the number of the port assigned for a given socket connection varies from the system to system and from the session to session, but its exact value is detected each time by *iv* module (InterViews)[2] and is sent (as a Pid) to all other modules involved in the communication through the given socket. The module called *sim* has the following structure:

```
%*********************************************************
%              S I M U L A T O R
%*********************************************************
% File:     sim.erl
% Version:  1.0
% Date:     03/08/94
% Author:   Virgil Dobrota
%*********************************************************
module(sim).
compile (export_all).
start() ->
  iv:start(),
  Link_ID    = spawn(v27, test_Link, []),
  TIMER1_ID  = spawn(vt6, test_TIMER, []),
  TIMER2_ID  = spawn(vt6, test_TIMER, []),
  CC1_ID     = spawn(v23, test_CC, []),
  CC2_ID     = spawn(v23, test_CC, []),
  SM1_ID     = spawn(v20, test_SM1, []),
  SM2_ID     = spawn(v22, test_SM2, []),
% ------------------------------------------------------
% C O N F I G U R A T I O N S
   SM1_ID    ! {config, CC1_ID, TIMER1_ID, Link_ID},
   SM2_ID    ! {config, CC2_ID, TIMER2_ID, Link_ID},
   CC1_ID    ! {config, SM1_ID},
   CC2_ID    ! {config, SM2_ID},
   TIMER1_ID ! {config, SM1_ID},
   TIMER2_ID ! {config, SM2_ID},
   Link_ID   ! {config, SM1_ID, SM2_ID}.
```

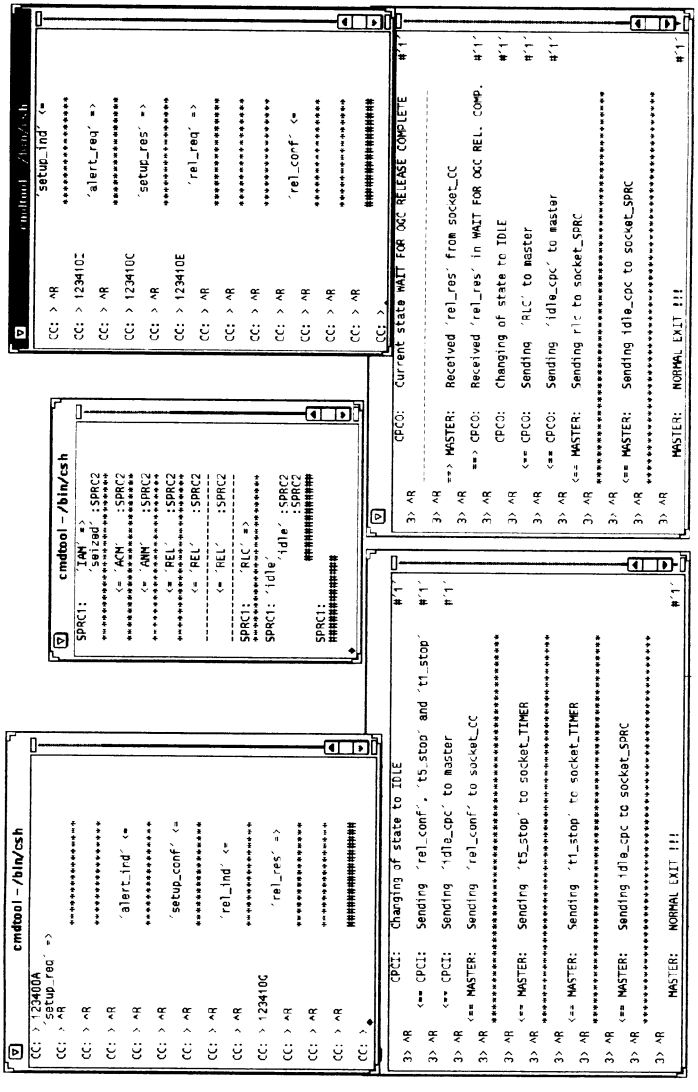

Figure 3. Experimental results for ISUP state machine implementation

3 B-ISDN FUNCTIONAL MODEL SIMULATOR

MAGIC Standard Functional Model Simulator has been adopted with three layers: CC *(Call Control),* RC *(Resource Control)* and BC *(Bearer Control).* Compared to previous models this one has a new layer RC, introduced between CC and BC to configure, allocate and control special resources that are internal to the network. The services requested in the future will determine the updating of RC only, which could be an advantage, as well as the fact that information flows within each layer can be simplified. The

disadvantages are related to the lack of IN features, which means all switches must be updated for new services and also a more intelligent broadband signalling system is required [3].

MAGIC Alternative Standard Functional Model Simulator has been adopted with two layers and the addition of the IN functional architecture to provide control of specialised resources (protocol converters, audio/video conference bridge, information distribution bridge, text to speech synthesis, synthesized voice/speech recognition etc.).The advantage of a simplified information flows within each layer remains as in the previous model, but with a simpler broadband signalling system and a clear location of the complex functions in the IN SRF. The disadvantages are related to the requirement addressed to all complex calls to refer out to IN and also it is necessary to define new interfaces between IN and CC/BC [3].

3.1 Software Simulator Concepts Applied to MAGIC Standard Functional Model

There are four types of modules: *State Machine*, *Timer*, *Agent* (OCCA or TCCA) *for user interface*, *Communications Link* between two state machines. Each module has a window to display the events and accepts commands from the keyboard in order to modify some parameters (except Communications link which displays only). The timer module has three interactive reconfigurable timers (t1,t2,t3) attached to each state machine.

Atomic action concepts have been implemented as follows: during the Phase 1 *a superior* uses a BEGIN to create a branch to a subordinate, asking it to accept an atomic action. *A subordinate* responds with READY or REFUSE. During the Phase 2 a subordinate will be instructed by the superior to COMMIT or to ROLLBACK and between Phase 1 and Phase 2 the subordinate could send ABORT if necessary. *A master* is a node that is a superior with one or more subordinates but has no superior of its own. *An intermediate* is a subordinate of another node but is also superior of one or more subordinates. *A leaf* is a subordinate of a node and superior to none and unlike masters or intermediates it only has a single relationship.

There are 21 modules: seven for CC, five for RC, five for BC, two for CC-RC LINK and two for RC-BC LINK (*Figure 4*). The strategies for allocation of resources could be the following three: before call offering, after response to a call offering by a compatible terminal equipment, after call acceptance by the called human user or user application.

The description of the modules:

OCC *(Outgoing Call Control):* SM1 (State Machine 1) is the master and superior for three subordinates: OCCA, RC, OCC-TCC LINK. **OCCA** *(Outgoing Call Control Agent),* subordinated to OCC, is a leaf. **OCC-TCC LINK:** it was not designed in the MAGIC model, displays only the messages exchanged between OCC and TCC, being subordinated to OCC, superior for

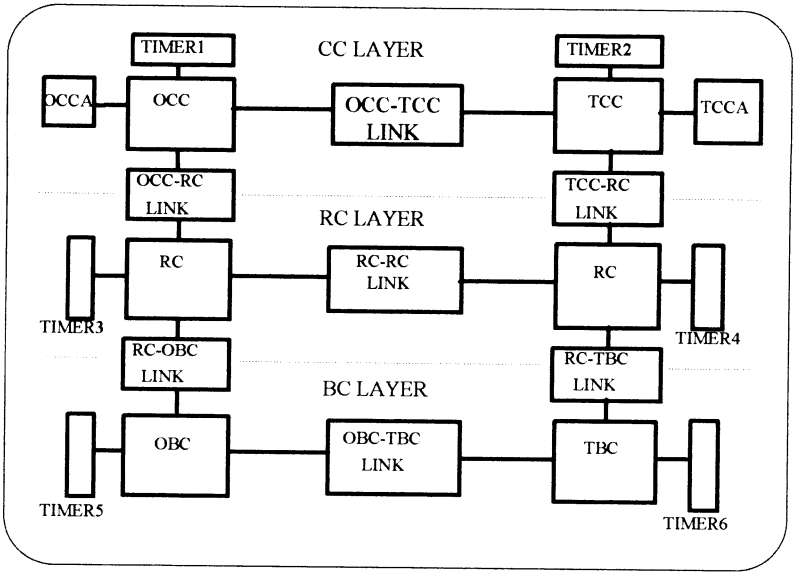

Figure 4. Software Simulator for MAGIC Standard Functional Model

TCC and intermediate. **TCC** *(Terminator Call Control):* SM2 (State Machine 2) is subordinated to OCC-TCC LINK, intermediate, superior for TCCA, TCC-RC LINK. **TCCA** *(TCC Agent)* is subordinated to TCC, leaf. **OCC-RC LINK** is subordinated to OCC, intermediate, superior for RC. **TCC-RC LINK:** is subordinated to TCC, intermediate, superior to RC. **RC** *(Resource Control):* SM3 (State Machine 3), subordinated to OCC-RC LINK, intermediate, superior for RC-RC LINK and RC-OBC LINK. **RC** : SM4 (State Machine 4) is subordinated to TCC-RC LINK, RC-RC LINK, intermediate, superior for RC-TBC LINK. **RC-RC LINK:** is subordinated to RC(SM3), intermediate, superior for RC (SM4). **RC-OBC LINK:** is subordinated to RC (SM3), intermediate and superior for OBC. **RC-TBC LINK:** is subordinated to RC (SM4), intermediate and superior for TBC.**OBC** *(Outgoing Bearer Control):* SM5 (State Machine 5), is subordinated to RC-OBC LINK, intermediate, superior for OBC-TBC LINK. **OBC-TBC LINK:** is subordinated to OBC, intermediate, superior for TBC. **TBC***(Terminator Bearer Control):* SM6 (State Machine 6), is subordinated to OBC-TBC LINK, RC-TBC LINK, leaf. **TIMER 1... 6:** have not been designed in MAGIC model, attached to OCC, TCC, RC (SM3), RC (SM4), OBC and TBC respectively, receives t1_start, t1_stop, sends t1_expired.

3.2 Experimental results

The software simulator was implemented in Erlang, running UNIX (SunOS) on a SunSPARC station. The configurable parameters to be modified by an interactive selection (even during the evolution of the protocol) were the values for the timers especially. t1, t2, t3 can be selected for each TIMER from milliseconds up to few minutes (the precise granularity depends on the performances of the host machine: for instance could be 200 ms). The optimum strategy can be selected by modifying the timers and the rules within each state machine. The very first experiments were done for a pure post-allocation of resources strategy with aborting strategy with no confirmation from the superior. It is for further study other strategies and also the implementation of the MAGIC Alternative Functional Model.

4 CONCLUSIONS

Two software simulators presented in this paper have been implemented using a symbolic concurrent programming language Erlang, on top of UNIX. The first one is a single/dual ISUP (ISDN User Part) state machine tool useful to test the interworking of signalling between Broadband-ISDN and the existing Narrowband-ISDN. The concepts and the methodology were generalised in the second software simulator which has six state machines and other building blocks and it is an implementation of the MAGIC Standard Functional Model, defined within a RACE project. An adaptive behaviour of the intelligent network could be established by measuring its performances such as resource allocation.

Acknowledgements: The authors would like to acknowledge the information and the support provided by staff involved in RACE-MAGIC project, in particular thanks go to Richard Boulter, Dick Knight, Graham Popple and Steve Towndrow from BT Laboratories (UK). In addition the assistance from Bjarne Dacker from Ellemtel Utvecklings Aktiebolag is also gratefully acknowledged. Very special thanks go to Paul McDonald from BT Laboratories for his competent help.

REFERENCES

[1] Joe Armstrong, Robert Virding, Mike Williams, *Concurrent Programming in Erlang*, Prentice Hall, 1993

[2] Marie Jonsson, *The Interviews Interface for Erlang, User Manual*, Ellemtel Utvecklings AB, 1992

[3] ***, *RACE Project R2044, MAGIC - Multiservice Applications Governing Integrated Control, Stage 2 Specification*, 6th Deliverable, 1993

[4] Virgil Dobrota, *N-ISDN ISUP Simulator*, British Telecommunications plc,1993

[5] Virgil Dobrota, *Use of the Symbolic Concurrent Programming Language Erlang to Test the Integrated Digital Networks Access*, presented at SEHE'94 (First International Conference on Software Engineering in Higher Education), Southampton,UK 23-25 November 1994, to be published in SEHE'95 Proceedings