

Formal Specification and Verification of a Data Replication Approach in Distributed Systems

ALIREZA SOURI

Department of Computer Engineering, Hadishahr Branch, Islamic Azad University,
Hadishahr, Iran

Data replication is an important optimization phase to manage large data by replicating data in various distributed systems. In distributed systems, reliability and performance are two important factors for using data replication. Also model checking techniques can be used to verify the correctness of these systems. In this paper, a Data Replication approach has been proposed. This paper presents a behavioral modeling the proposed approach with the goals of providing correctness and reducing verification time and memory consumption. Evaluating and analyzing the logical problems such as deadlock free, reachability and fairness for the considered data replication approach are provided. For verifying the behavioral models of the proposed data replication approach the NuSMV model checker is employed. The verification results are compared by user graphical interface and Kripke model verification methods.

Keywords: Distributed systems, Data replication, Formal verification, Behavioral model, Model checking.

1. INTRODUCTION

Distributed systems are popular in the industries and technology, as well as economy, management, research centers, military and medical organizations [Garcia-Garcia et al. 2012]. One of the important factors in distributed system architecture is the presence of multiple replicas of big quantities of data [Gray and Reuter 1992]. In an active distributed system, when a sequence updates happen, the newness term of a replica is critical [1]. Also some articles and researches evaluate their approaches only without considering a specific replication model using simulation and traditional experiments. On the one hand, in these experiments they cannot specify that how replication model and its performance are suitable for data replication architecture exactly [Souri and Rahmani 2014]. On the other hand, by using the simulation results the all of the state spaces of the problem is not checked and analyzed well. So, to resolving these problems formal verification as a powerful method for verifying and model checking of distributed systems is an appropriate methodology. Formal verification of software and hardware systems is an important subject in the many researchers studies them.

The model checking technique is very marvelous because of its simplicity of use joint with a compact theoretical foundation on verification approach. While it does not require high skilled specialists as usually needed for theorem proving, it offers a very expressive resource for properties specification through temporal logic. Also Model checking is an important technique for the verifying the distributed and software systems [Clarke Jr. et al. 1999]. It has some advantages over testing, simulation and empirical reasoning. The model checking procedure contains three phases: Modeling, Specification, and Verification. In first phase, the proposed model is normally created using an FSM-based formalism by a model checking tool such as PAT [Souri and Norouzi 2015] [Liu et al. 2010], Spin [Holzmann 1997], UPPAAL [Hammal et al. 2015] and SMV [McMillan 1992]. Then, a set of expected properties of the system define to satisfy these properties by using some temporal logic languages [Dwyer et al. 1998]. After checking an expected property, there are two possible conditions: the property is satisfied by the model that means the model does not contain any misbehavior, and the property violated by the model means that one behavior

in the model is not authorized by the specification. Then, the model checker tools generate a counter-example as output, which shows the behavior that violated the property [Manna and Pnueli 1995][Souri and Jafari Navimipour 2014].

Formal verification is used to mathematical demonstration of the correctness for a system. Formal verification methods have an important role in validation processes of systems [Safarkhanlou et al. 2015]. Formal verification of consistent replication approach is important by means of it can perceive strategy flaws that lead to propagation failure [Zhu and Wang 2010].

For example, Prez, Garca-Carballeira [Pérez et al. 2010] proposed a new replication method, called the Branch Replication Scheme (BRS) in large-scale distributed systems such as data grid. An analytical model of the replication structure and replica updating structure are formally described in this research. Using this model, operations such as write, read or process of a replica are examined. In the proposed model, each replica is composed of a different set of sub-replicas organized using a hierarchical topology. They simulated the proposed model using Omnet++¹ tool. Simulation results demonstrated the probability of BRS, they display that the proposed replication system increases data access, equaled by other replication systems such as hierarchical and server replication that normally used in data grids. Their method provided three main advantages over traditional approaches: optimizing storage usage, by creating sub-replicas; increasing data access approach, by relating parallel input or output techniques; and providing the possibility to modify the replicas, by preserving consistency amid updates in a competent method. The disadvantage of this research, using a simulator for showing experimental results, because all of the state-space of model cannot analyzed very well. This research has not specified the data update procedure in sub-replicas. Also the authors are not discussed about consistency models guarantee. So, it is not specified that how consistency model is support by their model. Abawajy and Mat Deris [Abawajy and Mat Deris 2013] proposed a new quorum-based data replication protocol with the objectives of minimizing the data update rate, in case great accessibility and data consistency. They compared the proposed approach with Grid structure and Read-Only-Write-All approaches using reply time, data availability, data consistency and communication costs. First, they formulated the data replication problem and designed a distributed data replication algorithm with consistency guarantee for data grid. So, they presented a replica placement policy, which determines how many replicas to create and where to place the replaces; A replica consistency control policy, which determines the level of consistency among data replicas; Investigate various tradeoffs in terms of cost, availability and algorithm complexity of the proposed replication scheme; and compare both theoretically and empirically, the response time, availability, communication overhead, and consistency guarantees of the proposed protocol with two other protocols. First advantage of this research is using quorum-based data replication protocol for guaranteeing all of read and write operations in a system with failure. Of course, they used GridSim² tool for showing experimental results. But, they did not specified that how consistency model is support by their model. Also the guaranteeing consistency between read and write operation in failure condition is not shown. In our research, we considered timed atomic broadcast and deferred update protocols for semi-active data replication protocol that the consistency guarantees supports when a failure or leader death is occurred. Principally for the update replication methods, most researchers discuss about proofs of their methods in a natural language report [Kemme and Alonso 2000][Garcia et al. 2011]. By notice the complexity of protocols and concurrency problems in execution of database systems, a formal verification approach is necessary. Armendriz-Iigo, Gonzalez de Mendvil [Armendáriz-Iñigo et al. 2009] presented a formal description and correctness proof for replication database systems. Also they explain database replicas and the causal replication protocol. They consider specifications for atomic broadcast communication as non-serialize methods and database consistency provided by a certification-based protocol. They used I/O automata to illustrate behavior system and

¹www.omnetpp.org

²www.buyya.com/gridsim simulator

verifying system properties.

To the best of our knowledge and as discussed and mentioned in this section, all the related works and researches on this scope have some defects [Souri and Pashzadeh 2014][Alami Milani and Jafari Navimipour 2016]. First is that there are not any detailed papers and research that considered the replica failure using formal verification and behavioral modeling in data replication protocols. Also in all works and research of distributed systems are evaluated only by simulation or traditional case study, therefore, it is possible that some part of the state spaces of the problem is not analyzed and checked well. To overcome this defect, formal verification and behavioral modeling approaches as a powerful technique for the verification of the systems are employed in our research. This paper presents behavioral modeling of a Combined Semi-Active Data Replication approach (CSADR) by using model checking techniques in distributed systems. This approach is based on combination of the Atomic Broadcast (AB), View Synchronous Broadcast (VSB) and Deferred Update Replication (DUR) protocols to ensuring correct progress of the replicas. In particular, this paper separates data replication behaviors into two types: propagation behavior and logical behavior based on behavioral modeling approaches [Hansen et al. 2003]. The relations between the two behaviors are modeled as mapping process. By analyzing logical problems and checking behavior specifications, it is possible to verify the CSADR approach. Especially, the contributions of this paper are:

- Proposing a Combined Semi-Active Data Replication approach (CSADR) based on the Atomic Broadcast (AB), View Synchronous Broadcast (VSB) and Deferred Update Replication (DUR) Protocols.
- Presenting a behavior model to separating propagation and logical behavior of CSADR approach. The separation of these behaviors enables the procedure of verification, maintenance and development of the data replication approach.
- Enabling the mapping process between two behaviors by means of the formal verification approach based on Binary Decision Diagram (BDD) [Clarke Jr. et al. 1999]. This formal method excerpts the expected properties of the data replication approach from logical behavior in the form of Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) formulas, and verifies the properties in the propagation behavior expansively.
- Evaluating the logical problems such as reachability, fairness and deadlock free for the considered data replication approach.
- Verifying the behavior models of the proposed approach by using the NuSMV model checker.

The rest of this paper is organized as follows. Section 2 presents a data replication approach based on semi-active replication in distributed systems. Section 3 describes the propagation and logical behaviors as well as defining some essential concepts and notations to formalize these behaviors. Section 4 presents a symbolic model checking approach for the proposed behavioral models. Also the behavioral models have converted to Kripke structure for checking some behavioral problems such as reachability, fairness and deadlock free automatically. Furthermore, the logical properties of behavioral models are defined by using linear temporal logic and computation tree logic languages. These properties can be checked by the specification of logical behavior which is mapped on propagation behavior. Section 5 focuses on the implementation of the proposed behavioral models by NuSMV (GUI) tool and verification of the Kripke structure model by NuSMV model checker. In section 6, a performance evaluation is presented for proposed approach. Finally, conclusions and future works are provided in Section 7.

2. PROPOSED DATA REPLICATION APPROACH

In this section, we present a Combined Semi-Active Data Replication approach (CSADR) based on the Atomic Broadcast (AB), View Synchronous Broadcast (VSB) and Deferred Update Replication (DUR) protocols. First, we discuss the replication context, semi-active replication and

differed update protocol. Then, we describe the CSADR approach with added properties in proposed approach.

2.1 The context of replication protocol

We briefly describe semi-active replication protocol according to [Schneider 1990][Powell 1994]. Semi-active replication is a middle explanation between active replication and passive replication. Semi-active replication has not need to a deterministic method in process of service request replicas. For a semi-active replication protocol, there are five generic stages. These stages represent the procedure of update operation in the protocol and have used to characterize the different methods according to [Wiesmann et al. 2000].

Request Phase (RE): in this phase, a client submits a request to all replicas by using Atomic Broadcast.

Server Coordination Phase (SC): During the server coordination phase, the replicas coordinate by using the order given by the atomic broadcast protocol.

Execution Phase (EX): In this phase, all replicas execute the submitted request in the order they are delivered.

Agreement Coordination (AC): this phase can navigate in case of a non-deterministic order to guaranteeing atomicity. In first phase a leader is chosen and this leader informs to the other replicas (as followers) using the View Synchronous Broadcast (VSB) protocol. This method is similar to the two phase commit in eager update everywhere with distributed locking approach.

Client Response (CL): The client response phase shows the send back operation in time when the client receives a response from the system. The replicas return the reply to the client.

By notice to the above steps, we can specify some problems to phases of semi-active replication protocol [Poledna 1994]. First, by using atomic broadcast in the SC phase we can guarantee three properties such as agreement, validity and integrity. Due to the high number of read and write operations, the execution time of each operation is a critical factor in distributed systems. Therefore, using a timed atomic broadcast protocol can support this problem. Second, when crash and failure conditions have been occurred for the selected replica as a leader, choosing a leader can be a critical point before sending a request in RE phase. So, the failure of replicas is limited since the follower replicas cannot detect out of order decisions of the leader. This selection causes that the system cannot have an appropriate behavior for choosing a leader in critical times. Third, there is a cycle between the EX and AC phases for coordination process and delivery process for all of replicas. In the nondeterministic point of execution phase, the update procedure has occurred according to VSB order in AC phase. If a replica encountered by failure in spread on the status, the agreement coordination phase cannot confirm the entire replica for responding to the client. Therefore, using a view delivery method is important for preventing this problem.

To overcome these defects, we present a combined semi-active replication approach in this section. So, not only this approach supports all of the factors of traditional semi-active replication but also prevent from the above defects.

To define the CSADR approach, we add a stage to the generic stages of traditional semi-active replication protocol. Also we use to Timed Atomic Broadcast (TAB) and Same View Delivery (SVD) protocols [Sciascia and Pedone 2013] in the proposed approach.

Figure 1 depicts 6 stages of the replication process in the CSADR approach. Before sending each request by the client, a bounded time is specified for broadcasting the request to replicas in stage 1. After specifying the bounded time, the client sends the request to the replicas by using a timed atomic broadcast in stage 1. By using the order property of the timed atomic broadcast, the replicas coordinate in stage 2. In stage 3, the execution process is performed for updating replicas. The replicas execute update process. Because execution stage navigates the updates of replicas in nondeterministic point, each replica that completes its update operation in minimum time, it is determined as the leader. Also other replicas are followers.

After specifying the leader in case of a nondeterministic choice, the leader notifies the result of

its update to the followers by using VSB in stage 4. This stage navigates the agreement coordination for the followers. In stage 5, each replica sends apply message to the leader using same view delivery (SVD). Finally, the leader sends back the response to the client in stage 6. This reply specifies that all of the replicas update itself according to the request of client.

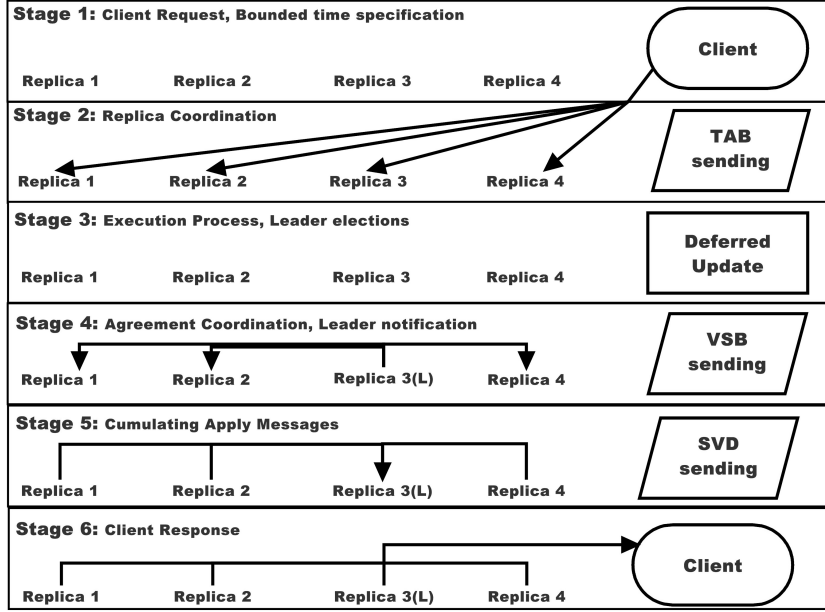


Figure 1. Combined semi-active replication approach.

3. DATA REPLICATION BEHAVIORAL MODEL

This section presents a behavioral model for the CSADR approach that includes: Propagation behavior and Logical behavior. First, we define a formal explanation of presented behavioral model for separating this model into Propagation behavior and Logical behavior.

Definition 1: The Combined Semi-Active Data Replication (CSADR) system behavior is a 4-tuple $CSADRB = (A, a, L, T)$ where:

- A is a finite set of states.
- $a \in A$ is the initial state.
- L is a set of transition labels.
- $T \in A \times L \times A$ is the transition relation. For illustrating a transition relation T , let $T = (a_i, l, a_j)$ that a_i and $a_j \in A$, and $l \in L$. This transition relation demonstrates the current relation between a_i and a_j by label l .

Figure 2 explains the Propagation behavior of the CSADR approach that includes client request sent, TAB propagated as the module 1, deferred update preceded as the module 2, VSB propagated as the module 3, SVD applied as the module 4 and client request received. The first input for module 1 is Δt for timed atomic broadcast process. An output is determined for module 2 after specifying the leader R in leader election process. The specified leader R is the input of module 3. After receiving all of the apply messages of followers by Leader R in SVD process, the output of module 4 is the final result. Finally, the final result of the leader R is sent

to the client.

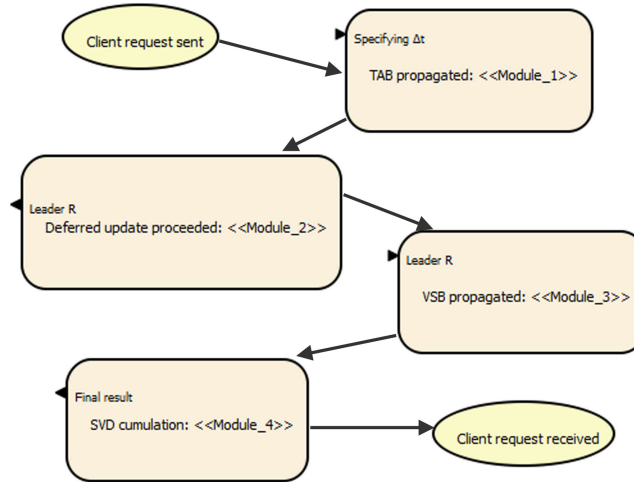


Figure 2. The propagation behavior of the CSADR approach.

Figures 3, 4, 5 and 6 show the modules of propagation behavior in CSADR approach. Figure 3 shows the module 1. In this module, the client sends the request for propagating the update to other replicas. Then, TAB protocol has executed the by determining a Δt . If the $Time-set \leq \Delta t$, then TAB protocol completes the propagation process by *TAB-Delivery*. Otherwise, the request suspended and rollback to the *Sendback-client* state. After delivery of timed atomic broadcast protocol, the result of this module transfers to module 2 (Figure 4). This result depicts the replica coordination process. Figure 4 shows module 2. This module illustrates the deferred update and leader specification respectively. After specifying leader and followers, the agreement coordination has started in next module. Figure 5 describes module 3. This module shows the VSB protocol. In this module, the leader notification process is done. Finally, figure 6 depicts module 4. This module illustrates the SVD protocol for apply messages of followers and cumulating these messages by the leader. After performing the accumulation of applies messages, the leader sends the response request to the client.

The logical behavior navigates the execution flow of the update propagation system. In the figure 7, the logical behavior of CSADR approach presents a number of states extracted from CSADR behavior. These states includes: Enabled, Received, Transferred, Executed, Suspended and Done. Initially, the logical behavior is enabled. In the propagation behavior, the Delta-t-determined, *TAB-Delivery*, *Followers-determined*, *First-update-completed* and *Leader-delivery* operations have been executed by Received state. The deferred update process, Leader specified, Leader notification and cumulating messages operations have been executed by Executed state. The Transferred state executes all of the *TAB-send*, *VSB-send* and *SVD-send* operations. The Suspended state executes the suspended operation. Finally, the *sendback-client* operation has been performed by Done state. When the propagation behavior procedure is accepted, then the Executed state moves to Done state. Otherwise, Executed state sends back to the Transferred state and the Transferred state moves to Suspended state and the procedure rollbacks to the Enabled state.

In figure 7, the state done is final state of the logical behavior. For example, in the figure 7 a path is: *Enabled* → *Received* → *Executed* → *Transferred* → *Received* → *Executed* → *Done*.

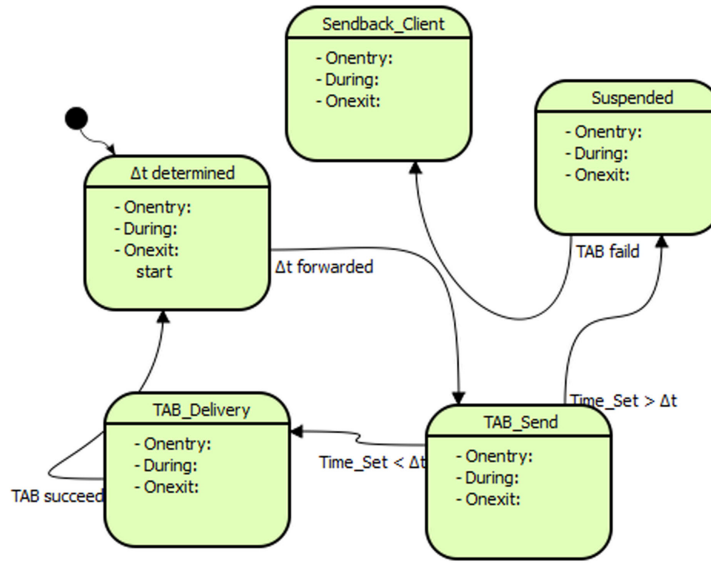


Figure 3. The module 1 of propagation behavior.

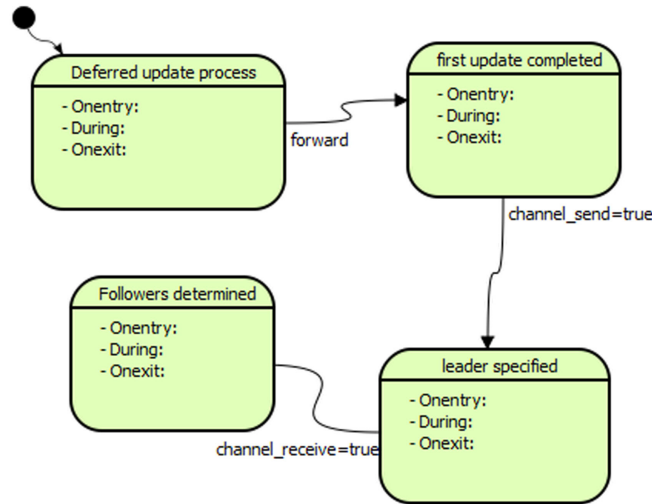


Figure 4. The module 2 of propagation behavior.

In this path the states are terminated to Done state at the *end*. There is a loop for following path: *Received* → *Executed* → *Transferred* → *Received* → *Executed* → *Transferred* → *Received*.

Definition 2 (Path in the CSADR Behaviors): A path $P^{i \rightarrow j}$ in the CSADR behavior is a finite sequence of the states and transitions starting from state a_i and finishing at state a_j , denoted as:

$P^{i \rightarrow j} = a^i \rightarrow a^{i+1} \rightarrow a^{i+2} \rightarrow \dots \rightarrow a^{j-1} \rightarrow a^j$ such that $\forall k \in (i, j-1) : (a^k, l, a^{k+1}) \in T$. For example, in figure 6, SVD-Send → Leader-Delivery → Cumulating-messages → Sendback-Client is a path in the propagation behavior of the CSADR approach.

There are a set of possible mapped paths in the propagation behavior with each state of the logical behavior. In following example, we see a mapping method in the CSADR approach where

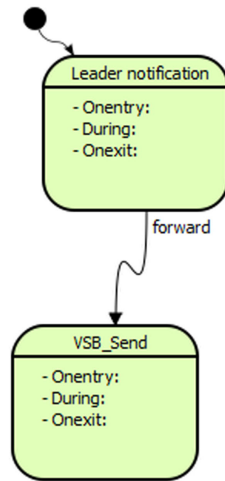


Figure 5. The module 3 of propagation behavior.

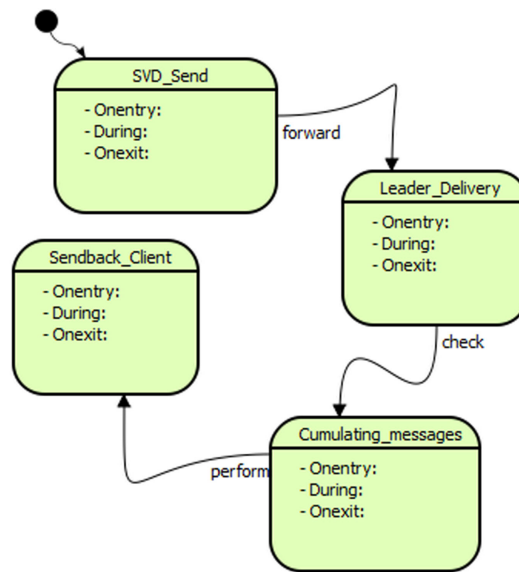


Figure 6. The module 4 of propagation behavior.

the Executed and Received states in the logical behavior are connected with some states in the propagation behavior as follow:

- **Propagation behavior:** *Deferred update process* → *First update completed* → *Leader specified* → *Followers determined*.
- **Logical behavior:** *Executed* → *Received* → *Executed* → *Received*.

4. SYMBOLIC MODEL CHECKING FOR BEHAVIORAL MODELS

In this section, we define the temporal logic languages such as the Linear Temporal Logic (LTL) and the Computation Tree Logic (CTL) formulas briefly. Also we illustrate converting behaviors of CSADR approach to Kripke structure mechanism.

There are two temporal logic languages for symbolic model checking that include: Linear Tem-

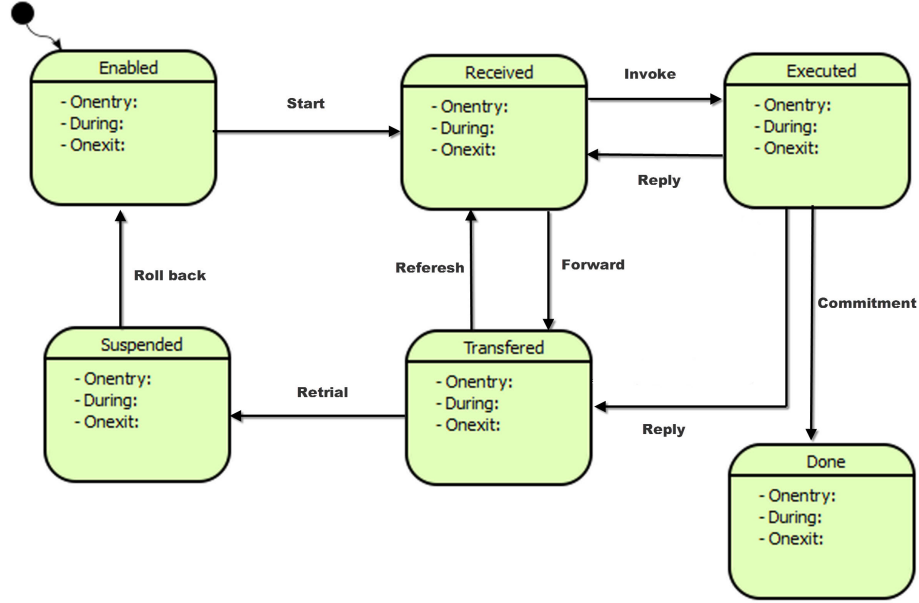


Figure 7. The logical behavior of the CSADR approach.

poral Logic (LTL) and Computation Tree Logic (CTL)[Clarke Jr. et al. 1999]. For showing the properties of the propagation behavior, these properties can formulate by using LTL and CTL formulas [Jafari Navimipour et al. 2015].

Now, we define some LTL and CTL properties for propagation model according to above rules and we let \rightarrow as the logical association:

- $L1: G (DeferredUpdate.state = DeferredUpdate-process) \rightarrow F (DeferredUpdate.state = Followers-determined)$. Globally a Followers-determined state comes in future after a DeferredUpdate-process state finally.
- $L2: G (DeferredUpdate.state = Leader-specified) \rightarrow F (VSB.state = VSB-send)$. Globally a VSB-send state comes in future after a Leader-specified-process state finally.
- $L3: G (TAB.state = Suspended) \rightarrow X (TAB.state = Sendback-Client)$. Globally a Sendback-Client state comes after a Suspended state.
- $L4: G (VSB.state = Leader-notification) \rightarrow X (VSB.state = VSB-send)$. Globally a VSB-send state comes after a Leader-notification state.
- $C1: AG (TAB.state = TAB-delivery) \rightarrow AF (TAB.state = Sendback-Client)$. There are always all paths from state TAB-delivery to state Sendback-Client in future.
- $C2: EF (TAB.state = TAB-delivery \rightarrow VSB.state = SVD-send)$. There is eventually a path from state TAB-delivery to state SVD-send finally.
- $C3: AG (VSB.state = VSB-send \rightarrow TAB.state = Suspended)$. There are always all paths from state VSB-send to state Suspended that it is not true.
- $C4: AG (TAB.state = Sendback-Client \rightarrow SVD.state = Sendback-client)$. There are always all paths from state Sendback-Client to state Sendback-client globally.
- $C5: AG (SVD.state = Leader-delivery) \rightarrow AX (SVD.state = Cumulating-messages)$. The state Cumulating-messages comes after state Leader-delivery always.
- $C6: AG (TAB.state = TAB-Send) \rightarrow EX (TAB.state = TAB-delivery \mid TAB.state = Suspended)$. The states TAB-delivery or Suspended comes in future after state TAB-Send eventually.

Also we can define logical properties to specify the logical behavior states for checking by using following initials: Initial= (Disable: Di; Received: Re; Executed: Ex, Transferred: Tr, Suspended: Su; Done: Do, Finish: Fi). Also we Let \rightarrow as the logical association. We define some examples of LTL and CTL properties which can be verified for the control behavior:

- L1: $G (Di \rightarrow X Re)$. Always a Received state comes after a Disabled state.
- L2: $G (Su \rightarrow X Do)$. Always a Done state comes after a Suspended state.
- L3: $G (Ex \rightarrow F Tr)$. Always a Transferred state comes in future after an Executed state.
- L4: $G (Re \rightarrow F (Tr | Do | Ex))$. Always a Done state or Transferred state or Executed state in the future comes after a Received state.
- C1: $AG (Di \rightarrow EF Do)$. There is always a path from state Enable to state Done eventually.
- C2: $AG (Re \rightarrow AF (Ex | Tr))$. There is always a path from state Received to state Executed or Transferred.
- C3: $AG (Ex \rightarrow EF Do)$. There is always a path from state Executed to state Done.
- C4: $AG ((Di \rightarrow EF Fi))$. There is always a path from state Disable to state Finish eventually.
- C5: $AG EF (Re \rightarrow Do)$. The path form Received to Done is always potentially reachable.
- C6: $AGEF (Fi)$. State Finish is always potentially reachable.

4.1 Kripke structure of CSADR approach

In this subsection, we define the Kripke structures of behavioral models. A Kripke structure is used to checking the system behavior.

Definition 3 (Kripke structure): A Kripke structure as a non-deterministic finite state machine is a 4-tuple $= (H, I, F, Q)$, where:

- H is a finite set of states
- I is a set of initial states.
- F $H \times H$ is a transition relation for $\forall h_1 \in H, \exists h_2 \in H : (h_1, h_2) \in F$.
- Q: 2^{AP} is a labeling functions true or false which each state with the atomic propositions holds in that state. AP is a nonempty set of atomic propositions. The Q illustrates to each state $h \in H$ that set Q (h) of all atomic propositions that are valid in h.

For verifying the propagation behavior, there are some problems such as state space explosion. To prevent the state space explosion, we convert the propagation behavior to a Kripke structure. We present a conversion of the propagation behavior into a Kripke structure. This conversion is performed as follow: Each state a in the propagation behavior is converted into a set of states and transition in the Kripke Model *KM* and each transition is converted into one or many transitions. If a is a simple state, it is converted to one state in *KM* with the same content. In this conversion, we follow propagation behavior states to the corresponding state in the logical behavior. This conversion is from propagation behavior states to the symbols Di, Re, Ex, Tr, Su, Do and Fi which these symbols are executed by using mapping process in definition 5. In figure 8, we illustrate converting the propagation behavior states to a Kripke structure. In this figure, the state Disable: Di is initial state and Finish: Fi is final state in Kripke structure. Figure 9 shows the states of propagation Kripke model with renaming by symbol names of logical behavior. Then a reduction of Kripke model for propagation Kripke model is provided by a decrease approach based on reduction of BDD. This approach provides a flat situation for converting the translating the Kripke model to SMV (Symbolic Model Verifier) code as follow:

- If (h1, h2) and (h2, h1) are two transitions, then these transitions are replaced by the transition (h1, h1).
- For all I, if (hi, h2) is a transition, then it is removed and replaced by the transition (hi, h1) if such a transition does not exist.

- For all j , if $(h2, h_j)$ is a transition, then it is removed and replaced by the transition $(h1, h_j)$ if such a transition does not exist.

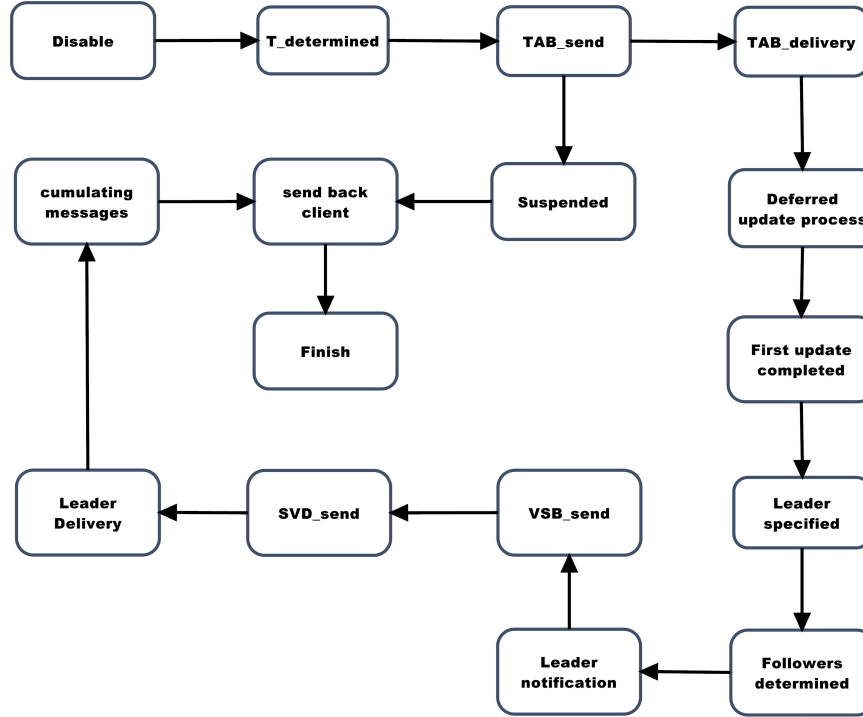


Figure 8. Kripke structure of propagation behavior.

The outcome of reducing the Kripke model is displayed in fig 10. Then the reduced model is converted to SMV code. The SMV code demonstrates the transition relation of the Kripke model.

In this section, we illustrated the model checking approach to verifying the combined semi-active data replication approach behaviors through the following three phases:

- Defining some examples of CTL and LTL properties which can be verified for the logical behavior.
- Converting the statechart of the propagation behavior to a Kripke model.
- Reducing the achieved Kripke model.

In the next section, to implement the proposed model the following steps are done:

- Translating the reduced Kripke model to SMV code in NuSMV graphical user interface model checker.
- Verification of the extracted properties from the logical behavior by using NuSMV interactive model checker.

5. VERIFICATION APPROACH

In this section, the verification of the proposed model based on state-of-art technologies such as symbolic model checking is provided. We used NuSMV-GUI which is an expanded the open source NuSMV tool for modeling the propagation and logical behaviors. We present two methods for verification of combined semi-active data replication model.

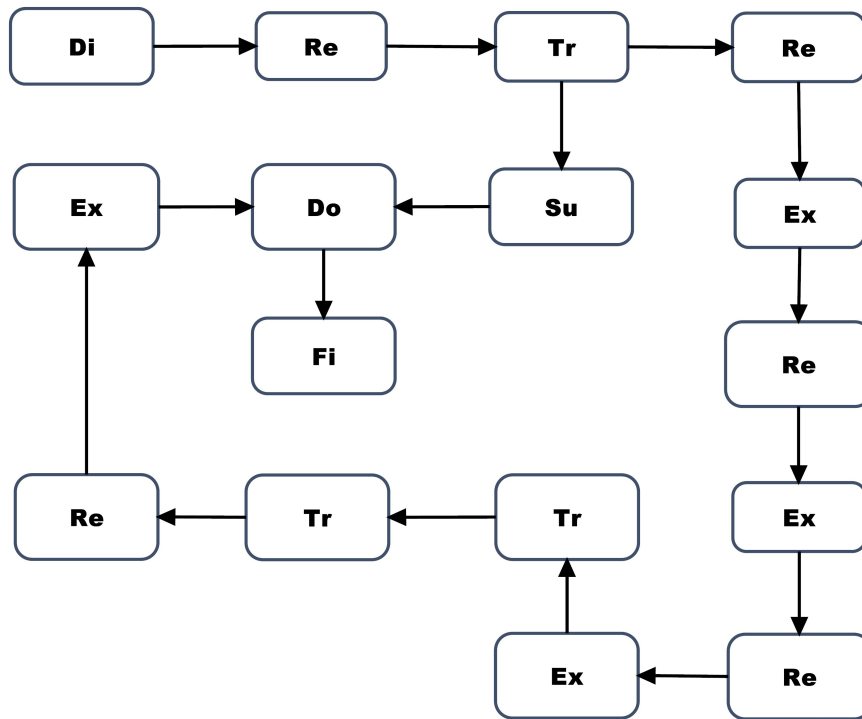


Figure 9. Kripke model of CSADR approach.

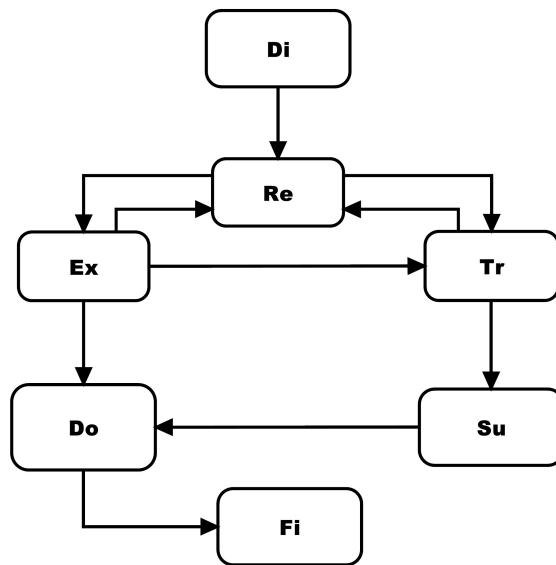


Figure 10. Reduced Kripke model of CSADR approach.

First verification method is generating SMV code by using NuSMV-GUI directly. Figure 11 illustrates the combined semi-active data replication model in the NuSMV-GUI tool. After specifying some LTL and CTL properties in specification section (as shown in figure 11), the smv code is generated automatically for propagation behavior by clicking on the generate SMV code icon. A java code has write for generating the SMV code in NuSMV-GUI tool. We enable this

code using eclipse tool and embed this code in file master.nusmvr for running NuSMV-GUI. When we click on generate SMV code, the file run automatically and generate SMV code for running in NuSMV-Interactive shell.

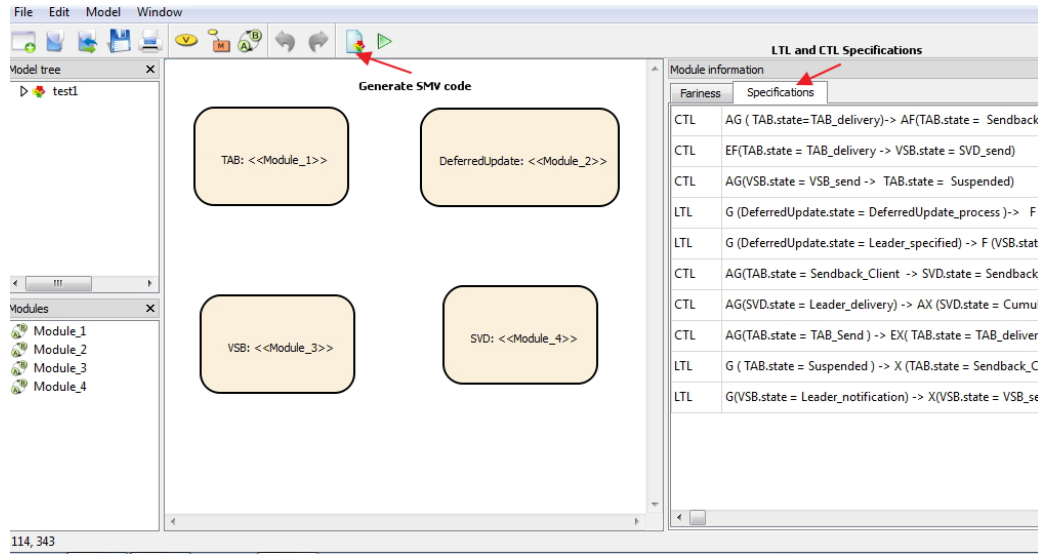


Figure 11. Generating SMVcode by using NuSMV-GUI tool.

After generating the SMV code, we verify generated code by using NuSMV-Interactive shell. Figure 12 depicts the LTL properties checked in NuSMV model Checker by green line. The L1, L2, L3 and L4 properties are satisfied properties.

```

NuSMV Interactive
WARNING *** This version of NuSMV is linked to the zchaff SAT solver ***
WARNING *** (see http://www.princeton.edu/~chaff/zchaff.html). ***
WARNING *** Zchaff is used in Bounded Model Checking when the ***
WARNING *** system variable "sat_solver" is set to "zchaff". ***
WARNING *** Notice that zchaff is for non-commercial purposes only. ***
WARNING *** NO COMMERCIAL USE OF ZCHAFF IS ALLOWED WITHOUT WRITTEN ***
WARNING *** PERMISSION FROM PRINCETON UNIVERSITY. ***
WARNING *** Please contact Sharad Malik (malik@ee.princeton.edu) ***
WARNING *** for details. ***

NuSMV > read_model -i test1.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > check_ltlspec
-- specification < G DeferredUpdate.state = DeferredUpdate_process -> F Deferre
dUpdate.state = Followers_determined) is true
-- specification < G DeferredUpdate.state = Leader_specified -> F USB.state = U
SB_send) is true
-- specification < G TAB.state = Suspended -> X TAB.state = Sendback_Client) i
s true
-- specification < G USB.state = Leader_notification -> X USB.state = USB_send)
is true
NuSMV > =

```

Figure 12. The Checking LTL properties of generated GUI model in NuSMV Interactive mode.

Also Figure 13 displays the results of true CTL properties by green line and false CTL property by red line. The generated counter-example of false property shows when VSB operation is executed, then the state Suspended in TAB propagation cannot execute. So this property is

false.

The second verification method is the translating the reduced CSADR Kripke model to the SMV syntax according last section methods. Fig. 14 shows the translated SMV code from the reduced CSADR Kripke model (shown in figure 10) by Roudabeh tool which is a tool developed by using Java language for writing, saving and editing Promella and SMV models [25]. We specify properties of the logical behavior in LTL and CTL formulas (defined in 5.2) that can be checked in the NuSMV model checker. After adding some properties, we can confirm the SMV file by saving with .smv format in Roudabeh tool.

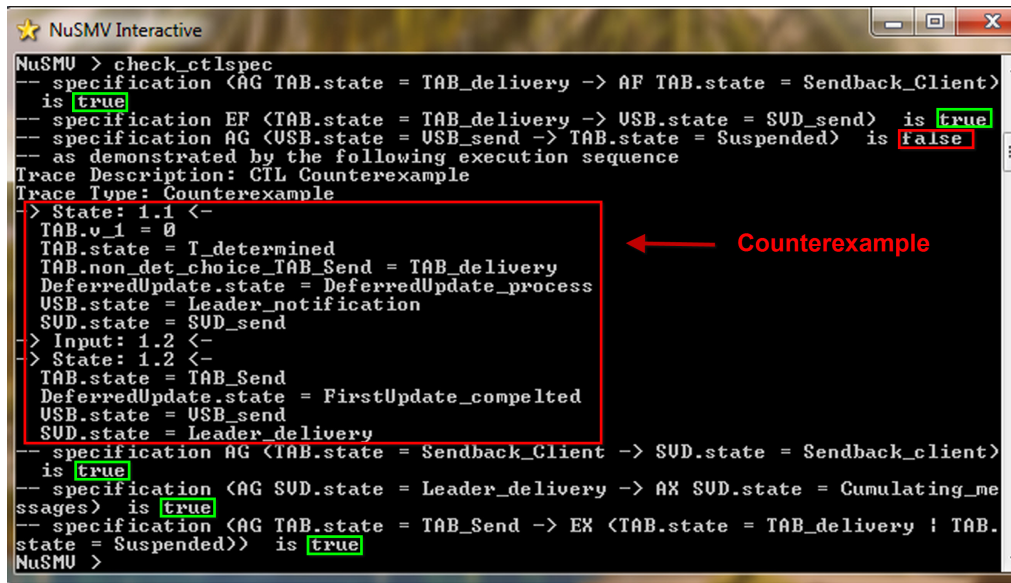


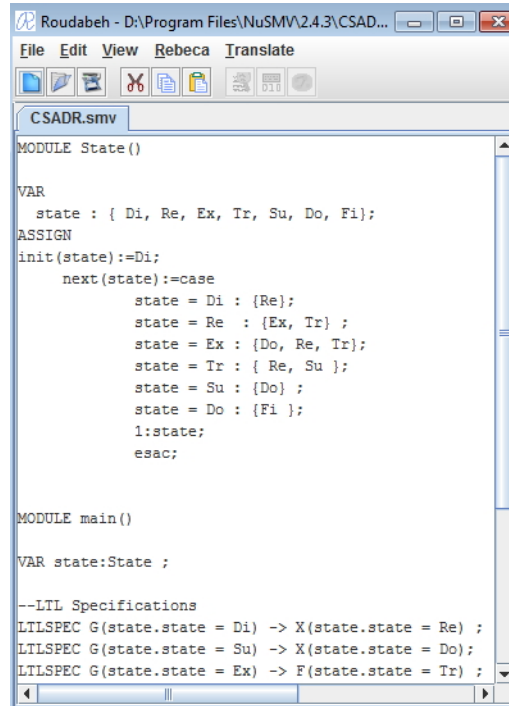
Figure 13. The Checking CTL properties of generated GUI model in NuSMV Interactive mode.

After generating SMV code by using Roudabeh tool , we verify the SMV code by using NuSMV-Interactive shell model checker. To check the properties which are illustrated in subsection 4.2, the following commands are used in NuSMV model checker.

- NuSMV: read_model i CSADR.smv
- NuSMV: flatten_hierarchy
- NuSMV: encode_variables
- NuSMV: build_model
- NuSMV: check_ltlspec (to check LTL specifications)
- NuSMV: check_ctlspec (to check CTL specifications)

Figure 15 depicts the LTL properties checked in NuSMV model Checker by green line. The L1, L2, L3 and L4 properties are satisfied properties. Also this figure displays the results of CTL properties by yellow dash-line. In the implementation, our system detected successfully the logical problems of all properties described in section 4.2.

Also in Figure 16, we evaluate the states and transitions of CSADR approach in reachability and fairness conditions. By using *compute_reachable*, *Print_fair_states* and *Print_fair_transitions* commands, we can check state reachability, state fairness and transition fairness. By specifying a yellow underline in below of each number, the result shows that there is the reachability and fairness for all states and all transitions of CSADR approach. By using *check_fsm* command, the deadlock problem can be checked by red border line in finite state machine of combined semi-active data replication approach. To evaluate the performance of the proposed approach, next



```

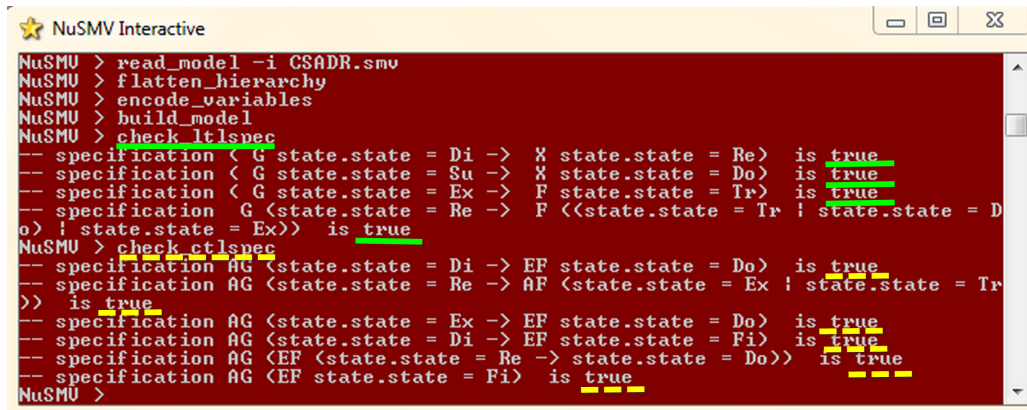
Roudabeh - D:\Program Files\NuSMV.2.4.3\CSAD...
File Edit View Rebeca Translate
CSADR.smv
MODULE State()
VAR
  state : { Di, Re, Ex, Tr, Su, Do, Fi};
ASSIGN
init(state):=Di;
  next(state):=case
    state = Di : {Re};
    state = Re : {Ex, Tr};
    state = Ex : {Do, Re, Tr};
    state = Tr : {Re, Su};
    state = Su : {Do};
    state = Do : {Fi};
  1:state;
  esac;

MODULE main()
VAR state:State;

--LTL Specifications
LTLSPEC G(state.state = Di) -> X(state.state = Re);
LTLSPEC G(state.state = Su) -> X(state.state = Do);
LTLSPEC G(state.state = Ex) -> F(state.state = Tr);

```

Figure 14. The conversion of Kripke model to SMV code by Roudabeh tool.



```

NuSMV Interactive
NuSMU > read_model -i CSADR.smv
NuSMU > flatten_hierarchy
NuSMU > encode_variables
NuSMU > build_model
NuSMU > check_ltlspec
-- specification < G state.state = Di -> X state.state = Re > is true
-- specification < G state.state = Su -> X state.state = Do > is true
-- specification < G state.state = Ex -> F state.state = Tr > is true
-- specification G <state.state = Re -> F <(state.state = Tr ! state.state = D
o) ! state.state = Ex >> is true
NuSMU > check_ctlspec
-- specification AG <state.state = Di -> EF state.state = Do > is true
-- specification AG <state.state = Re -> AF <state.state = Ex ! state.state = Tr
>> is true
-- specification AG <state.state = Ex -> EF state.state = Do > is true
-- specification AG <state.state = Di -> EF state.state = Fi > is true
-- specification AG <EF <state.state = Re -> state.state = Do >> is true
-- specification AG <EF state.state = Fi > is true
NuSMU >

```

Figure 15. The Checking LTL and CTL properties in NuSMV Interactive mode.

section provides some analytical results. These results specify that our proposed combined semi-active data replication approach is reachable, fair and deadlock free in the temporal language.

6. EVALUATION

The proposed combined semi-active data replication approach has three advantageous in comparison of similar works. First, unlike many papers in this scope, the proposed approach used to time atomic broadcast and same view delivery protocols for navigating the updates propagation in distributed databases. Second advantageous is using leader specification instead specifying a leader in first of protocol process. Third advantageous is decreasing time and memory consumption of system verification. The proposed approach is evaluated according to the faithfulness of the formal models and their usefulness for model checking. Due to the size and complexity of

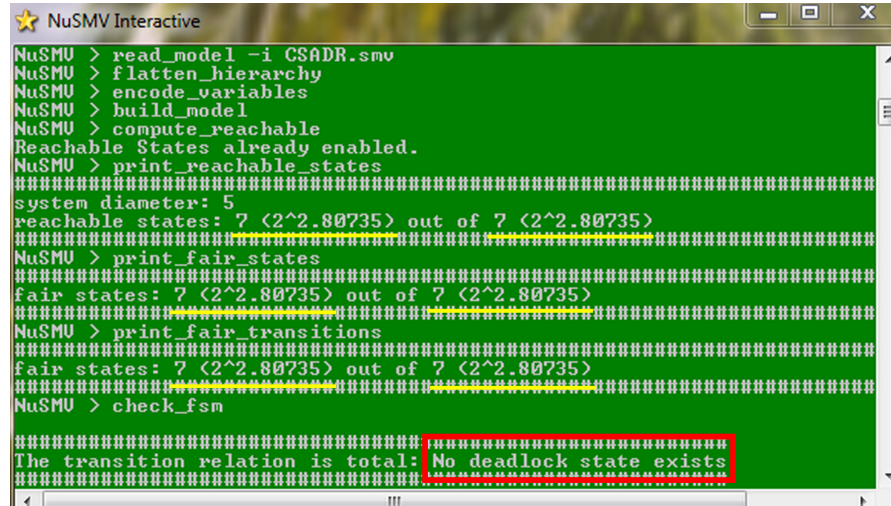


Figure 16. Checking reachability and fairness of CSADR approach.

current systems reachability and fairness are not always possible, therefore two requirements is necessary to verifying the model:

- The presented model contains all the necessary hints for checking a required property.
- The model covers only the effective behaviors of the actual system.

The correct traces of combined semi-active data replication approach checked by appropriated paths in the LTL and CTL formulas using two implementation methods:

- (1) Generating SMV code using NuSMV (GUI) tool and
- (2) Generating SMV code using the reduced Kripke structure mechanism.

Figure 17 shows the comparison of verification time for the combined semi-active data replication approach and the semi-active model by using GUI and Reduced Kripke implementation methods. We see that the verification time for the combined semi-active data replication approach by using GUI and reduced Kripke methods are lower than the semi-active model as the number of sifted states increases.

Also Figure 18 illustrates the comparison of memory consumption for verifying the combined semi-active data replication approach and the semi-active model by using GUI and Reduced Kripke implementation methods. We note that the memory consumption for the combined semi-active data replication approach using GUI and reduced Kripke methods are lower than the semi-active model as the number of sifted states increases.

Table 1 shows the evaluation results to check the model of combined semi-active data replication approach using reduced Kripke and GUI implementation methods which are achieved by NuSMV model checker tool.

Table I. Verification statistics for CSADR approach

CSADR implemented by Reduced Kripke	CSADR implemented by GUI
Memory in use (byte): 4635798	Memory in use (byte): 4959492
Number of BDD variables: 7	Number of BDD variables: 15
Number of sifted variables: 1000	Number of sifted variables: 3200
Number of swapped variables: 1500000	Number of swapped variables: 3200000
total number of nodes: 1012	total number of nodes: 2465

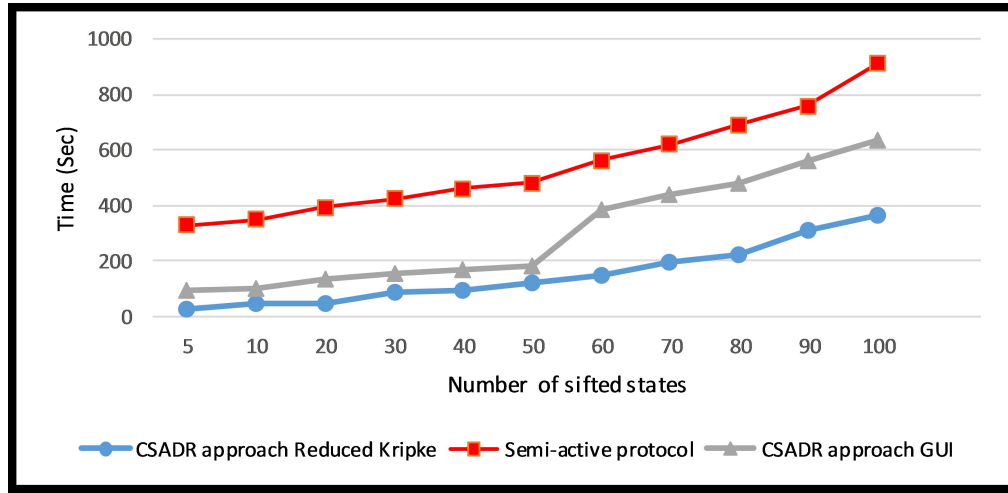


Figure 17. The results of Verification time.

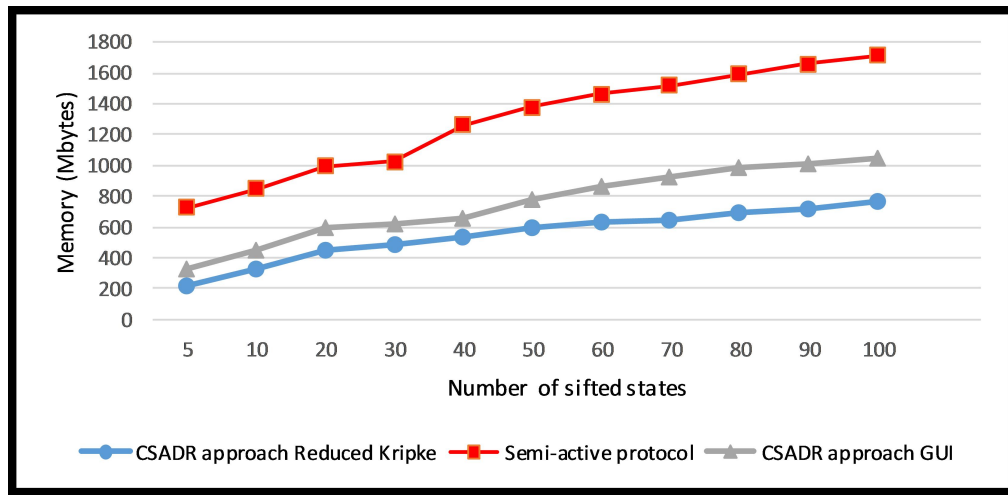


Figure 18. The results of memory consumption.

As exposed in Table 1 for reduced Kripke method, the verification of 2^7 model are computed for all state space, the number of BDD variables is 7, the number of sifted variables are 1000, the number of swapped variables are 1500000 and the total number of nodes are 1012. Also for GUI method, the verification of 2^{15} model are computed for all state-space, the number of BDD variables is 15, the number of sifted variables are 3200, the number of swapped variables are 3200000 and the total number of nodes are 2465.

7. CONCLUSION AND FUTURE WORK

This paper presented a combined semi-active data replication approach in distributed systems. The proposed approach is based on semi-active replication protocol. Unlike other approaches, the combined semi-active data replication approach is separated into propagation and logical behaviors which enabled the mapping process between two behaviors by means of the symbolic model checking approach based on binary decision diagram. We take out the probable specification of combined semi-active data replication approach from logical behavior in the form of

LTL and CTL temporal logic formulas. Also we verified the behavior models of the combined semi-active data replication approach by NuSMV-GUI tool and the NuSMV-Interactive shell. The verification results displayed that the combined semi-active data replication approach can propagate the update request of each client to other replicas successfully. Finally, a comparison of the execution time and the memory consumption for the combined semi-active data replication approach and semi-active protocol is analyzed in two verification methods. The comparison results show that the verification time and memory consumption for the combined semi-active data replication approach by reduced Kripke method is lower than the semi-active model as the number of sifted states increases. In the future work, we will research to extend this approach for fault tolerant conditions as well as its behavioral modeling and formal verification.

REFERENCES

- ABAWAJY, J. AND MAT DERIS, M. 2013. Data Replication Approach with Data Consistency Guarantee for Data Grid. *Computers, IEEE Transactions on PP*, 99, 1.
- ALAMI MILANI, B. AND JAFARI NAVIMIPOUR, N. 2016. A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. *Journal of Network and Computer Applications* 64, 229–238.
- ARMENDÁRIZ-IÑIGO, J. E., DE MENDIVIL, J. R., GARITAGOITIA, J. R., AND MUÑOZ-ESCOPI, F. D. 2009. Correctness proof of a database replication protocol under the perspective of the I/O automaton model. *Acta Informatica* 46, 4, 297–330.
- CLARKE JR., E. M., GRUMBERG, O., AND PELED, D. A. 1999. *Model Checking*. MIT Press, Cambridge, MA, USA.
- DWYER, M. B., AVRUNIN, G. S., AND CORBETT, J. C. 1998. Property Specification Patterns for Finite-state Verification. In *Proceedings of the Second Workshop on Formal Methods in Software Practice*. FMSP '98. ACM, New York, NY, USA, 7–15.
- GARCIA, R., RODRIGUES, R., AND PREGUIÇA, N. 2011. Efficient Middleware for Byzantine Fault Tolerant Database Replication. In *Proceedings of the Sixth Conference on Computer Systems*. EuroSys '11. ACM, New York, NY, USA, 107–122.
- GARCIA-GARCIA, J., ORDONEZ, C., AND TOSIC, P. T. 2012. Efficiently repairing and measuring replica consistency in distributed databases. *Distributed and Parallel Databases* 31, 3, 377–411.
- GRAY, J. AND REUTER, A. 1992. *Transaction Processing: Concepts and Techniques*, 1st ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- HAMMAL, Y., BEN-OTHTMAN, J., MOKDAD, L., AND ABDELLI, A. 2015. Formal Modeling of Greedy Nodes in 802.15.4 WSN. *ICT Express* 1, 1 (jun), 10–13.
- HANSEN, H., VIRTANEN, H., AND VALMARI, A. 2003. Merging State-Based and Action-Based Verification. In *Proceedings of the Third International Conference on Application of Concurrency to System Design*. ACSD '03. IEEE Computer Society, Washington, DC, USA, 150—.
- HOLZMANN, G. J. 1997. The model checker SPIN. *IEEE Transactions on Software Engineering* 23, 5, 279–295.
- JAFARI NAVIMIPOUR, N., HABIBIZAD NAVIN, A., RAHMANI, A. M., AND HOSSEINZADEH, M. 2015. Behavioral modeling and automated verification of a Cloud-based framework to share the knowledge and skills of human resources. *Computers in Industry* 68, 65–77.
- KEMME, B. AND ALONSO, G. 2000. A New Approach to Developing and Implementing Eager Database Replication Protocols. *ACM Trans. Database Syst.* 25, 3 (sep), 333–379.
- LIU, Y., SUN, J., AND DONG, J. S. 2010. *Automated Technology for Verification and Analysis: 8th International Symposium, ATVA 2010, Singapore, September 21-24, 2010. Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter Developing, 371–377.
- MANNA, Z. AND PNUELI, A. 1995. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag New York, Inc., New York, NY, USA.
- McMILLAN, K. L. 1992. Symbolic Model Checking: An Approach to the State Explosion Problem. Ph.D. thesis, Pittsburgh, PA, USA.
- PÉREZ, J. M., GARCÍA-CARBALLEIRA, F., CARRETERO, J., CALDERÓN, A., AND FERNÁNDEZ, J. 2010. Branch replication scheme: A new model for data replication in large scale data grids. *Future Generation Computer Systems* 26, 1 (jan), 12–20.
- POLEDNA, S. 1994. Replica determinism in distributed real-time systems: A brief survey.
- POWELL, D. 1994. Distributed Fault Tolerance - Lessons Learned from Delta-4. In *Revised Papers from a Workshop on Hardware and Software Architectures for Fault Tolerance*. Springer-Verlag, London, UK, UK, 199–217.

- SAFARKHANLOU, A., SOURI, A., NOROUZI, M., AND SARDROUD, S. E. H. 2015. Formalizing and Verification of an Antivirus Protection Service using Model Checking. In *Procedia Computer Science*. Vol. 57. Elsevier, 1324–1331.
- SCHNEIDER, F. B. 1990. Implementing Fault-tolerant Services Using the State Machine Approach: A Tutorial. *ACM Comput. Surv.* 22, 4 (dec), 299–319.
- SCIASCIA, D. AND PEDONE, F. 2013. Geo-replicated storage with scalable deferred update replication. In *Proceedings of the International Conference on Dependable Systems and Networks*.
- SOURI, A. AND JAFARI NAVIMPOUR, N. 2014. Behavioral modeling and formal verification of a resource discovery approach in Grid computing. *Expert Systems with Applications* 41, 8, 3831–3849.
- SOURI, A. AND NOROUZI, M. 2015. A new probable decision making approach for verification of probabilistic real-time systems.
- SOURI, A. AND PASHZADEH, S. 2014. CONSISTENCY OF Data Replication protocols in database Systems: A review. *International Journal on Information Theory (IJIT)* 3, 4, 19–32.
- SOURI, A. AND RAHMANI, A. M. 2014. A survey for replica placement techniques in data grid environment. *International Journal of Modern Education and Computer Science* 6, 5, 46.
- WIESMANN, M., PEDONE, F., SCHIPER, A., KEMME, B., AND ALONSO, G. 2000. Understanding replication in databases and distributed systems. *Proceedings 20th IEEE International Conference on Distributed Computing Systems*, 464–474.
- ZHU, Y. AND WANG, J. 2010. Client-centric consistency formalization and verification for system with large-scale distributed data storage. *Future Generation Computer Systems* 26, 8 (oct), 1180–1188.

Alireza Sourì received his B.S. degree in Software Engineering from University College of Nabi Akram, Iran, in 2011 and his M.Sc. degree in Software Engineering from Science and Research Branch, Islamic Azad University, Iran in 2013. Currently, he is a researcher and lecturer in Islamic Azad University. His research interests include Formal Specification & Verification, Model checking, Grid & Cloud computing. He is member of The Society of Digital Information and Wireless Communications.

