

Experimental Evaluation of Simplified Verification Methods for Responsive Communication Protocols

Shin'ichi Nagano Tohru Kikuno

Department of Informatics and Mathematical Science
Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
Phone: +81-6-850-6568 Fax: +81-6-850-6569
E-mail: {s-nagano, kikuno}@ics.es.osaka-u.ac.jp

Abstract

We consider responsiveness verification of communication protocols on the client-server system CS_N , which consists of one client site and N ($N \geq 2$) server sites, under the assumption that a single transient failure occurs on any one server site. For the verification, a typical naive method *NAIV* generates exhaustively all abnormal system states reachable after the failure, and then calculates the recovery time needed for returning to a normal state. In this paper we propose two simplified methods (*RSYM* and *EST*) to reduce the overheads needed for the method *NAIV*. The first method (*RSYM*) takes notice of symmetric properties on the system CS_N and considers the case where a single transient failure occurs on a specific server site. The second method (*EST*) verifies the responsiveness on the restricted system CS_2 , and then estimates from its result the responsiveness on the system CS_N .

Then we perform simulation experiments to evaluate the proposed methods. The experimental results show that (1) The first method *RSYM* can reduce the overheads of the method *NAIV*, while keeping accuracy in evaluating the recovery time. (2) The second method *EST* succeeds in both reducing further the overheads of the method *RSYM* and assuring good approximation for the recovery time.

Key words: client-server system, communication protocol, responsiveness, verification.

1. Introduction

Generally, communication protocols with fault-tolerant and real-time properties are called responsive communication protocols, shortly responsive

protocols[3]. In order to design a responsive protocol, we must prove that the protocol can recover to a normal state within a permissible time, even when it transits to an abnormal state due to a failure[9]. Thus the responsiveness verification method must include the following two steps: (1) generating all abnormal states to which the protocol may transit due to a failure, and (2) calculating the recovery time required for returning from an abnormal state to a normal state. However, since there exist a lot of abnormal states to be considered, it is very difficult to check whether the designed specification is equipped with all necessary functions for recovery.

In this paper, we consider responsiveness verification of communication protocols on the client-server system CS_N , which consists of one client site and N ($N \geq 2$) server sites, under the assumption that a single transient failure occurs on any one server site. For the verification, there exists a typical naive method *NAIV* which generates exhaustively all abnormal system states reachable after the failure, and then calculates the recovery time needed for returning to a normal system state. In this paper, we propose two simplified methods (*RSYM* and *EST*) to reduce the overheads needed for the method *NAIV*. The first method (*RSYM*) takes notice of symmetric properties on the system CS_N and considers the case where a single transient failure occurs on a specific server site. The second method (*EST*) verifies the responsiveness on the restricted system CS_2 , and then estimates from its result the responsiveness on the system CS_N .

In order to evaluate the proposed methods, we perform simulation experiments using a connection establishment protocol. In the experiments, we measure the following four kinds of metrics: (*m1*) the number of generated system states, (*m2*) the memory size, (*m3*)

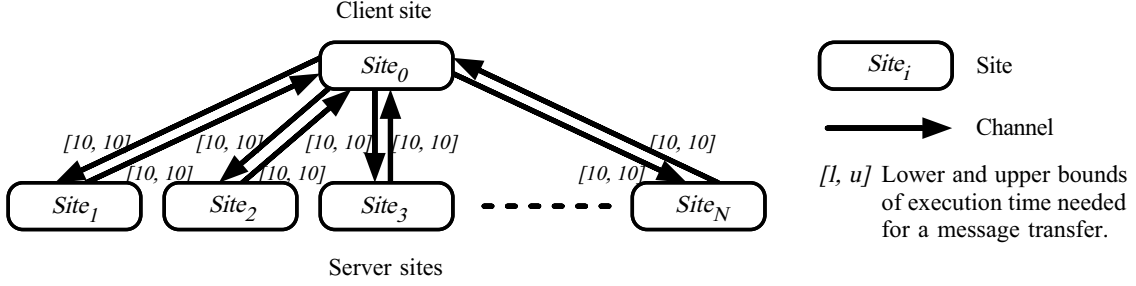


Figure 1. Example of network

the execution time, and ($m4$) the recovery time. The experimental results show that (1) The first method *RSYM* can reduce the overheads ($m1$ through $m3$), while keeping accuracy in evaluating $m4$. (2) The second method *EST* succeeds in both reducing further the overheads and assuring good approximation for $m4$.

2. Preliminaries

2.1. Communication Protocol

Generally, real-time communication systems such as plant control systems consist of client sites, server sites, and communication channels among them[6, 8]. In such a system, a client site and its server site must establish a connection between them before message passing. A series of interactions between sites for connection establishment is called a connection establishment protocol[1, 2].

The protocol is usually specified at three distinct levels: site level, process level, and EFSM level, hierarchically. At site level, each site is considered as a black box and only the interactions between sites through communication channels are specified. At process level, each site is specified explicitly using several processes and communication channels. However a process itself is still treated as a black box, and only the interactions between processes are described. Finally, at EFSM level, each process and channel are modeled by an EFSM and an FIFO queue, respectively[2, 7]. We assume that the lower and upper bounds of the execution time needed for each message transfer are given. For each transition in an EFSM, these lower and upper bounds are assigned as a label.

Figure 1 shows an example communication system CS_N which consists of one client site $Site_0$ and N server sites $Site_i$ ($1 \leq i \leq N$). Additionally, there exist $2N$ communication channels between $Site_0$ and $Site_i$'s, and the time needed for each message transfer

in the channels is exactly 10 time units. We discuss in this paper the connection establishment between a client site and a server site. Client site $Site_0$ consists of 5 processes and each server site $Site_i$ consists of 4 processes. We omit the descriptions at process level and EFSM level (please refer to [8]).

2.2. Responsiveness Condition

Definition 1 A system state ss_i at global time T in a communication protocol \mathcal{P} is defined as (gs_i, ge_i) . The component gs_i consists of the state variables for processes and channels in \mathcal{P} . The component ge_i is defined as $\{(e_1, l_1, u_1), \dots, (e_p, l_p, u_p), \dots, (e_{q_i}, l_{q_i}, u_{q_i})\}$, where l_p and u_p ($1 \leq p \leq q_i$) are non-negative integers and denote the lower and upper bounds of a residual time (see [7, 9]), respectively, needed for event e_p to be completed.

A normal system state is a system state reachable from the initial system state. On the other hand, an abnormal system state is any system state which is not a normal system state. We represent a set of normal system states and a set of abnormal system states as \mathcal{S}_{normal} and $\mathcal{S}_{abnormal}$, respectively.

Definition 2 A communication protocol \mathcal{P} is said to be a responsive protocol iff \mathcal{P} satisfies the following responsiveness condition **C**. Here, rt_i denotes a recovery time from the abnormal system state ss_i to a normal system state, and R is a positive integer representing a permissible recovery time.

Condition C: For any abnormal system state $ss_i \in \mathcal{S}_{abnormal}$, the following two conditions **C1**, **C2** hold: (**C1**) $\exists ss_j \in \mathcal{S}_{normal}$ [ss_j is reachable from ss_i], and (**C2**) $rt_i \leq R$.

2.3. Responsiveness Verification

The following (1)-(4) are given as the inputs for the responsiveness verification: (1) a communication protocol \mathcal{P} modeled by EFSMs, (2) the lower bound l_i and upper bound u_i of execution time of each event e_i , (3) the initial system state ss_0 , and (4) a permissible recovery time R . The output is *yes* or *no* (whether \mathcal{P} can recover to a normal system state within a given time limit R , or not).

The outline of the basic method for responsiveness verification is summarized as follows (please refer to [9] for the details):

Step 1: Calculate the set \mathcal{S}_{normal} of the protocol \mathcal{P} using the timed reachability analysis[7].

Step 2: Generate a set of initial abnormal system states, denoted by $\mathcal{S}_{abnormal}^0$, to which \mathcal{P} may transit due to an undefined transition, and then set $\mathcal{S}_{abnormal} = \mathcal{S}_{abnormal}^0$. For each $ss_p \in \mathcal{S}_{abnormal}$, generate all abnormal system states reachable from ss_p using the timed reachability analysis, and append them to $\mathcal{S}_{abnormal}$.

Step 3: Check whether R and $\mathcal{S}_{abnormal}$ obtained at Step 2 satisfy the responsiveness condition C .

2.4. Simulator for Responsiveness Verification

We have developed a simulator based on the basic responsiveness verification with C language and Tcl/Tk (about 10,000 lines)[9]. The simulator consists of the following three modules:

Editor: This editor supports drawing EFSMs for processes on GUI, which can be saved as the input for generation of system states.

Generation of system states (for Steps 1 and 2): The sets \mathcal{S}_{normal} and $\mathcal{S}_{abnormal}$ are calculated using the timed reachability analysis method. In order to reduce drastically the time needed for searching the equivalent system states, we adopt hashing as data structure of system states.

Responsiveness verification (for Step 3): This module checks whether $\mathcal{S}_{abnormal}$ and R satisfy the responsiveness condition C or not, and outputs the result (yes or no).

3. Simplified Verification Methods

3.1. Assumptions

Assumption 1 (System model) The communication system CS_N is composed of one client site and N server sites, as shown in Figure 1. The connection establishment protocol is designed on the system CS_N , and the descriptions of server site $Site_i$'s ($1 \leq i \leq N$) at process level and EFSM level are the same.

Assumption 2 (Fault model) A single transient failure occurs in any server site $Site_i$ ($1 \leq i \leq N$). No other failures occur within the system (even during the recovery from abnormal system state).

Assumption 3 (Execution time)

All sites $Site_i$ ($0 \leq i \leq N$) start their executions at time 0. Reception of a message in $Site_0$ takes two time units.

3.2. Three Methods for Verification

We first present a naive method *NAIV* for responsiveness verification.

- (1) Naive verification (*NAIV*): By generating exhaustively all system states on the system CS_N , calculate the sets \mathcal{S}_{normal} and $\mathcal{S}_{abnormal}$. Then check the responsiveness condition C .

Next, we propose two simplified methods to improve the overheads of the method *NAIV*.

- (2) Reduced verification based on symmetric properties (*RSYM*): By assuming that a single transient failure occurs on a specific server site (rather than any server site) on the system CS_N , say $Site_1$, calculate the sets \mathcal{S}_{normal} and $\mathcal{S}_{abnormal}$. Then check the responsiveness condition C .
- (3) Estimation based on transformed simple model (*EST*): Assume the restricted system model CS_2 (rather than CS_N) and assume that $10 + 2(N - 2)$ time units are needed for a message transfer between $Site_0$ and $Site_i$ ($i = 1, 2$). Then calculate the sets \mathcal{S}_{normal} and $\mathcal{S}_{abnormal}$ on CS_2 . Based on these sets, check and estimate the responsiveness condition C on the target model CS_N .

We explain the term $2(N - 2)$ in the proposed method *EST*: In the CS_N the messages from $Site_j$ ($1 \leq j \leq N$) may reach $Site_0$ at the same time. The delay in the response to $Site_j$ from $Site_0$ becomes maximum when $Site_0$ receives the message

from $Site_j$ after receiving the messages from all other $Site_k$'s ($1 \leq k \leq N, k \neq j$). According to Assumption 3, the maximum delay is estimated at $2(N - 1)$. Thus the difference of the maximum delays between the systems CS_N and CS_2 is $2(N - 2)$.

4. Simulation Experiments

In order to evaluate the proposed methods $RSYM$ and EST , we perform simulation experiments using a connection establishment protocol for a certain plant control system. In the experiments we measure the following four kinds of metrics:

- (m1) the number of system states generated by the simulator.
- (m2) the memory size needed by the simulator.
- (m3) the execution time needed by the simulator.
- (m4) the recovery time calculated or estimated by the simulator.

The simulator is executed on SUN UltraSPARC UA1 workstation with Solaris 2.5 augmented by 448 MB RAM.

We consider the following objectives for the simulation experiments:

- (1) To show that the simplified method $RSYM$ reduces drastically the values of metrics $m1$, $m2$, $m3$ compared with the method $NAIV$, but obtains the same value of $m4$ as method $NAIV$.
- (2) To show that the estimation method EST reduces further the values of metrics $m1$, $m2$, $m3$ compared with the method $RSYM$, but assures the small estimation error for the value of $m4$.

5. Analysis of Experimental Results

The experimental results are shown in Figure 2. Figures 2 (a), (b), (c), (d) show the number of system states ($m1$), the amount of memory ($m2$), the execution time ($m3$), and the recovery time ($m4$), respectively. Based on these results, we discuss the objectives (1), (2) mentioned in Section 4.

5.1. Comparison between $RSYM$ and $NAIV$

In Figures 2 (a), (b), (c), the values by the both methods $RSYM$ and $NAIV$ grow exponentially with respect to the number of server sites. However, the values by the method $RSYM$ is almost $1/N$ times as

large as that by the method EST , where N denotes the number of server sites.

The reason of this reduction is that the topology of the system CS_N is symmetric as shown in Figure 1, and that client site $Site_0$ deals fairly with all server sites $Site_i$'s ($1 \leq i \leq N$). Furthermore, since all server sites $Site_i$'s are modeled by the same EFSM, the occurrence of the failure in any server site is also symmetric. Using these symmetric properties, the method $RSYM$ can decrease drastically the number of system states.

On the other hand, these symmetric properties mean that the maximum recovery time is the same for any case of the failure in server site $Site_i$ ($1 \leq i \leq N$). Thus, the maximum recovery time calculated by the method $RSYM$ is the same as that by the method $NAIV$.

According to the above discussions, we can conclude that objective (1) is proved affirmatively.

5.2. Comparison between EST and $RSYM$

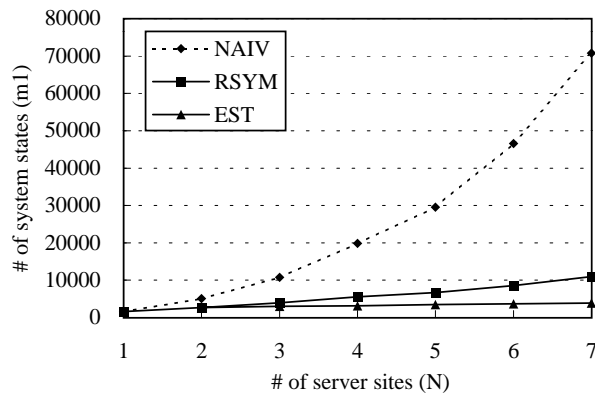
Figures 2 (a), (b), (c) show that the values by the method EST are almost located on the horizontal line, while the values by the method $RSYM$ grow exponentially. The reason of the horizontal line is that the method EST estimates the system CS_N ($N \geq 3$) quite well using the system CS_2 . Therefore, the method EST can reduce drastically the values of $m1$, $m2$ and $m3$.

The negative side-effect of such a drastic reduction appears clearly in the computational error for the maximum recovery time calculated by the method EST . But, Figure 2 (d) shows that the difference of the maximum recovery time between the methods EST and $RSYM$ is at most four time units.

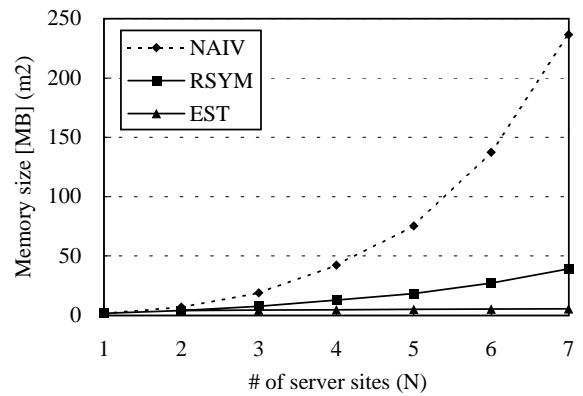
From the analysis of experimental results, we can conclude the objective (2) is also proved affirmatively.

6. Conclusion

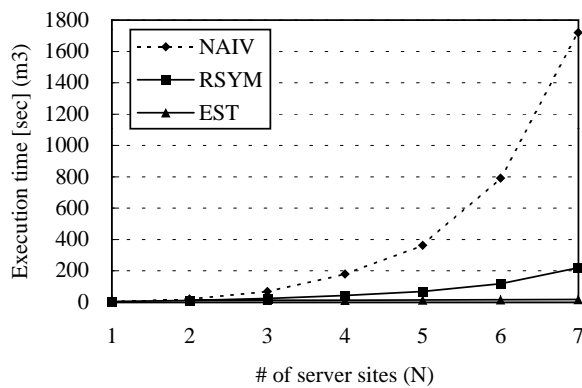
We have proposed two simplified methods for responsiveness verification: the method $RSYM$ and the method EST . The experimental results show that we can apply the method $RSYM$ to critical systems effectively. Then we find that the method EST is applicable at the early test phase iteratively, since it gives a good approximation in a very short time. Future research works include the further analysis of the proposed methods and the applications of the proposed methods to practical design problems.



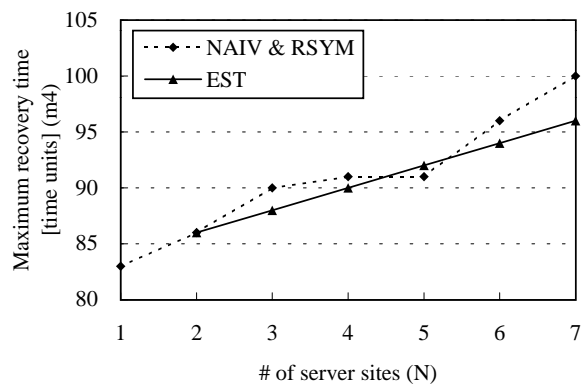
(a) The number of system states



(b) Memory size



(c) The execution time



(d) The maximum recovery time

Figure 2. Comparison among three methods

References

- [1] D. Ferrari and D. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, 1990.
- [2] G. Holzmann. *Design and Verification of Computer Protocols*. Prentice Hall, 1991.
- [3] H. Kopetz and Y. Kakuda, eds. *Responsive Computer Systems*, volume 7 of *Dependable Computing and Fault-Tolerant Systems*. Springer-Verlag, 1993.
- [4] Z. Liu and M. Joseph. Verification of fault tolerance and real time. In *Proc. of the 26th International Symposium on Fault-Tolerant Computing (FTCS-26)*, pp.220–229, 1996.
- [5] N. Lynch. Simulation techniques for proving properties of real-time systems. In S. Son, editor, *Advances In Real-Time Systems*, chapter 13, pp.299–332. Prentice Hall, 1995.
- [6] I. Mizunuma, C. Shen, and M. Takegaki. Middleware of distributed industrial real-time systems on ATM networks. In *Proc. of the 17th IEEE Real-Time Systems Symposium*, pp.32–38, 1996.
- [7] S. Nagano, Y. Kakuda, and T. Kikuno. Timed reachability analysis method for communication protocols modeled by extended finite state machines. *Trans. of IPSJ*, 37(5):698–710, 1996.
- [8] S. Nagano, Y. Kakuda, and T. Kikuno. Experience of responsiveness verification for connection establishment protocols. In *Proc. of the 1st IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC'98)*, pp.383–392, 1998.
- [9] S. Nagano, Y. Kakuda, and T. Kikuno. A new verification method using virtual system states for responsive communication protocols and its application to a broadcasting protocol. *IEICE Trans. on Fundamentals*, E81-A(4):103–111, 1998.