23

Time Oriented Protocol Testing Simulator

Ken Ohta, Nobumasa Nakano, Sanshiro Sakai,
Takashi Watanabe and Tadanori Mizuno
Department of Computer Science Faculty of Engineering Shizuoka University
3-5-1 Johoku Hamamatsu 432, Japan

As networking grows, requirements for communication services become more complex, such as 'time-critical-communication real time system.' Existing FDTs for communication services and protocols aren't sufficient for specifying of real time applications. In this paper, we propose *T-PROMELA* which can specify communication services for real time applications and *T-SPIN* which is the tool for testing and simulation of software in *T-PROMELA*.

1 INTRODUCTION

Designing efficient and unambiguous communication protocols and services are necessary since protocol works in parallel on the computer network installed in various vender's equipments [1]. Various kinds of formal description techniques(FDT), e.g.,LOTOS, Estelle, SDL, Petrinet, Z and so forth, are developed and used for designing communication protocols and services[2] [3].

The fields of multimedia application and factory automation application require time-critical-communication that guarantees communication services within certain time limit. Communication services specified by existing FDTs have concerned with mainly only time *order* problems. In the meantime, there have been proposed some formal description techniques that have taken the real time concept; e.g., LOTOS-T[4], EXT-LOTOS[5] and LOTOS/ T [6].

In this paper, the time extension of *PROMELA* [7], which is one of the FDTs, is proposed. We name it *T-PROMELA* , and it can be applied to formal specifications of real time applications, for instance time-out, event scheduling on time, measurement of elapsed time, and so on. The specifications can be traced and tested by the simulator *T-SPIN* .

In Section 2, the basic concept of time-critical-communication [8] and real time problem(requirements) are described. In Section 3, we introduce *PROMELA* and *SPIN*, which is Simulator and Validator for specifications written in *PROMELA*. Section 4 proposes *T-PROMELA* and *T-SPIN*. In Section 5, we illustrate the use of *T-PROMELA* and *T-SPIN* and finally, Section 6 summarizes the work done up to now and the planned activities.

T. Mizuno et al. (eds.), *Protocol Test Systems*

2 Time Critical Communication

In this section, we discuss the basic concept of time-critical-communication and the real time problem(requirements) for FDT.

2.1 Time Critical Communication

The standardization effort regarding a Time Critical Communication Architecture (TCCA)[8] which will support the real time communications within the area of MAP(Manufacturing Automation Protocol) and IEC Fieldbus is carried out in ISO/TC 184/SC 5/ WG 2/ TCCA.

- "The communication is time-critical if the application process sending a message requires it to be received(or received and acted upon, or received and acted upon and confirmed) within a certain time after it submits its send request to the system"
- "The term *time-critical* is used to represent the presence of a *time window*, within which one or more specified actions is required to be completed with some defined level of certainty"
- "The network time constraints are determined by the requirements of the application, and there is no intention of defining absolute time constraints for any particular architecture"

The *time window* is parameterized over the following parameters.

- TW_w: The time window that expresses the limit of the elapsed time between one TCCE's send time and another TCCE's receive time
- TW_R: The time window that expresses the limit of the elapsed time between one TCCE's send time, and the receive time of the reply sent by another TCCE after receiving the message
- TW_T: The time window that expresses the limit of the elapsed time of transaction

2.2 Requests of real time concepts

For example, specifications of ST(send time), RT(receive time) and TT(transfer time) are parameters specifying performance of the time-critical-communication services in FDT. Furthermore, we want to specify time out and real-time requirements, which claim the sequence of operations completes within the real time constraint. The following explicit specifications of real time related description are necessary.

1. Time type variables
2. Specification of elapsed time of executive statement
   When the elapsed time of executive statement is pre-defined, it can be described in specifications.
3. Absolute and relative time expression
4. Global time and local time

3 PROMELA AND SPIN

In this section, we introduce *PROMELA*[7] and its simulator and validator *SPIN*[7]. The reason why we use *PROMELA* is that specifications written in *PROMELA* are executable and testable without additional specifications. Furthermore, *PROMELA* validation functions are very effective. In consideration of the above characteristic, *PROMELA* is suitable to be extended for real time communications.

PROMELA(PROtocol MEta LAnguage) is a language for protocol design and is based on CSP[9] and C. Guarded command and select command are introduced from CSP, and declarations of variables and expressions are affected by C.

SPIN is the tool that simulates and validates protocol validation models written in *PROMELA*. The simulator of *SPIN* quickly tells users whether they are on the right track with a design or not. The validator of *SPIN* performs full state space search(for small systems) or supertrace search (for larger systems). The implemented validation techniques are assertion, deadlock detection, temporal claims and proper end-state. The structure of *SPIN* is shown in Figure 1.

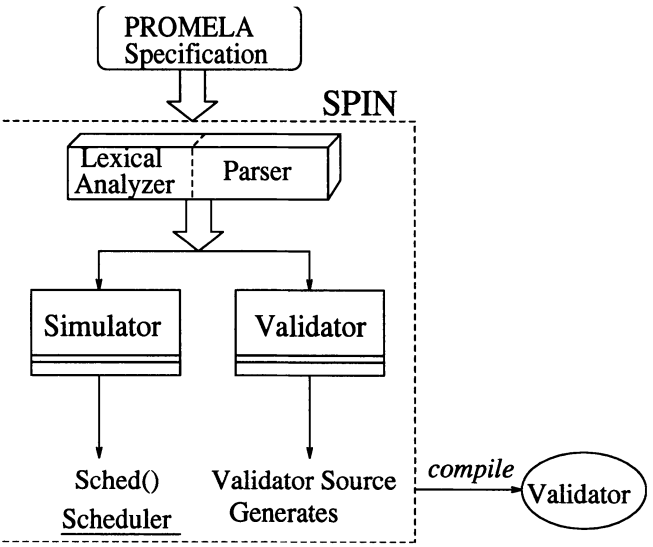

Figure 1: SPIN Structure

4 TIME FACILITIES

In this section, we propose *T-PROMELA* and *T-SPIN*, which introduce time extension of *PROMELA* and *SPIN*. *T-SPIN* is the tool that simulates the specifications written in *T-PROMELA*.

4.1 T-PROMELA

4.1.1 Extension of syntax

We extend *PROMELA* on the basis of the requirements described in 2.2. The syntax and examples of specifications that are extended from *PROMELA* are as follows;

- The elapsed time of execution of statement
  The pre-defined explicit elapsed time of the execution of statement can be described.

```
1 | sequence ::= step [t_exprs] {';' step [t_exprs]}
2 |      step ::= [decl_lst] statement
3 |  t_exprs ::= COST t_expr [':' t_expr]
4 |   t_expr ::= '('t_expr')'
5 |              | t_expr binop t_expr  ..(binop : +, -..)
6 |              | var_ref ..variable
7 |              | DCONST  ..real number
```

  Ex. *channel* ! *data*@10
  It takes 10(ms) to send *data* to *channel*.

- Transfer time
  In the declaration of a channel, users can specify the transfer time through the channel.

  ch_init ::= '[' *CONST* ']' *OF* '{' *TYPE* {',' *TYPE*} * '}' [*t_exprs*]

  Ex. *chan ipc_channel* = [2] *of* {*byte*, *byte*}@20
  The above description specifies the transfer time through channel *ipc_channel* as 20(ms).

- Time type variables
  Users can use variables preserving a real number for time specifications.

```
1 | one_decl ::= [TYPE ivar {',' ivar}* ]
2 |     ivar ::= var_dcl | var_dcl ASGN expr
3 |  var_dcl ::= NAME [ '[' CONST ']' ]
```

  Ex. *time timeout* = 5.0
  The initial value of time type variable *timeout* is specified as 5.0.

- Variable of time type *gtime*
  Users can specify the special variable *gtime* that indicates the time since the simulation has started.

  time gtime

  Ex. *cur_time* = *gtime*
  This expression means assignment of the present time(elapsed time) to the variable *cur_time* with time type.

4.1.2 Example

Using the extensions of syntax shown in 4.1.1, following examples can be described.

- Example of time-out

```
1 | chout!data;                          /* Sending */
2 | st_time = gtime;                     /* Preserving time of start */
3 | do
4 | :: chin?ack- > break                /* Receive ack */
5 | :: (gtime - st_time > 5000)- >       /* Supervising time-out */
6 |    printf("time - out occurs")- >    /* Time-out occurs */
7 |    break
8 | od
```

After sending *data* through channel *chout*, the system receives *ack* through channel *chin*. In this application, time-out is defined as 5000(ms). Supervising time-out is specified as the expression comparing (*the present time* − *time of start*) with 5000.

- Example of real time requirement using assertion

  Real time requirement is specified by using the expression meaning "a certain sequence of statements has to be completed within a specified time".

```
1 | st_time = gtime;                     /* Preserving time of start */
2 | chout ! req;                         /* Transaction processing */
3 | chin ? ack;                          /* Transaction completes */
4 | assert(gtime - st_time < 2000);      /* Assertion "Transaction has to be
5 |                                      completed within 2000(ms)" */
```

  When the transaction didn't complete successfully within 2000(ms), then the assertion of violation is reported to a user.

4.2 T-SPIN

T-SPIN is the extension with time, based on the simulator-part of *SPIN*, and simulates specifications written in *T-PROMELA*. *T-SPIN* presupposes the following.

1. Round robin scheduling
   T-PROMELA deals with concurrent processes by round robin scheduling.

2. *T-PROMELA* model
   T-PROMELA deals with the communication service within one machine(one CPU).

3. IPC
   The communication service in *T-PROMELA* is restricted to interprocess communications within one machine(one CPU).

4. Measuring elapsed time of executive statement
   When the elapsed time was specified, *T-SPIN* assigns it as the elapsed time of the statement. When the elapsed time wasn't specified, *T-SPIN* measures the actual elapsed time of the statement.

The reasons for 2 and 3 are that *T-PROMELA* can't allow specifying resources for each process described in *T-PROMELA*. Consequently all processes are created in single environment in single machine. *T-SPIN* structure is shown in Fig.2.

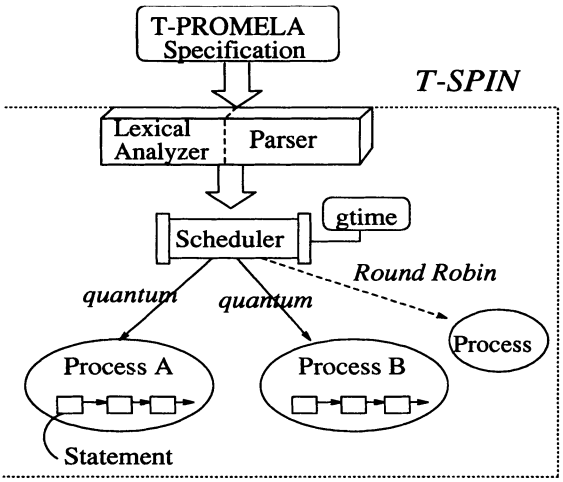

Figure 2: T-SPIN structure

5 APPLICATION TO CLIENT SERVER SYSTEM

In the previous section, *T-PROMELA*'s additional extended syntax, its applications and *T-SPIN* have been proposed. In this section, we discuss the application of *T-PROMELA* and *T-SPIN* to client server system.

5.1 Model and example of applied specifications

We apply *T-PROMELA* to a communication model of client server system using message queue, which is the same model of IPC(interprocess communications) facility. The facility allows asynchronous communication ; all messages are stored in and also passed through message queue.

The time aspect of this communication model is as follows; a client requests again when the reply from a server didn't arrive within claimed time($1000ms$). This communication model is shown in Fig.3. On this system, we suppose that one server and ten clients are created and each client sends request messages 50 times at one trial. The specification of *client* on this model is as follows.

```
1 proctype client( int pid; chan srv ){    /* specifications for client */
2 chan client_q = [4] of { byte } @0 ;     /* receive channel */
3 byte seq = 1, buf, ok = 0 ;              /* sequence number */
4 time st_time, begin  ;
5 begin = gtime ;                          /* start time of process */
6 do                                       /* request 50 times */
7 :: ( ok <  50 ) -> st_time = gtime ;     /* timer start */
8      ( count < MAXQ ) ->
9      atomic{ srv ! client_q( seq ) ->    /* send request */
```

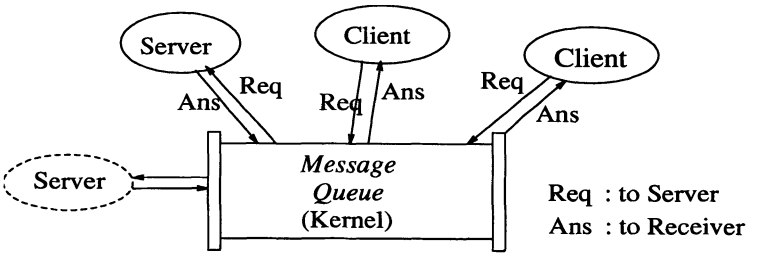

Figure 3: Client / Server Model

```
10              count = count + 1 } ;
11      do
12     :: atomic{ client_q ? buf ->       /* receive */
13                 count = count -1 } ->
14         if
15         ::( buf < seq )                 /* reply for previous request */
16         ::( buf== seq ) ->              /* normal reply */
17             ok = ok + 1 -> break
18         fi
19      :: ( gtime>st_time+tmout )->break  /* supervise time-out */
20      od ;
21      seq = seq + 1                      /* update sequence number */
22 :: ( ok >= 50 ) -> break                /* client ends */
23 od ;
24 printf("client [%d] completed  ok = %d fail = %d\ntime< start %f
25         ---end %f [%f]\n",pid,ok,seq-ok-1,begin,gtime,gtime-begin )
}
```

5.2 Simulation

There are mainly two functions of *T-SPIN* ; the tracing of the system behavior and the evaluation of the performance. Regarding the former, *T-PROMELA users* can use *T-SPIN*'s functions (e.g., the printing send/receive operation and global/local variables) to confirm their specifications. Regarding the later, when *T-PROMELA users* specified the printing commands for time parameters(e.g., line24 – 25 in the above specification), they can evaluate the performance of their specifications. The simulation result of the above specification is as follows;

```
tspin example
client [15] complete  ok = 50 fail = 0
time start 100.0 --- end 1460.0 [1360.0]
client [14] complete  ok = 50 fail = 0
time start 100.0 --- end 1560.0 [1460.0]
.........................
client [3] complete  ok = 50 fail = 1
time start 110.0 --- end 2771.0 [2661.0]
```

```
client [2] complete  ok = 50 fail = 1
time start 110.0 --- end 2871.0 [2761.0]
17 processes created
```

The report of 15 clients is printed. For example, client 14 completed 50 communications and didn't cause time-out(fail). It worked from 100(ms) to 1560(ms) and its elapsed time was 1460(ms). Clients 2 and 3 cause time-out and send its request again.

6 CONCLUSION

Using *T-PROMELA*, which has a time related extension facility based on PROMELA, can describe, validate and also evaluate specifications for communication services that includes real time specifications(e.g., Time-critical-communication services, real time communication services). The development of T-SPIN that simulates specifications written in *T-PROMELA* makes it possible to confirm the system behavior , to evaluate the performance and to test real time requirements.

For the time being, a model of specifications is targeted to communication services in a machine. *T-PROMELA* should allow multi-CPU model and deal with communication services on distributed systems. So, further extension of syntax and the method of implementation on Simulator are being investigated.

References

[1] M.T.Liu: Introduction to Special Issue on Protocol Engineering, IEEE Transactions on Computers,Vol.40,No.4,pp.373-375(1991.4).

[2] B.Sarikaya: PRINCIPLES OF PROTOCOL ENGINEERING AND CONFORMANCE TESTING, ELLIS HORWOOD(1993).

[3] K.Turner: USING FORMAL DESCRIPTION TECHNIQUES, WILEY(1993).

[4] Initial draft on Enhancements to LOTOS, Part 6 ISO/IEC JTC 1/SC 21 N8023(1993.11).

[5] A.M.Clenaghan: Mapping Time-Extended LOTOS To Standard LOTOS, FORTE91, pp.239-254(1991.10).

[6] A.Nakata, T.Higashino, and K.Taniguchi: LOTOS enhancement to specify time constraint among non-adjacent actions using 1st-order logic, FORTE93, pp.453-468(1993.10).

[7] G.J.Holzmann: Design and Validation of Computer Protocols, PRENTICE HALL(1991.5).

[8] N.Nakano: Time Critical Communication Architecture in Factory Automation,JSPE-IFIP WG5.3 Workshop on THE DESIGN OF INFORMATION INFRASTRUCTURE SYSTEMS FOR MANUFACTURING, pp.241-252(1993.11).

[9] C.A.R.Hoare: *CommunicatingSequentialProcesses*, Prentice Hall(1985).