PROBABILISTIC MODEL CHECKING OF AN AUTOMATIC IDENTIFICATION SYSTEM

Takashi TOYOSHIMA
Graduate School of Science&Technology
Kwansei Gakuin University
2-1, Gakuen, Sanda, Hyogo, 669-1337, JAPAN
email: bwn21358@kwansei.ac.jp

Kazuko TAKAHASHI
School of Science&Technology
Kwansei Gakuin University
2-1, Gakuen, Sanda, Hyogo, 669-1337, JAPAN
email: ktaka@kwansei.ac.jp

ABSTRACT
This paper shows the analysis and verification of an Automatic Identification System (AIS) using a probabilistic model checker. AIS is a communication system between ships for safe sailing. Its communications are not interactive, and hence ships may try to send messages at the same time and be unaware that the messages have not been received. If such communication failures occur frequently, dangerous conditions will result. In actual practice, each ship is assigned a turn for message transmission, and this is determined probabilistically. However, neither the limit of this probability nor the allowable rate of communication failure is known. We construct several models with different strategies for this system, investigate the probability that safety is ensured, and verify it.

KEY WORDS
AIS, model checking, PRISM, probability, formal method

1 Introduction

In 2002, the Marine Safety Association of the International Marine Organization declared that every ship should be fitted with the Automatic Identification System (AIS) with data communication to ensure safety at sea. AIS is a ship-to-ship communication system for ensuring nautical safety [1]. It allows ships to communicate with other ships and land-based facilities, periodically and automatically sending and receiving information such as ship identification, chart position, velocity, and so on. Use of AIS enables the early detection of ships at night or in the shade of an island, which is difficult to accomplish with radar alone. However, the introduction of AIS is such a recent occurrence that its effectiveness and possible drawbacks have not been adequately analysed.

Because AIS communications are not interactive, simultaneous transmission of messages may occur. In such a case, both messages are lost, and the senders are unaware of this. To avoid this there is a reservation system, but there is no verification of this system. In fact, ”double-booking” does happen, but it has not been given serious consideration. One reason for this is that it occurs infrequently in practice, because the range of available times to send a message is large compared to the number of ships. Another reason is that even if double-booking does occur, the problem can be eliminated over the long run by changing the ships' transmission times. Thus, the problem seems to be solved by introducing randomness rather than using a complicated protocol. However, the probability and allowable rate of double-booking are unknown, and strict formal analysis has not been done so far.

In this paper, we introduce a formal method of using a probabilistic model checking tool to verify this system. Formal methods are mathematically- or logically-based techniques for the specification, development and verification of software and hardware systems with its designer's intentions. They have been actively investigated as tools for system verification, and good results have been obtained. Model checking is one such formal method [2, 5]. It utilises a state transition graph to describe the behaviour of a system and verifies whether the given specifications are satisfied by checking all possibilities. A number of tools for model checking have been developed and are in wide use including industrial applications [3, 4, 5]. These earlier model checkers can handle the nondeterministic behaviour of a system but not probabilistic behaviour. However, many practical applications include probability in both their behaviour and their specifications: for example, “power off occurs with probability 0.05” or “the reliability of packet sending should be more than 99 percent.” This has led to the development of model checkers that include probability [8, 9, 11, 15, 16]. PRISM is one such probabilistic model checker [11].

In this paper, we apply PRISM to the verification of AIS. We focus on its reservation system and construct several models. These models use different strategies for reservation. We implement each, perform a probabilistic analysis using PRISM, and compare the probabilities of the different strategies. We then investigate the probability that safety is ensured, and verify it. The result of experiment shows that we can propose a new protocol, not so complicated, which improves an actual one. The experiment is performed in small size of models, since this is a preliminary attempt to assure the effectiveness of these models.

The paper is organised as follows. In section 2 we describe AIS. In section 3 we describe probabilistic model checking. In section 4 we present the models for the AIS reservation system. In section 5 we analyse the system and

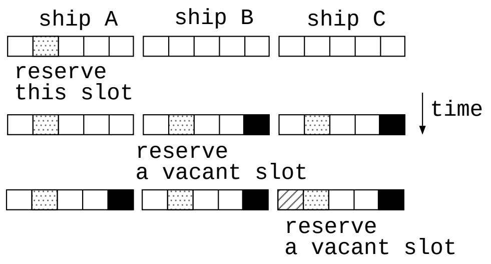

Figure 1. AIS Communication: normal

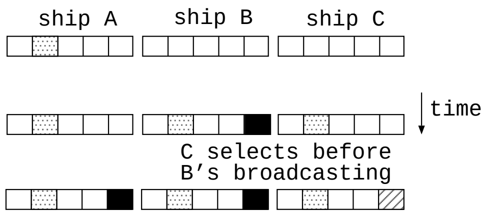

Figure 2. AIS Communication: burst

verify its safety. In section 6 we show the discussion. Finally, in section 7 we present our conclusions.

2 Automatic Identification System

The AIS communication system uses the Self Organized Time Division Multiple Access (SOTDMA) method. This method defines each minute as one frame and each frame is divided into 2250 slots. Each ship broadcasts a message in some slot of every frame. If more than one ship transmits a message in the same slot, data collision occurs and the messages are lost. To avoid such collisions, each ship reserves its own slot in the next frame.

Every ship has its own table in which the reservations for the current frame and the next frame are written. When its turn comes in the current frame, the ship sends a message whilst reserving a slot in the next frame. When a ship receives a message from another ship, it writes the information in its own table and selects its next slot from amongst the unoccupied ones. Basically the same slot is selected before “the slot timeout,” a time limit imposed in advance. After the slot time out, the nearest available slot is selected.

The entire ocean is divided into several smaller areas with a base station and a different frequency is assigned for each area. Only ships in the same area communicate. When a ship moves from one area to another, it resets the current table. Because the number of ships in an area changes frequently, ships cannot determine their slots in advance, but they do so frame by frame.

Figure 1 presents an example of the communication procedure. To begin, ship A reserves its slot. It writes the reservation in its own table and transmits its message. When the other ships receive the message, they write it in their respective tables. Then ship B reserves a vacant slot in its own table. It writes the reservation in its table and transmits its message. When the other ships receive the message, they write it in their respective tables. Ship C then proceeds similarly.

Communication in AIS is not interactive. It is accomplished by broadcasting. One ship cannot hear another’s message whilst sending its own. This mechanism is different from most multi-agent systems and network protocols, which either have a common table or know the status of a partner by receiving an acknowledgment. In AIS, each ship has its own reservation table, and there is a time lag between sending and receiving a message. This may result in more than one ship attempting to transmit information at the same time. More serious, a ship has no awareness of its own failed transmissions. The other ships are aware that a message collision has occurred but do not know which ships caused the collision. These situations are called *bursts*; if they continue, conditions will become very dangerous because ships will not know the positions of other ships.

Figure 2 presents an example of a burst. In this case, ship C selects a vacant slot before receiving ship B’s message. As a result, B and C reserve the same slot. Their tables are different, but both of them believe their reservations have succeeded, and they attempt to transmit information at the same time. This causes the burst.

3 A Probabilistic Model Checker

3.1 Model Specifications

PRISM is a probabilistic model checking tool developed at the University of Birmingham [11]. It is designed to handle unreliable or unpredictable phenomena which occur frequently in the real world. It is based on a Markov chain that is a sequence of trials in which the probability of any event depends only on the current state and not on previous states. PRISM supports three types of probabilistic models: discrete Markov chains (DTMCs), Markov decision processes, and continuous-time Markov chains. In a DTMC, time is modelled in discrete time steps, and the transition probabilities are also discrete. Such processes are suitable for analysing systems with simple probabilistic behaviour and no concurrency. Markov decision processes extend DTMCs by permitting a combination of nondeterminism and probability and are well suited to modelling multiple probabilistic processes occurring in parallel or cases in which some system parameters or environmental behaviours are unknown. Continuous-time Markov chains do not support nondeterminism but model time in a continuous fashion. Here, we use DTMC.

The behaviour of multiple processes can be written as a parallel composition of reactive modules. Each module is defined by a set of finite-ranging variables, and its be-

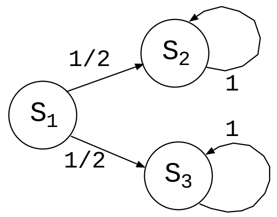

Figure 3. Example of the behaviour of probabilistic system

```
dtmc
module sample
 q: bool init true;
 r: bool init false;
  [](r=false)->
      1/2:(q'=true)&(r'=true) + 1/2:(q'= false)&(r'=true);
  [](r=true) -> true;
endmodule
```

Figure 4. PRISM code for the system in Figure 3

haviour is described by a set of probabilistic guarded commands.

For example, consider the transition system illustrated in Figure 3. There are three states: S_1, S_2, and S_3. $\neg q \wedge \neg r$ holds on S_1, $\neg q \wedge r$ holds on S_2, and $q \wedge r$ holds on S_3. Transitions from S_1 to S_2 and to S_3 occur with the same probability. The PRISM code for this system is shown in Figure 4.

3.2 Property Specification

The properties of a model are specified in terms of Probabilistic Computational Tree Logic (PCTL) for DTMC.

PCTL is an extension of CTL [2] so that probabilistic events can be handled. An interpretation structure for PCTL is a quadruple $K = \langle S, s_0, \mathcal{T}, L\rangle$ where S is a finite set of states, $s_0 \in S$ is an initial state, $\mathcal{T}$ is a transition probability function $\mathcal{T} : S \times S \rightarrow [0,1]$, and for all $s \in S$, $\sum_{s' \in S} \mathcal{T}(s,s') = 1$, and L is a labelling function, $L : S \rightarrow 2^A$ that assigns an atomic proposition to each state. *A path* σ from a state s_0 is an infinite sequence of states $s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n \rightarrow \ldots$. For each path $\sigma = s_0, s_1, \ldots$, the probability that a formula holds on σ is computed by $\Pi_{i=0}(\mathcal{T}(s_i, s_{i+1}))$. PCTL formulas are interpreted over DTMCs, and the state transition is considered to be a step.

Pure PCTL allows only U(until) as a temporal operator, but the current version of PRISM permits several extensions. We explain the syntax of PCTL in PRISM.

Let α be a propositional formula, p a probability (i.e., a number in the interval [0,1]), $\beta \in \{<, \leq, >, \geq\}$, and λ a time step (i.e., a natural number or ∞). Formulas in PRISM are divided into two classes, *state formulas* (denoted by ϕ) and *path formulas* (denoted by ψ), in a manner analogous to CTL.

$$\phi ::= \alpha | \neg\phi | \phi_1 \vee \phi_2 | \phi_1 \wedge \phi_2 | \phi_1 \Rightarrow \phi_2 | P_{\beta p}[\psi]$$
$$\psi ::= X\phi | \phi_1 U^{\leq\lambda} \phi_2 | G^{\leq\lambda}\phi | F^{\leq\lambda}\phi$$

Four temporal operators represent path properties: X (next), U (until), G (globally), and F (eventually). Their intuitive meanings are as follows:

- $X\phi$: ϕ holds in the next state.
- $\phi_1 U^{\leq\lambda}\phi_2$: If ϕ_2 holds within λ steps from the current state, then ϕ_1 is preserved until that state.
- $G^{\leq\lambda}\phi$: ϕ holds in all states within λ steps from the current state.
- $F^{\leq\lambda}\phi$: ϕ holds in some state within λ steps from the current state.

A probabilistic event is denoted by $P_{\beta p}[\psi]$, which means that the path formula ψ occurs with probability βp. For example, $P_{\leq 0.25}[G^{\leq 10}\phi]$ means that "ϕ remains true within 10 steps from a state s with probability 0.25 or less." Here *a state* s means any state that appears within 10 steps, unless a particular state is specified. Accordingly, this really means that "for every state s, ϕ remains true within 10 steps from s with probability 0.25 or less." In contrast, $\phi' \Rightarrow P_{\beta p}[\psi]$ means that "for every state in which ϕ' holds, ψ occurs from this state with probability βp". It is also permissible to write this in the form $''name'' \Rightarrow P_{\beta p}[\psi]$ by giving the name "name" to the state that satisfies ϕ'.

Consider the following two properties of the state transition system shown in Figure 3.

1. $P_{\geq 0.5}[G^{\leq 2}q]$
   q holds on the subsequent two states with probability 0.5 or more.
2. $P_{\leq 0.5}[G^{\leq 2}q]$
   q holds on the subsequent two states with probability 0.5 or less.

If we verify these properties, PRISM returns FALSE for both cases. This is correct, although it is counterintuitive. If state S_1 is regarded as an initial state, then $[G^{\leq 2}q]$ holds with probability 0. If state S_2 is regarded as an initial state, then $[G^{\leq 2}q]$ holds with probability 1. It follows that neither property holds for every state.

PRISM also computes the probability that ψ holds, denoted by $P = ?[\psi]$. It does this by examining all paths starting from every state. For example, $P = ?[F^{\leq 10}[\phi]]$ means that "for every state s, compute the probability that ϕ holds within 10 steps from s." The probability is computed independently for every state, and all results are stored in a log file. The user can inspect the log file, but only one case is displayed on the console. For example, consider the following property of the state transition system shown in Figure 3:

$P = ?[G^{\leq 2}q]$

This expression means "compute the probability that q continues to hold in the subsequent two states." If, for example, the value 0.5 is displayed as a result, it represents the probability when S_1 is regarded as an initial state.

If $P\ =?[\psi\ \{\phi'\}]$ is entered, PRISM examines all paths starting from every state in which ϕ' holds. It is also permissible to write this in the form $P\ =?[\psi\ \{"name"\}]$ by giving the name *"name"* to the state that satisfies ϕ'.

3.3 Tools

PRISM provides several GUI tools, including a graphic representation of the simulation analysis. In principle, it is impossible to generate a counterexample when verification fails, because all paths are considered in computing a probability, and a single path cannot be chosen. Hence, one is obliged to determine the reason for failure by inspecting the log file. In a sense, this is a drawback of PRISM as a model checker, because it does not identify the part to be corrected. Instead, it affords a view-specific trace of model execution, which is very useful for debugging.

4 System Modelling

We construct several models for a reservation system described in section 2. To avoid state explosion during the execution, we simplify the model in two respects. First, we ignore information other than slot reservation information. Second, we construct a synchronous model, whereas the actual system behaves asynchronously. In the actual system, each ship refers to its own table and determines the next slot asynchronously, prior to message transmission. The table is updated incrementally. In our model, each ship refers to its own table in the current frame and determines the next slot synchronously.

We construct a model **real** which simulates the behaviour of the actual reservation system most closely. "The slot timeout" is set to six.

real: Set $p_t = 0.8 - 0.16t$ ($t = 0, 1, 2, \ldots$) and $p_t = p_{t'}$ if $t = t'(mod\ 6)$. The same slot is selected with probability p_t and the adjacent slots are selected both with probability $(1 - p_t)/2$ at each step t.

Note that if the number of slots is large, the probability of a burst is low, but it is never equal to 0.

We also construct *strategic models* that execute the following functions:

- avoiding slots that are already reserved by other ships,
- performing cyclic reservation,
- adding the probabilistic factor.

A slot is referred to by its position. If there are k slots, the first slot is on the right, and the k-th slot is on the left. $(k+1)$-th slot is identified with the first slot. The $(i+j)$-th and $(i-j)$-th slots are said to be *n-neighbours of the i-th slot* when $1 \leq j \leq n(< k)$.

In our strategic models, each ship compares the numbers of occupied slots within n-neighbours on either side of the slot which is occupied by itself in the current frame (called *a current slot*). The adjacent slot on the less congested side is selected with probability p, and the current slot and the adjacent slot on the other side are selected both with probability $(1-p)/2$. If the number of occupied slots is the same on both sides, one of these three slots is selected at random.

We construct four strategic models based on different strategies:

prob1: probabilistic 1-neighbour model ($n = 1, p = 0.8$)

det1: deterministic 1-neighbour model ($n = 1, p = 1$)

prob2: probabilistic 2-neighbour model($n = 2, p = 0.8$)

det2: deterministic 2-neighbour model ($n = 2, p = 1$)

We perform the analysis and verification for each model using PRISM, and investigate their effectiveness.

5 Analysis and Verification

We first compute and analyse probabilities for several specifications, and then verify the specifications with probabilities. In the process of computation and analysis, the probability of the occurrence of an event for a particular state is calculated and analysed independently, whereas in the process of verification, the corresponding probability for all states is used. The version of PRISM used is PRISM3.3beta.

5.1 Computation and Analysis

Taking into account the actual ratio of the number of ships to the number of slots, we set the number of ships equal to 3 and the number of slots equal to 10 and 15. The slots occupied by three ships are represented by an ordered triple. Four patterns, (1,2,3), (1,3,5), (1,3,7), and (1,4,7), are used as the initial state. The objective of the analysis is to inspect the probability for each data set and find the strategy that yields the lowest probability that a burst will occur. We give the name "bst" to a state in which a burst occurs and "init" to an initial state. bst_sum is a variable denoting the number of "bst" states.

spec1 $P =?[trueU^{\leq 20}bst\_sum \geq 5\ \{``init"\}]$

First, we compute the probability that a burst occurs more than five times within 20 steps, since burst should be avoided as much as possible.

The results for 10 slots are shown in Figure 5.

For the four strategic models, the probability does not exceed 0.09. In contrast, the probability is 0.7 for the real model (data not shown in the graph). This demonstrates that the strategic models are much more effective than the real model. The more sparsely occupied the slots in the initial state, the better the results. Hence, when the selected

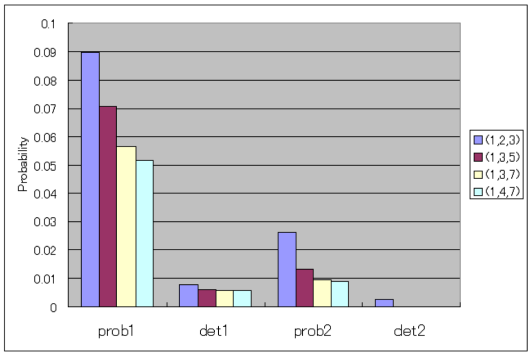

Figure 5. Computed probabilities (spec 1)

slots are congested, a burst is more likely to occur. Deterministic selection produces better results than probabilistic selection, and the 2-neighbour strategy yields better results than the 1-neighbour strategy. These observations seem to imply that slot inspection over a wider range produces better results, but this is not always the case. Suppose a 3-neighbour strategy is adopted. When the i-th slot is the target slot, and the $(i-1)$-th, $(i+2)$-th, and $(i+3)$-th slots are all occupied, the $(i-1)$-th slot will be selected as the next slot, resulting in a burst. It follows that a 3-neighbour strategy is too wide for this situation, and a suitable number of neighbours should be determined in accordance with the number of slots.

If the number of slots is chosen to be 15, the probability is lower for all models than when the number of slots is 10.

spec2 $P = ?[G^{\leq 2} \text{“}bst'' \, \{''bst''\}]$

Second, we compute the probability of two subsequent occurrences of burst, since the system should not remain in a state of burst.

In this case, the initial state is chosen randomly, because the probability does not depend on the initial conditions.

The results for 10 slots are shown in Figure 6.

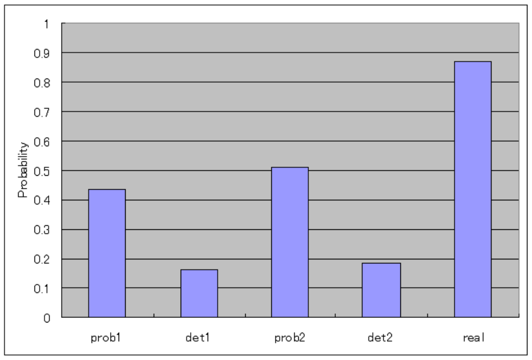

Figure 6. Computed probabilities (spec 2)

Comparing the determistic selection and probabilistic selection, the former yieds better results. The result of the real model is worst. In strategic models, the less congested side is selected by inspecting the neighborhood when burst occurs, while the situation of congestion is ignored in the real model. This result shows the effectiveness of neighborhood inspection. There is little difference between the 1-neighbour strategy and the 2-neighbour strategy.

If the number of slots is chosen to be 15, the probabilities are almost the same as when the number of slots is 10.

spec3 $P = ?[G^{\leq 10} bst\_sum = 0 \, \{''init''\}]$

Third, we compute the probability that a burst never occurs within 10 steps.

The results for 10 slots are shown in Figure 7.

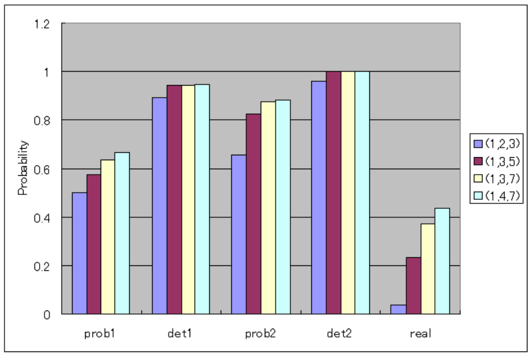

Figure 7. Computed probabilities (spec 3)

The probability is higher in the strategic models than that for the real model. Specifically, when the selection is deterministic, the probability is almost equal to 1. This means that a burst can be avoided with high probability in the strategic models. The 2-neighbour strategy yields better results than the 1-neighbour strategy, and this is because the congested part of a frame can be detected early by inspecting a wider range of slots; hence, burst is more likely to be avoided.

If the number of slots is chosen to be 15, the probabilities are slightly higher than when the number of slots is 10, because the density of selected slots is lower.

In Table 1, the probabilities for all the strategies are compared. In this table, ”prob” denotes probabilistic selection and ”det” denotes deterministic selection. Inequality symbols indicate which strategy or pattern yields a better result. The symbol $\ll$ indicates that the strategy on the right produces much better results than the strategy on the left, whereas $\approx$ indicates that the results of both strategies are nearly the same. From this table, the following observations can be made:

- Deterministic selection is better than probabilistic selection.

spec	selection		n-neighbour		init		#slots		probability	
	prob	det	1	2	sparce	congested	10	15	strategic	real
1	$\ll$		$\ll$		$>$		$<$		0.0001-0.09	0.70
2	$\ll$		$>$		$-$		$\approx$		0.16-0.15	0.87
3	$<$		$<$		$>$		$<$		0.45-1.0	0.30

Table 1. Comparison of strategies (analysis)

- When the initial state is sparsely occupied, better results are obtained.
- When the number of slots is larger, better results are obtained.
- The strategic models are more advantageous than the real model in all cases.

5.2 Verification

To verify that communication succeeds with high probability, we break down this requirement into three more specific ones based on the analysis: (i) A burst seldom occurs. (ii) A subsequent burst seldom occurs. (iii) A burst seldom occurs following a burst-free state. The number of ships is three, and the number of slots is 10.

spec4 $\quad P_{\geq X}[F^{\leq N} \neg''bst'']$

First, we verify that successful communication has a probability of X or more. The specification signifies that the property holds for every state.

The verification is performed for $N = 3$ and $N = 20$ and for various values of X.

Table 2 displays the results for spec 4. In the table, the figures in the TRUE column are the minimum values of X when TRUE is returned, and the figures in the FALSE column are the maximum values of X when FALSE is returned. Hence, the probability threshold is between these values.

model	(N=3)		(N=20)
	TRUE	FALSE	TRUE
prob1	0.78	0.79	0.99
det1	0.71	0.72	0.99
prob2	0.74	0.75	0.99
det2	0.71	0.72	0.99
real	0.31	0.32	0.99

Table 2. Verification results (spec 4)

When $N = 3$, the probability is more than 0.71 in the strategic models, which means that there exists a state in which the property does not hold. Amongst the strategic models, probabilistic selection yields slightly better results than deterministic selection. The reason for this is explained as follows. When multiple ships reserve the same slot in the current state, they all behave in the same way, i.e., all of them select the same slot on the non-congested side as the next slot. Hence, another burst occurs. If the occupancies of the left n-neighbour and that of the right n-neighbour are the same, slot selection among the current one and the adjacent ones is performed at random. This is the only case in which burst disappears in the deterministic selection. When a burst state or a high-density state is chosen as the initial state, a factor of randomness should be added to avoid repetition of the same behaviour. This is why the probabilistic selection produces better results. The 2-neighbour strategy yields slightly better results than the 1-neighbour strategy.

When $N = 20$, the communication is almost always successful at least once, and this implies that 20 frames are adequate to ensure safety when there are three ships and 10 slots.

spec5 $\quad ''bst'' \Rightarrow P_{\leq X}[G^{\leq N}\, ''bst'']$

Second, we verify that if a burst occurs, there is a probability of X or less that it continues to occur in the subsequent N frames.

Table 3 displays the results for spec 5. In the table, the figures in the TRUE column are the maximum values of X when TRUE is returned, and the figures in the FALSE column are the minumim values of X when FALSE is returned.

model	(N=3)		(N=20)
	TRUE	FALSE	TRUE
prob1	0.25	0.24	0.01
det1	0.29	0.28	0.01
prob2	0.29	0.28	0.01
det2	0.29	0.28	0.01
real	0.79	0.78	0.01

Table 3. Verification results (spec 5)

When $N = 3$, the probability is less than 0.29 for the strategic models while it is less than 0.79 for the real model. Amongst the strategic models, prob1 produces the best result, although it is not significantly better than those of the other strategic models. The number of n-neighbours has little effect.

When $N = 20$, a subsequent burst almost never occurs in all models; this means that 20 frames are adequate to ensure safety when there are three ships and 10 slots.

spec6 $\neg''bst'' \Rightarrow P_{\leq X}[F^{\leq N}$ “bst'']

Third, we verify that the probability is X or less that a burst occurs following the state without a burst.

Table 4 shows the result of spec 6. In the table, the figures in the TRUE column are the maximum values of X when TRUE is returned, and the figures in the FALSE column are the minumim values of X when FALSE is returned.

model	(N=3)		(N=20)	
	TRUE	FALSE	TRUE	FALSE
prob1	0.4	0.39	0.75	0.74
det1	0.21	0.2	0.3	0.29
prob2	0.21	0.2	0.45	0.44
det2	0.13	0.12	0.12	0.11
real	0.95	0.94	0.999	0.99

Table 4. Verification results (spec 6)

All of the strategic models produce better results than the real model. Amongst the strategic models, deterministic selection yields better results than probabilistic selection. The 2-neighbour strategy yields better results than the 1-neighbour strategy. The reasons for these results are the same as in the case of spec 3.

If deterministic selection is used, the probability is almost the same for both $N = 3$ and $N = 20$. This implies that a burst can be avoided if it does not occur at an early stage, and that the states will be stable afterward. In contrast, if probabilistic selection is used, the probability is much higher when $N = 20$ than when $N = 3$. This implies that randomness causes a burst to occur.

5.3 Evaluation

We present an evaluation of all experimental results. In Table 5, the verification strategies are compared. From this table, the following observations can be made:

- Once a burst has occurred, it is better to select a slot probabilistically.
- If a burst does not occur, it is better to select a slot deterministically.
- In general, the 2-neighbour strategy is better than the 1-neighbour strategy.
- Even if bursts occur, most will disappear within 20 frames.
- The strategic models are more advantageous than the real model in all cases.

When $N = 3$, with prob1, the probability that a burst occurs is more than 0.78, and the probability of two subsequent bursts is less than 0.25. These are the best results for spec 4 and spec 5, but they differ little from the results of the other models. However, with det2, the probability that a burst occurs following a burst-free state is less than 0.13. This is the best result for spec 6, and it differs greatly from the results of the other models.

When $N = 20$, the probabilities are 0.99 for spec 4 and 0.01 for spec 5 in all models including the real model.

Overall, det2 offers the best strategy. Although it is not guaranteed to be effective if the number of slots or the number of ships is changed, it does suggest that there may be a better strategy than the one currently in use.

6 Discussion

There have been few studies of the safety of AIS via formal methods. The primary drawback of this communication system is that ships cannot be sure whether their messages have been successfully transmitted. This communication mechanism resembles that of the network communication protocol CSMA/CD. In CSMA/CD, there is a certain probability that a message may be lost, but the sender is not directly aware of the failure. The failure is known only by timeout, and several attempts are made to send the message. This behaviour has been modelled, and the relationship between message collisions and delay has been analysed probabilistically with PRISM [6, 10]. The difference between our works and theirs is that each ship knows the state of the current reservation of other ships and that state effects on its next behavior in the strategic models in AIS, whereas processes communicating by a unique channel do not know the states of other processes in CSMA/CD. That is, behaviors of processes directly affect those of other processes in AIS, which is almost impossible to compute the probability in a long term manually.

Several case studies of PRISM have been performed: systems biology, communication protocols, dynamic power management and so on [13]. However, in these works, the emphasis has been on analysis rather than verification and there has not yet been sufficient discussion of how probabilistic systems should be handled by a probabilistic model checker. In this paper, we have demonstrated the use of PRISM first for independently computing a probability for a single initial state with varied parameters and then for verifying a specification for all initial states.

Many practical systems have probabilistic factors in their behaviour. For simple behaviour, one can manually compute the probability that a specific event occurs. However, this procedure is annoying and is often beset with errors, especially when a complicated system or multiple conditional behaviours are involved. Moreover, the correspondence between an event and a behaviour at each state is not always clear from manual calculations. A probabilistic model checker is a powerful tool since it computes probabilities in accordance with behaviour analysis, and it also guarantees the correctness of the behaviour.

spec	$N = 3$			$N = 20$		
	selection	n-neighbour	model	selection	n-neighbour	model
	prob det	1 2	strategic real	prob det	1 2	strategic real
4	$>$	$>$	$\gg$	$\approx$	$\approx$	$\approx$
5	$>$	$>$	$\gg$	$\approx$	$\approx$	$\approx$
6	$\ll$	$\ll$	$\gg$	$\ll$	$\approx$	$\gg$

Table 5. Comparison of strategies (verification)

7 Conclusion

We have presented an analysis and verification of AIS using the probabilistic model checker PRISM. It has shown that the actual AIS protocol can be improved. Our contributions are twofold: we are first to apply a formal method to AIS which requires safe behaviour at high probability, and we have given an effective usage of verification procedure for probabilistic systems in a probabilistic model checker.

PRISM is said to be capable of handling 10^9 states [11]. However, in principle the system constructs models for all reachable states and explores them, and this can readily result in state explosion. When asynchronous behaviour is involved, the problem becomes serious. Recently, state reduction methods in PRISM have been proposed [12, 14]. This reduction called *symmetry redution* can be used for AIS, since ships in AIS is interchangeable and the frame has a cyclic structure. It is prosperous that this enables verification on a factorially smaller model. In future, we will apply the strategic models up to a realistic problem size and perform verifiction on the reduced model.

References

[1] Japan Coast Guard: *AIS: Universal Ship-Borne Automatic Identification System* (1999).

[2] Bérard,B., M.Bidoit, A.Finkel, F.Laroussibie, A.Petrucci, Ph.Schnoebelen and P.Mckenzie: *Systems and Software Verification - Model-Checking Techniques and Tools -* . Springer (1999).

[3] Behrmann,G. A.David and K.Larsen: A Tutorial on UPPAAL. *Formal Methods for the Design of Real-Time Systems Information*, pp.200-236 (2004).

[4] Cimatti,A., E.Clarke, E.Giunchiglia, F.Giunchiglia, M.Pistore, M.Roveri, R.Sebastiani and A.Tacchella: NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. of International Conference on Computer-Aided Verification (CAV 2002)*, pp.27-31 (2002).

[5] Clarke,E., O.Grumberg and D.A.Peled: *Model checking, Computer Performance Evaluation: Modeling Techniques and Tools*. The MIT Press (2001).

[6] Duflot,M., L.Fribourg, T.Herault, R.Lassaigne, F.Magniette, S.Messika, S.Peyronnet and C.Picaronny: Probabilistic Model Checking of the CSMA/CD Protocol Using PRISM and APMC. *Electronic Notes in Theoretical Computer Science*, Vol. 128, No. 6, pp.195-214 (2005).

[7] Hansson,M. and B.Jonsson: A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing*, Vol. 6, pp.512-535 (1994).

[8] Herault,T., R.Lassaigne, F.Maniette and S.Peyronnet: Approximate Probabilistic Model Checking. In *Proc. of Fifth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI04)*, pp.73-84 (2004).

[9] Hermanns,H., J-P.Katoen, J.Meyer-Kayser and M.Siegel: A Markov Chain model Checker. In *6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS00)*, pp.347-362 (2000).

[10] Kwiatkowska,M., G.Norman and J.Sproston: Probabilistic Model Checking of the IEEE802.11 Wireless Local Area Network Protocol. In *Second International Workshop PAPMPROBMIV 2002*, pp.169-187 (2002).

[11] Kwiatkowska,M., G.Norman and D.Parker: PRISM: Probabilistic Symbolic Model Checker. In *Computer Performance Evaluation: Modeling Techniques and Tools*. pp.113-140 (2002).

[12] Kwiatkowska,M., G.Norman and D.Parker: Probabilistic Model Checking in Practice: Case Studies with PRISM. *ACM SIGMETRICS Performance Evaluation Review*, Vol.32, No.4, pp.16-21(2005).

[13] Kwiatkowska,M., G.Norman and D.Parker: Symmetry Reduction for Probabilistic Model Checking. In *18th Conference on Computer Aided Verification (CAV06)*, pp.234-248 (2006).

[14] Kattenbelt,K., M.Kwiatkowska, G.Norman and D.Parker: Game-Based Predicate Abstraction in PRISM. *Proc. of 6th Workshop on Quantitative Aspects of Programming Languages (QAPL08)*, pp.5-21 (2008).

[15] Sen,K., M. Viswanathan and G. Agha: On Statistical Model Checking of Stochastic Systems. In *17th Conference on Computer Aided Verification (CAV05)*, pp.266-280 (2005).

[16] Younes,H:: Ymer: A Statistical Model Checker. In *17th International Conference on Computer Aided Verification (CAV05)*, pp.429-433 (2005).