Using SPIN Model Checking for Node Order Protocol Verification

Chanjuan Li
1.School of Mathematics and Statistics
2.School of Information Science and Engineering
Lanzhou University .P.R.China
South Tianshui Road 222. Lanzhou
lichanjuan04@gmail.com

Nicholas McGuire
School of Information Science and Engineering
Lanzhou University .P.R.China
South Tianshui Road 222. Lanzhou
mcguire@lzu.edu.cn

Qingguo Zhou
School of Information Science and Engineering
Lanzhou University .P.R.China
South Tianshui Road 222. Lanzhou
zhouqg@lzu.edu.cn

Mingqiang Yang
School of Information Science and Engineering
Lanzhou University .P.R.China
South Tianshui Road 222. Lanzhou
mqyoung@gmail.com

Abstract

Formal verification play an important role in development and application of safety-critical system, which is highly recommended at SIL4 in IEC61508. SPIN as a powerful model checker verifies the correctness of distributed software models in a rigorous and mostly automated fashion, against a set of presumably correct specifications. The object to be verified is the protocol - NOP designed for safety-critical distributed embedded system, which provides a communication pattern through deciding the ordering of access to shared. medium by a pre-defined node order. NOP use an event-trigger paradigm instead of time-triggered, thus communication events including error detection is based on discrete events instead of real time, which make it possible to use SPIN as validation tool.

The objective of this paper is to verify the specification of NOP showing that it meets the fault tolerant requirement under a single-fault assumption. According to the fault hypothesis, NOP has performance failure semantic, that is a faulty node deliver correct results in the value domain,but in the time domain, the results may be delivered early or late.

In this paper, we make use of the faulty hypothesis assumption to partition the protocol model into 7 equivalence classes according to the fault analysis result of a FMEA into dedicated models, which ensures that each model can be treated independently of the others, significantly reducing the size of the overall state space to be checked by the validation process and making the model more understandable. Finally, based on properties verified by each model shown at the last section of the paper, the safety and liveness properties of NOP can be validated by integrating the results of each model because the full coverage of faults and independence of models. Though we can not establish the correctness by formal verification, we provides a higher level of assurance of the consistency between the specification and requirement.

1 Introduction

Model checking is an automated technique that, given a model of the system and some property, checks whether the model satisfies the specified properties. Notably modeling can detect concurrency defects that are otherwise difficult to discover. Within appropriate constrains, a model checker can perform an exhaustive state-space search on a software design or implementation and alert the implementing organization to potential design deficiencies. Compared to the other (semi)automated formal techniques (for instance, deductive methods, like theorem provers) model checking is relatively easy to use. The specification of the model is very similar to programming and as such it does not require much additional expertise from the user. Most notably SPIN allows relatively streight forward model development based on existing code due to this strong similarity. The verification procedure is completely automated and often takes only several minutes. Another important advantage of the method is that, if the verification fails, the possible erroneous behavior of the system can be reproduced. This significantly facilitates the location and correction of the errors.

This paper describes a practical application of model checking for validating the requirements for a real-time communication protocol. The case study described here is of verification of the specification of a communication protocol - NOP, which was designed for the distributed safety-critical system. In the protocol, detection of all possible faults defined in fault hypothesis and maintaining consistent among correct nodes (minimal 3) is provided. However, it was not possible to determine whether the specification provide the desired level of fault tolerance. More important, testing of the eventual implementation would not necessarily provide this validation either, due to the difficulty of ensuring test case coverage for all possible fault occurrence scenarios. The approach described here uses a formal automata based model derived from the specification. We define a set of safety and liveness properties, which are required to be satisfied when the system encounters an errors.

We used the model checker SPIN[1] in this paper, which will identify traces in the model for which these properties were violated. The formal method are used to model critical chunks of an informal specification, to check that key properties hold. The aim is to find errors, rather than to prove correctness.Application of formal method is driven by the need of the project, and is used as a modeling tool to answer questions that arise during verification and validation. In the case of NOP we are using during the design phase to allow to get the code right from the very outset.

The paper is organized as follows. In section 2, we describe the fault hypothesis defined in the NOP requirement. With a single-fault assumption, the basic communication semantic is designed. Due to error detection at the protocol layer, an effective membership service can be provided in NOP. To verify the consistency between the requirement and the specification, we derive the properties to be verified. In the first attempt, modeling all fault types in one model proved to be impractical due to state space explosion and high complexity. Thus we de-component the full featured model into several small submodels by the independence of different fault scenarios in Sectin 3. In Section 4, by using semi-formal method - FMEA, all possible hazards in the NOP are listed. Based on the result, we determine the maping between failure modes and submodels. The detailed modeling is described in Sectin 5, in which the verification result is displayed including performance data of verification as well as satisfied properties. Section 6 presents conclusions and describes our future work.

2 NOP High Level Model Description

Event-triggered Node Ordering Protocol [2] is designed for small-scale distributed safety-critical real-time system. The system model of NOP we assumed consists of nodes and a bus channel. In the model, no duplicated channel and bus guardian are involved, which make the fault semantic of NOP increase to performance failure from fail-silent failure [3], that is in NOP, a faulty node can send arbitrary message at an arbitrary point in time. And the fault PROPAGATES to the whole system instead of contain within a node. In addition, network omission failure must be considered at protocol layer. The requirement of NOP assumes a single-fault hypothesis that is at most one fault is present in the system at any time. Moreover, a restriction is put on fault arrival rate in NOP that a new fault will not occurs before the last error is diagnosed. Starting at a TMR(Triple Modular Redundancy) on the top of NOP, we require the minimal configuration of NOP to be 3 nodes, and the error must be detected within a FTU slot [4].

According to the requirement above, the protocol switches among three modes. First, all nodes are configured at the initialization with common parameters. Then the protocol begins operation and keeps going even in the present of a single fault. But when the fault scenario violates the defined fault hypothesis, each node will detect and quit communications, then the protocol is in a fail-safe state. Obviously, there is a loose couple between initialization and operation mode, thus we separate the specification into two parts and verify each part individually. In this paper, we only cover the verification on operation mode.

The basic communication semantic is similar to TTP. In NOP, medium assess is controlled by predefined Node Ordering Delivery List(NODL) which is the correspondent of MEDL in TTP. During the transmission slot, the sender is permitted to send at most one message, then the next transmission slot is assigned to the next node in NODL. As a receiver, the node first check if the incoming message follows the definition of NODL. If it is in-order, then the node accepts the frame and update the expected node in the next transmission slot. If not, the frame is discarded, the node keeps waiting for the correct message until it reaches a timeout. According to the common knowledge among nodes, we can safely deduce that if one receiver accepts the transmitted message, then all correct receivers will accept it. Therefore, the implicit acknowledgement integrated into the next transmitting message is used to eliminate handshakes.

In NOP specification, error detection is implemented at the protocol level by two error case: timeout and out-of-order message reception. Thus each node monitors the channel continuously and has common knowledge of ordering of arriving frames, then The failed node can be detected at system-wide based on the NODL. Based on error detection, at protocol level, NOP provides membership service for the upper layer protocols, which maintains an active node list of the system continuously during the running time. When a node is detected to have failed, it should be removed from the membership vector as soon as possible. Thus the reliable membership service must satisfy the following two requirements:
Agreement: The membership lists of all non-faulty nodes are the same.
Validity: The membership lists contain all non-faulty nodes and at most one faulty node.
The Agreement requires all correct node to have the same active node list, while Validity requires the faulty node must be detected and remove from the communication activities. Because a fault must be manifested before it can be diagnosed, we can not remove of the faulty node immediately.

From above, the main safety concern of NOP is error detection coverage and reliability of membership service. To verify such requirement, model checking is a powerful tool to perform exhaustive exploration of all possible behaviors, and then locate potential design deficiencies.

In this paper, we verify whether the design of NOP fulfil the following requirement:

- fault coverage of NOP
- NOP can keep operation under a single-fault hypothesis
- all correct nodes have the same membership list
- the faulty node can be detected and removed from the membership lists of all correct nodes
- The minimal number of nodes in NOP is 3.

3 Modeling NOP

3.1 The First Attempt of Modeling

In the specification of NOP, the run-time state machine of one node is as shown in Fig. 1.

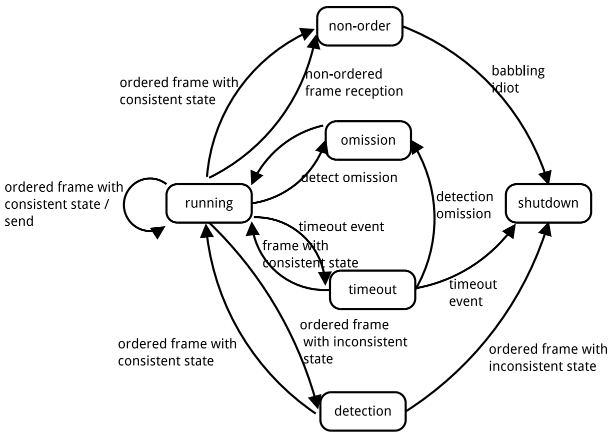
non-order
ordered frame with consistent state
non-ordered frame reception
babbling idiot
omission
ordered frame with consistent state / send
running
detect omission
detection omission
shutdown
timeout event
frame with consistent state
timeout
timeout event
ordered frame with consistent state
ordered frame with inconsistent state
ordered frame with inconsistent state
detection

FIGURE 1: *run-time state machine of NOP*

In the state machine, the input triggering each state transition is generated by a fault occurrence. So to implement the state machine, all kinds of faults must be injected into the model. It is hard to estimate the state space, but we can provide a number for comparison - that the fault-free model of NOP of 3 nodes has 481 states and 88-byte state-vector (in SPIN). It is obvious that the state space will explode if you implement the state machine in one model. Moreover, the model is hard to maintain as the specification evolves.

3.2 Reducing The State Space

There are a number of ways in which the state space can be reduced to a size amenable to model checking. The most reasonable way for our project is to partition the functional requirements of the system into equivalence classes by exploiting natural symmetries or subclasses that may be present in the domain. For NOP, we partitioned the functional behavior by separating the classed of fault that occur. The requirement assume a single-fault hypothesis and put restriction on fault occurrence rate, which implies independence of the different fault scenario. That is not to say that faults can't propagate but the propagation is assumed to not induced concurrent faults. Thus, we can split the state machine into several sub-state machine with a single fault injection each.

The state space of the model could be exhaustively searched allowing critical functional requirements to be validated down to the design level by judiciously abstracting away extraneous complexity. Such mechanism was been used to verify a fault tolerant spacecraft controller[5]. Because the method allow validation of partial specification, it is an practical approach for maintaining consistency between a co-evolving specification and an implementation.

4 Fault Analysis

In this section, the submodels of NOP is described in detail. According to the system model see Fig. 2,

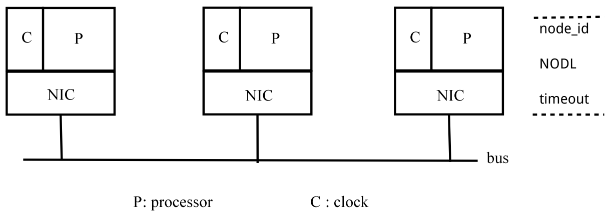

FIGURE 2: *the system model of NOP*

we do fault analysis by semi-formal method FMEA, which is helpful to increase the probability of fault coverage. In this paper, I only list the failure modes of FMEA, while omitting other parts, see Table 1.

failure modes	scope
processor:	
crash	detection in protocol
transient sending nothing	detection in protocol
sending message too late	detection in protocol
sending message at incorrect time	detection in protocol
sending invalid output	detection in protocol
babbling idiot failure	detection in protocol
reception omission	detection in protocol
network link:	
failed physical link	detetion in protocol
duplicated message	detection in protocol
corrupt message	handled by underlying network
loss of messages	detection in protocol
physic clock:	
dead clock	mitigated
too fast clock	detection in protocol
too slow clock	mitigated
clock draft effect	mitigated
node_id:	
invalid node_id	mitigated in initialization
duplicated node_id	mitigated in initialization
masquerade node_id	mitigated in initialization
NODL:	
truncated NODL	mitigated in initialization
corrupt NODL	mitigated in initialization
inconsistent NODL	mitigated in initialization

In the above failure mode list, the failure modes in value domain such as invalid message or corrupt message can be detected by a simple checker or CRC implementing at underlying physical layer, which have no effect on protocol operation, thus can ignore such failure modes at the protocol level. The failure modes of critical parameters of NOP are handle at initialization stage instead of in running time, but even they occur at running time, the symptom is that same as a faulty processor. From the protocol perspective, there is no difference between physical link failure and omission fault, and duplicated messages is seen as non-ordered frame sending by a faulty processor. Thus the only failure of network link the protocol must be handled is the transient omission failure. As far as failure modes of a faulty processor, babbling idiot failure manifests as a faulty sender sending message at incorrect time, thus it is detected by receipt of non-ordered message from one node.

After analysis of possible fault in NOP, we figure out the mapping between failure modes to the submodels as follows. The failure modes are independent of each other, and has a corresponding SPIN model. In the following section, we number the submodels in the follwoing order.

1. fail-silent processor
2. transient sending omission
3. message sending too late
4. transient receiving omission
5. transient channel omission
6. too fast clock
7. arbitrary processor failure with inconsistent state

5 Submodels

5.1 Properties To Be Verified

In Section 2, we list the requirement of NOP. Then we must transform these requirement to the statement SPIN can verify automatically. In above section, we take care of fault coverage issue. For Requirement 2, we use the following claim to verify:
P1(Liveness): there is no non-progress cycle in any correct node.
it means there is any correct node going on normal communication under specified fault scenario. Strictly speaking, Agreement requirement states that all correct nodes keep consistent at any time. But this is too strict to implement, due to divergence of processing speed and jitter. The practical requirement is that the inconsistent state is allowed but with the upper boundary. In NOP, the agreement requirement is described as follows:
P2(Safety): at any state, the divergence of the membership vectors of correct nodes is at most one bit.
As far as requirement that a faulty node is removed by all correct nodes, we express it in temporal logic as follows:
P3:

$$[](S \rightarrow \Diamond Q) \tag{1}$$

where S = node[i].mb[FAULT_ID] == 0
and Q = node[i].mb = node[j].mb (i,j $\in$ correct nodes)
the temporal logic means that when any correct node responses to the detected error by removing the faulty node ,then eventually all correct nodes will do the same thing and resynchronize.

5.2 Modeling

In the following section, we introduce abstracting the protocol and injecting fault during modeling. Obviously, each node must be a individual process. About bus, it is only responsible for transmitting the message and do nothing with the requirement to be verified, thus it is not necessary to put it in a process(exception in transient channel omission model). The message format only includes the control states of NOP with 4 field. And there is one message channel for each node to sending and receiving messages. According to the communication semantic of NOP, the message channel need at most 1 slot. SPIN supports two types of communication channel. One is synchronous or rendezvous channel, the other is asynchronous or buffered channel. Using synchronous model, the system seems there is a precise global time that events will be issued at exact the same time at individual process. While the asynchronous model can execute at any possible order. In NOP, there is no global time at protocol layer at all, and the communication activities are issued by themselves. Thus jitter must be considered as a potential factor impacting the behavior of the protocol. Therefore, asynchronous communication model is used to modeling NOP. In addition, we assume that the system is synchronous implicitly that is before the new message arrival, the node finishes processing the last message. But in SPIN model, due to asynchroniztion execution and no assumption on execution at all, such implicit assumption must be explicit by synchronization among processes. We implement such assumption with a global variable.

Due to non-deterministic feature of SPIN, the fault-injection is easy to implement, see the following example injecting a failed processor failure:

```
if
:: myturn -> { ch!msg; change_state;}
:: error = 0 -> { inject fault; error = 1;}
fi;
```

In above pseudo code, SPIN will select one of the two block to execute randomly when the two conditions hold at the same time, then the fault will be injected into the model randomly. For simplicity, we assign a node as a fault node which does not impact the consistency of the model only for reducing state spaces. The other issue about injecting fault is the constrains on a single-fault and fault rate which can be implemented by a global variable.

The core of the model is processing message including detailed error diagnosis mechanism. Here, with the limitation of length, I only take fail-silent processor model as an instance. The state machines of a faulty node and a correct node in the fail-silent processor failure are shown in Fig. 3 and Fig. 4,which are simplified from the state table generated by SPIN verifier by removing transition due to synchronization and assignment statement.

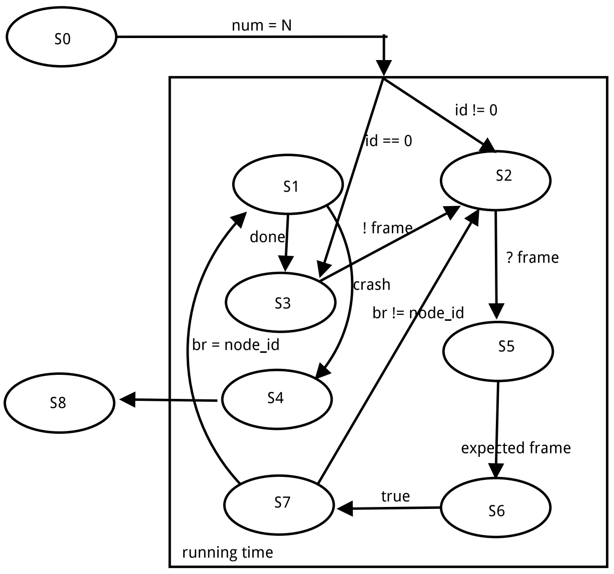

FIGURE 3: *The state transition diagram of a fail-silent processor*

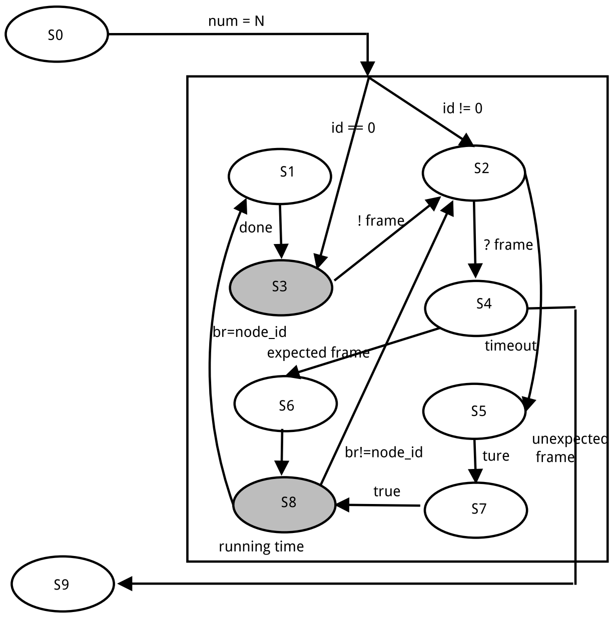

FIGURE 4: *The state transition diagram of a correct processor*

In Fig. 3, each node initializes some data structure at S0 and enters run-time state when all other nodes finishing initialization. The first sender will begin transmitting message (S3) and other nodes listen to the network (S2). When the sender finishes the transmission, it transits to S2 waiting for expected frame. As a receiver, the node will transit from S2 to S7 presenting the process from reception of a message to processing of the message in the protocol. The node will enter S1 to wait for that all nodes process the frame, then a new transition slot starts. For the fault node, once the fault is triggered, the node can not sending any message but listening to the network (S8). While in the correct node model, when a fault is triggered, SPIN will enter a timeout state in which there is no executable statement and the system is stalled. At this point, the correct node detects the error and makes diagnosis by removing the faulty node from its membership vector, shown by S5 and S7. Then the correct node can go on operation entering S1 or S2.

For the safety property, we assert that the divergent distance is less than 1 in the state S8 of the correct node and that the agreement is reached at the state S3. As far as the liveness property is concern, we verify that by setting the S3 as a progress state to show there is no non-progress cycle in a correct node model.

5.3 Result Analysis

Whether the requirement is met by the protocol specification is up to the verification result of each model. Only when all the model satisfy the properties, we can show the consistency between the requirement and the specification. The first verification is done with the minimal configuration of 3 nodes. The result is shown in Table 2. As shown in the table, all the models pass the verification of P1 and P2(the No. 0 model is the fault-free model.). The property with star mark means that during verification, the weak fairness must be enabled. Because we introduce omission recovery mechanism, not all models need to verify P3, we mark the models which can recovery from the error. Then, We repeat the same verification with more than three nodes, and there is no model violate the properties either.

submodels	P1	P2	P3
0	$\surd$	$\surd$	-
1	$\surd^*$	$\surd$	$\surd$
2	$\surd^*$	$\surd$	$\surd$
3	$\surd^*$	$\surd$	$\surd$
4	$\surd^*$	$\surd$	-
5	$\surd^*$	$\surd$	$\surd$
6	$\surd^*$	$\surd$	-
7	$\surd^*$	$\surd$	-

In addition, the state space and resource consumption of models are interested. All related data is shown in Table 3 which is the verification of 3 nodes with the safety property. The state space of models are in the order of thousand, and memory usage and execution time have no big difference among models.

submodels	state space	mem usage	time(s)
0	481	4.653 MB	0
1	2266	4.849MB	0.03
2	3060	4.946MB	0.02
3	6703	5.141MB	0.03
4	2125	4.849MB	0.01
5	4208	5.044MB	0.04
6	1843	4.751MB	0.01
7	4438	5.044MB	0.02

For more than 3 nodes, the performance results of fault-free model and failed processor model are shown in Table 4 and Table 5. As shown in Table 4 and Table 5, the state space increases exponentially with the number of nodes. With 6 nodes, the state space of failed processor model reaches the order of million. At the same time, memory usage and execution time are increased dramatically. The objective of SPIN model is not for scalability issue, thus it is sufficient of six nodes for the verification, though there left a space for further reducing space states of models.

nodes	state space	mem usage	time(s)
3	481	4.653MB	0
4	3128	4.946MB	0.03
5	19695	7.192	0.16
6	120302	23.013	1.33

nodes	state space	mem usage	time
3	2266	4.849MB	0.03
4	21820	6.997MB	0.11
5	208128	30.044	1.45
6	1945400	287.075MB	20.6

6 Conclusion

In this paper,we verify the design of NOP to show that it meets the fault tolerant requirement under a single-fault assumption that is a faulty node deliver correct results in the value domain,but in time domain, the results may be delivered early or late. The first attempt to model the protocol into one model is impractical due to huge state space which consumes much more computation time and resources. Due to the single-fault assumption and the constrains on fault rate, we find a way to reduce state space by partitioning the model by separating out the classes of fault that can occur into dedicated models. Then we derive the fault classes the result of FMEA to get the full fault coverage and make each model can be treated independently of the others. Finally, based on properties verified by each model, the safety and liveness properties of NOP is validated by integrating the results of each model.

It is important to note that with this approach,the focus is not on proving correctness, but on revealing errors. We have shown in the case study that the approach is capable of finding subtle errors that are otherwise almost impossible to detect. If we did not find any error, that would not establish correctness, but is does provide a higher level of assurance than is otherwise possible.

Because we models the specification partially by a set of independent submodels, the models are easy to maintain with a co-evolving specification. Moreover, We think this approach make it possible to be used to verify the consistency between the specification and an implementation. In the future work, we will try to evolve the models to the implementation and provide a higher level of confidence for the consistency of the implementation.

References

[1] *The SPIN Model Checker:Primer and Reference Manual*,Gerard J. Holzmann, 2003, ADDISON-WESLEY

[2] *New Real-time Network protocol-Node Order Protocol*, Chanjuan Li, Nicholas McGurie, Qingguo Zhou, 2009, rtlws11.

[3] *Fault-Tolerant Real-Time Systems: the Problem of Replica Determinism,Stefan Poledna, Kluwer Academic Publishers,1st edition,1996*

[4] *TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems*, H.Kopetz, G. Grunsteidl, 1993

[5] *Validating Requirements for Fault Tolerant Systems using Model Checking*, Francis Schneider, Steve M, Easterrook, et.al.