Fault Tolerant Multiple Observers Using Error Control Codes

Guevara NOUBIR, Berthe Y. CHOUEIRY and Henri J. NUSSBAUMER
Department of Computer Science
Swiss Federal Institute of Technology in Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{noubir|choueiry|nussbaumer}@di.epfl.ch
Tel : +41-21-693.27.87 Fax : +41-21-693.47.01

Abstract

We address the problem of detecting execution errors in communication protocols. A communication protocol is modeled as a Finite State Machine (FSM) that can be used as an external observer for detecting execution errors. In [16, 17], Wang and Schwartz introduce the concept of multiple observers obtained by an adequate decomposition of the FSM. In this paper, we first address the decomposition procedure from the perspective of error control codes and show that the decomposition algorithm can be restated as a simple state coding algorithm. Then, we discuss the features of fault tolerance of the resulting decomposition. We generalize the concept of multiple observers into the one of fault tolerant *multiple observers. A set of observers is said to be fault tolerant if it is capable of detecting the execution errors of a protocol even when a subset of the observers is faulty. We show that error control codes can be used to generate multiple observers that are fault tolerant. We illustrate our approach on the ISO transport protocol class 4 (TP4). Finally, we give some hints on how to assign codes to the states while maximizing the fault coverage of the resulting decomposition.*

1 Introduction

Communication protocols are a central mechanism for enabling communication among the distributed entities of a computer network. First, these protocols are specified then they are implemented. At a later stage, they are tested to ensure that they comply to the initial specifications and are free from coding errors. Finally, they are integrated into a real system and made available for use.

Despite the various stages of testing before implementation, it is often true that some errors remain undetected and appear only when the protocol is made operational. Other errors may also appear when an underlying software or hardware component of the protocol is modified. Finally, execution errors may be induced by the physical and software environment itself (e.g., memory errors, electro-magnetic perturbations, etc.). If errors are not adequately handled when they occur during the execution of the protocol, they may cause a snowball effect and result in a more or less long interruption of communications, which may yield important financial losses.

Various techniques and strategies [16, 17, 13, 3] for handling execution errors in communication protocols are proposed in the literature. Some strategies detect the errors and then proceed to a diagnosis task in order to fix them afterwards. Other strategies try to correct the errors on-line then resume a normal execution of the protocol. Fig. 1 illustrates both strategies.

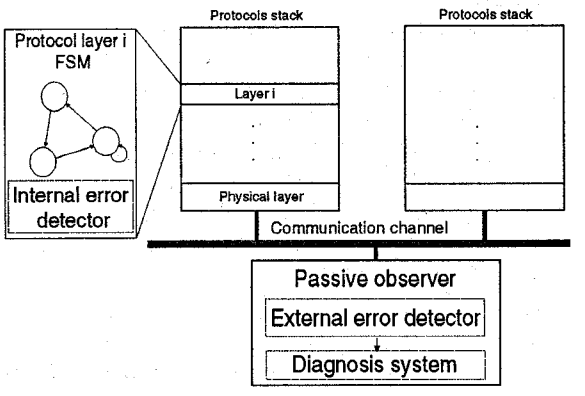

Figure 1. *Communication protocols are generally modeled as finite state machines.* Two main strategies for error detection exist: *passive* (external) that aims at diagnosing the errors and *active* (internal) that aims at correcting the errors.

When error detection is carried out in real-time (online), the mechanism for fault detection is required to be efficient. In this context, efficiency means high execution speed and low complexity. Efficiency becomes a critical issue when the error detection mechanism is integrated into the communication protocol itself.

In this paper, we address the problem of error detection a in communication protocol, which we model as a Finite State Machine (FSM). The entity that carries out the error detection process is composed of *multiple* independent observers. Together, these observers can detect the set of possible faults.

This paper is organized as follows. First, we summarize the state of the art of techniques for detecting execution errors in communication protocols (Section 2). In Section 3, we recall the technique of multiple observers, which we generalize in this paper. We introduce our contribution in Sections 4 and 5 and illustrate it through an example of the ISO transport protocol class 4 (TP4). Finally, we give a few hints on how to reduce the amount of non-determinism in the decomposed sub-FSMs.

2 Methods for fault detection in communication protocols

In this section, we recall some techniques proposed in the literature for low cost detection of execution errors in communication protocols.

2.1 Signature analysis

In [8, 7, 6], we show how the signature analysis of the paths of a control flow graph can be used to detect execution errors of communication protocols. The basic idea is based on the following mechanism: at each control state, the signature of the current path must be equal to the signature of a correct path leading to the current state.

In order to enhance the efficiency of this technique, Leveugle [5] proposes to encode the states of the FSM in such a way that all signatures of correct paths leading to a same state are equal. Thus, every state has one *static* entering signature, which is compared to the signature computed during execution. In [8, 7, 6], we introduce new signature functions based on the evaluation of polynomials. These functions have the following advantages: (1) They can be efficiently implemented for software applications. (2) They are suited for an algebraic optimization of the signature controller.

Although the signature approach is very efficient, it suffers from being a probabilistic approach and may require a modification of the original FSM. Consequently, it is particularly suited for those cases where the computational complexity of error detection is an important issue. This holds, for instance, when the error detection mechanism is carried out by the entity that executes the protocol itself (self-checking).

2.2 Abstraction

Oikonomou [11, 12] proposes to use an abstraction technique in order to reduce the complexity of the observer. An abstract FSM is obtained from the initial FSM by aggregating several states into one. Note that the abstract machine obtained by this mechanism allows only the detection of a fraction of all possible errors. Oikonomou distinguishes several error types (e.g., immediately detectable errors, statistically detectable errors, undetectable errors, etc.). However, the problem of building (or even approximating) the *smallest* FSM that allows the detection of a given error type often appears to be **NP**-hard. Moreover, the user cannot control the complexity of the resulting abstract machines.

Lee et al. [4] propose abstraction techniques to reduce the complexity of the implementation of a communication protocol. Their contribution is based on the following idea. A given communication protocol offers a set of services. If one is interested only in a subset of the provided services, it is possible to extract, from the original implementation, a partial one that offers only the required services. This partial implementation has a lower complexity and higher efficiency than the original one. This specialization or abstraction technique is also called *protocol prunning* or *protocol thinning*. It can also be useful for conformance testing. However, since this technique yields a deterministic implementation, it cannot reduce the complexity of an optimized implementation that offers all services of the considered protocol. The adaptation of these techniques to the detection of execution errors has not been, in our knowledge, addressed by the community and is a potentially promising direction for future research.

2.3 A fault tolerant TTP protocol

Kopetz and Grünsteidl [3] propose a fault tolerant communication protocol for real-time systems. One of the techniques used by their protocol consists of systematically verifying that the state variables of the entities involved in the communication process are all identical. This operation is carried out without increasing the size of the protocol frames by computing the Cyclic Redundancy Code (CRC) of the concatenation

of the transmitted data and of the values of the state variables. When a node in the network is in a faulty state (i.e., the state variables are incompatible with those of other nodes), the CRC is different from the one expected by the other nodes. This is illustrated in Fig. 2. This technique is interesting because it allows the detection of some errors without increasing the size of the communication frames. Its disadvantage is that it requires the cooperation of the two communicating entities and the modification of the protocol that is monitored.

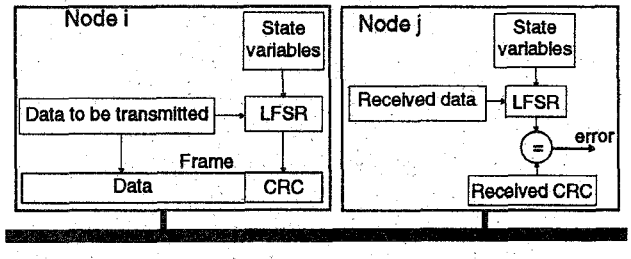

Figure 2. *In the TTP protocol,* the CRC is computed (using a Linear Feedback Shift Register, LFSR) over the transmitted data and the values of the state variables, which must be identical in all nodes. When a frame is received, a node recomputes the CRC using its own values for the state variables. This allows us to simultaneously detect communication errors and verify the coherence of state values across the nodes.

3 The technique of multiple observers

Wang and Schwartz [16, 17] propose a very interesting technique for detecting the execution errors of communication protocols using multiple external observers. The communication protocol is modeled as an FSM. The multiple observers are obtained by an adequate decomposition of the FSM. An error is detected whenever at least one observer is in a state such that there is no transition starting at this state and labeled with the received event. This technique reduces the computational complexity of error detection and exhibits an interesting behavior of *graceful degradation* when the detection mechanism itself is faulty. Indeed, as long as a subset of the observers remains operational, a subset of the errors can still be detected. This technique has been extended by Vijayananda [15] for the detection of an even bigger class of errors through the use of extended finite state machines.

In this section, we introduce the technique of multiple observers of Wang and Schwartz that we generalize in the rest of the paper.

3.1 Basic principal

The basic principal of the multiple observers technique is to decompose the studied FSM of N states into $\lceil \log_2 N \rceil$ independent FSMs each comprising two states and thus called 2-FSM, as shown in Fig. 3. Since

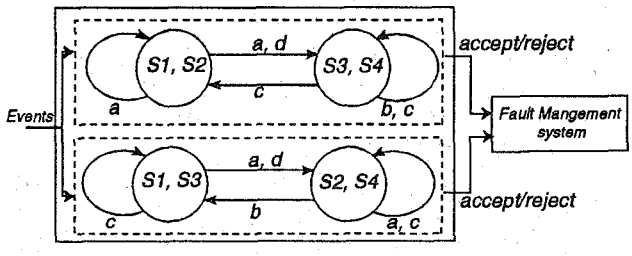

Figure 3. *Error detection mechanism using multiple observers.*

the number of the resulting 2-FSMs is very small with respect to the total number of states and since the computational complexity of each 2-FSM is very low, the computational complexity of the error detection mechanism is considerably reduced. Moreover, since the 2-FSMs are independent, it remains possible to detect a subset of the occuring errors even when some 2-FSMs become faulty.

The disadvantage of this technique is that some non-determinism may appear at the level of the 2-FSMs because of the decomposition process[1]. In order to cope with this non-determinism during execution, Wang and Schwartz propose to process the events to eliminate the non-determinism before sending them to the 2-FSMs. The resolution of the non-determinism is made possible through the use of a buffer that memorizes all ambiguous event sequences until ambiguity can be resolved.

The decomposition algorithm proposed in [16, 17] for building m 2-FSMs out of the initial FSM of N states (with $m = \lceil \log_2 N \rceil$) is the *perfect shuffle* algorithm recalled in Section 3.2. Each 2-FSM is built by *partitioning* into two sets the N states of the initial FSM.

A finite state machine A is defined as follows: $A = (Q, \Sigma, \delta)$; where Q is the set of states ($|Q| = N$), Σ is the set of events (vocabulary), and δ is the transition function. A is decomposed into m machines $A_1, \ldots, A_m$. Each machine $A_i = (\{Q_1^i, Q_2^i\}, \Sigma, \delta_i)$ is chosen such that Q_1^i and Q_2^i are a partition of Q, the set of states of A. Consequently, $Q_1^i \cup Q_2^i = Q$ and $Q_1^i \cap Q_2^i = \emptyset$. The transition function of A_i is chosen

[1] As a reminder, non-determinism occurs when, starting from a given state, it is possible to reach two or more different states at the occurrence of a given event.

as follows:

$$\forall j,k \in \{1,2\}, \forall E \in \Sigma;\ \delta_i(Q_j^i, E) = Q_k^i \\ \Leftrightarrow \exists S_u \in Q_j^i, S_v \in Q_k^i \mid \delta(S_u, E) = S_v$$

Informally speaking, a transition from the state Q_j^i to the state Q_k^i occurs at the event E, when a transition happens in the machine A from a state $S_u \in Q_j^i$ to a state $S_v \in Q_k^i$ at the occurrence of the event E.

Furthermore, an *optimal* decomposition is obtained when each state S of the machine A is uniquely determined by a combination of states of the machines A_i.

3.2 Perfect Shuffle algorithm

In this section, we recall the perfect shuffle algorithm proposed by Wang and Schwartz [16] for decomposing an FSM of N states, with $N = 2^m$. When N is not a power of 2, an adequate number of "ghost" states[2] is added to N in order to make it a power of 2. The decomposition algorithm generates $m = \log_2 N$ 2-FSMs. The elements of a state Q_j^i are ordered and denoted $q_{j,1}^i, \ldots, q_{j,n/2}^i$. The algorithm generates partitions that are symmetric (i.e., with states Q_j^i of equal size).

Algorithm 1 *[17] Perfect shuffle(Q)*

$Q_1^1 = \{S_1, \ldots, S_{N/2}\}$; $Q_2^1 = \{S_{N/2+1}, \ldots, S_N\}$;
For $i = 1$ **To** $m - 1$ **Do**
$Q_1^{i+1} = \{q_{1,1}^i, q_{2,1}^i, \ldots, q_{1,N/4}^i, q_{2,N/4}^i\}$;
$Q_2^{i+1} = \{q_{1,N/4+1}^i, q_{2,N/4+1}^i, \ldots, q_{1,N/2}^i, q_{2,N/2}^i\}$;
end

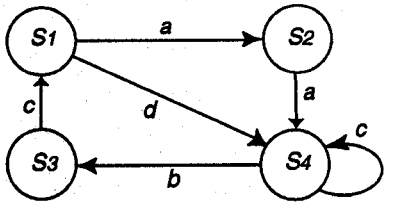

Figure 4. *An example of a Finite State Machine.*

Consider the FSM of Fig. 4 with $Q = \{S_1, S_2, S_3, S_4\}$. The application of the Algorithm 1 reported above yields the following decomposition:

$$Q_1^1 = \{S_1, S_2\} \quad Q_2^1 = \{S_3, S_4\} \\ Q_1^2 = \{S_1, S_3\} \quad Q_2^2 = \{S_2, S_4\}$$

[2]The "ghost" states are then eliminated from the decomposed sub-FSMs.

4 Decomposition by state coding

In this section, we show that the Algorithm 1 can be interpreted as the state coding algorithm reported below (Algorithm 2), which is much simpler than the former. Then, we generalize Algorithm 2 in order to allow decompositions into finite state machines of k states (k-FSMs instead of 2-FSMs).

4.1 Decomposition into 2-FSMs by binary coding of states

The basic idea here is to encode each of the Q states of the initial FSM in base 2. Each state can be coded in $\lceil \log_2 N \rceil = m$ bits. Such a coding can be made arbitrarily provided that distinct states are given distinct codes[3]. Then, the states of a machine A_i are constructed by only considering the i^{th} bit of each state in Q, as shown in Algorithm 2.

Algorithm 2 *Decomposition by binary coding*
For $i = 0$ **To** $m - 1$ **DO**
$Q_1^i = \{S_j \in Q \mid$ the i^{th} bit of S_j is equal to 0$\}$
$Q_2^i = \{S_j \in Q \mid$ the i^{th} bit of S_j is equal to 1$\}$
end

This algorithm yields exactly the same result as Algorithm 1. Consider the example in Fig. 4. The Algorithm 2 generates the following codes for the states: $S_1 = 00$, $S_2 = 01$, $S_3 = 10$, and $S_4 = 11$. This yields the states Q_1^1, Q_2^1, Q_1^2 and Q_2^2, as follows (see Fig. 5):

$$Q_1^1 = \{S_1, S_2\} \quad Q_2^1 = \{S_3, S_4\} \\ Q_1^2 = \{S_1, S_3\} \quad Q_2^2 = \{S_2, S_4\}$$

It is easy to check that such a decomposition strategy

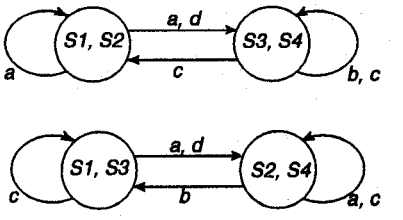

Figure 5. *An example of a decomposition by binary coding.*

uniquely identifies a state. Indeed, two states that belong to the same sets $Q_{j_1}^1, \ldots, Q_{j_{m1}}^m$ must have the same binary encoding, which is impossible.

[3]In Section 6, we argue that an arbitrary coding can lead to non-determinism whereas careful coding can reduce non-determinism.

Theorem 1 *Each state of the protocol is uniquely determined by the state of the 2-FSMs.*

Proof: When the initial machine A is in a state S, the state of each 2-FSM is uniquely determined. This holds since the binary coding of a state selects either 0 or 1 for any bit of the code. Thus, a state S cannot belong to both Q_i^1 and Q_i^2.

Conversely, when the 2-FSMs A_i are in states $Q_1^{j_1}, \ldots, Q_1^{j_m}$, the initial machine A can be in one and only one state. Since each bit of the binary code of the current state of A is defined by the $Q_1^{j_i}$ states, the current state is thus uniquely determined. □.

4.2 Decomposition into k-FSMs using a coding in base k

A k-FSM is a finite state machine of k states. A decomposition into k-FSMs ($k > 2$) yields sub-machines that are more complex than the 2-FSMs obtained by a decomposition by binary coding. However, it is often useful (sometimes necessary) to resolve to such a decomposition for the following reasons:

1. Non-determinism of a decomposition into 2-FSMs reduces the efficiency of error detection (i.e., the capacity to detect errors). A decomposition into k-FSMs reduces the probability of this non-determinism.

2. When the decomposition is required to be tolerant to faulty observers (see Section 5), the codes for error control that need to be used must have a symbol size equal to the number of states in the k-FSMs. A decomposition into 2-FSMs requires binary codes. These codes are difficult to find for arbitrarily chosen Hamming distances. For instance, a decomposition into 4-FSMs (by coding in base 4) generates $\log_4 |Q|$ sub-machines. These 4-FSMs are more complex than 2-FSMs but allow the use of Reed-Solomon codes [2, 10], which can be built for different Hamming distances. Thus, the use of Reed-Solomon codes enables us to tolerate more errors from the observers (see Section 5). Note that, although two decompositions of an FSM, one into 2-FSMs and another one into k-FSMs, are equivalent from the point of view of fault tolerance, the use of k-FSMs enables the use of a larger class of error correcting codes (of symbol size k) and thus can be used when no optimal binary code exists for the desired Hamming distance.

For a decomposition into k-FSMs, one can encode the Q states in base k. The decomposition of an FSM of N states into FSMs of k states generates $\lceil \log_k N \rceil$ finite state sub-machines ($\log_k$ denotes the logarithm in base k).

Note that it is possible to combine decompositions of several bases. Generally speaking, an FSM of N states is decomposable into n t_i-FSMs of respective sizes[4] $t_1, t_2, \ldots, t_n$ if and only if $t_1 \times t_2 \times \cdots \times t_n \geq N$.

For instance, a machine containing 6 states can be decomposed into one machine of 2 states and one machine of 3 states. Fig. 6 shows such a decomposition for the case of the Alternated Bit Protocol (ABP) [1]. The states S_1, S_2, S_3, S_4, S_5, and S_6 are coded as follows:

$$S_1 = 00 \quad S_3 = 02 \quad S_5 = 11$$
$$S_2 = 01 \quad S_4 = 10 \quad S_6 = 12$$

Moreover, the decomposition into a combination of

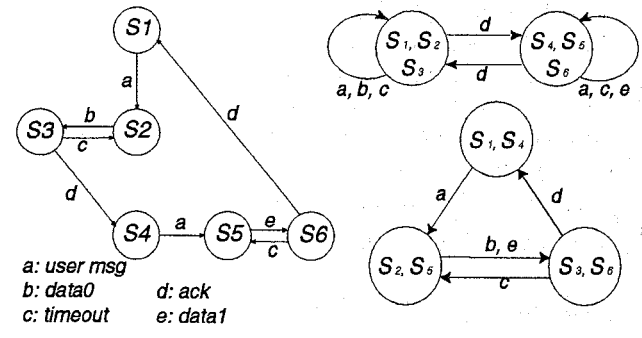

Figure 6. *Decomposition of the alternated bit protocol (ABP) into one 2-FSM and one 3-FSM.*

k-FSMs of various sizes allows the resolution of the non-determinism locally by mixing decompositions of k-FSMs ($k > 2$) (when non-determinism must be prevented) and decompositions of 2-FSMs (when non-determinism does not occur).

5 Error control codes for fault tolerant decompositions

In this section, we show how to use error detecting/correcting codes so that the decomposition remains robust against a breakdown of a subset of the observers. This characteristic of fault tolerance of our decomposition strategy should be contrasted to the behavior of the decomposition obtained by the perfect shuffle algorithm of Wang and Schwartz, which can only detect a reduced subset of faults when faced with the same problems.

[4] The size of an FSM designates its number of states.

5.1 The principle

The decomposition into 2-FSMs is interesting in that the system for error detection does not suddenly collapse but remains capable of detecting a subset of the faults as long as some 2-FSMs remain operational. In this section, we propose a decomposition strategy that is fault tolerant to the malfunctioning of k 2-FSMs. The goal here is to decompose an FSM A into n 2-FSMs such that any subset of 2-FSMs of size m be a decomposition of A. Two cases may arise:

1. A faulty 2-FSM *stops* contributing to the error detection process. In this case, the remaining 2-FSMs need only be a decomposition of the initial FSM. A code for error detection with a Hamming distance of $k+1$ and of size n is sufficient for this purpose[5]. Since a state in the machine A can be uniquely determined from any $(n-k)$ bits, it is thus possible to tolerate k faulty 2-FSMs.

2. A faulty 2-FSM *continues* to participate in the error detection process. The results of the faulty 2-FSM may be incorrect and may induce the global fault detection mechanism into error. To avoid such a situation, it becomes necessary to use an error correcting code of Hamming distance equal to $2k+1$. A state in the original FSM is determined uniquely with n bits even when k bits (out of n) are faulty [2, 9].

Whenever a faulty observer is again made operational, it needs to be resynchronized with the current state of the protocol. The mechanism proposed for this purpose by Wang and Schwartz in [16, 17] requires the availability of input classes of minimal length to allow a quick re-synchronization of the observer when it is brought back into operation. Another advantage of the use of a set of redundant observers is that the newly made operational observer need only get the current state of the protocol from those observers that are still functioning. Such a feature remains true as long as the number of faulty observers does not exceed the capacity of detection (or correction) of the error control code that is used.

5.2 Example

Consider the set of Q states of the example of Fig. 4. It is possible to use a parity code to generate three 2-FSMs such that any 2-FSMs be a decomposition of the original FSM. The states can be chosen as follows: $S_1 = 000$, $S_2 = 011$, $S_3 = 101$, and $S_4 = 110$. This yields the three 2-FSMs shown in Fig. 7:

$$Q_1^1 = \{S_1, S_2\} \quad Q_2^1 = \{S_3, S_4\}$$
$$Q_1^2 = \{S_1, S_3\} \quad Q_2^2 = \{S_2, S_4\}$$
$$Q_1^3 = \{S_1, S_4\} \quad Q_2^3 = \{S_2, S_3\}$$

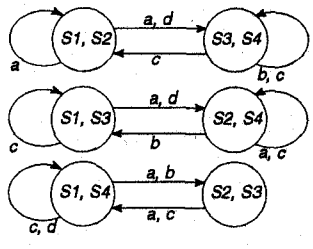

Figure 7. *An example of a fault tolerant decomposition.* In this case, when any 2-FSM becomes faulty, the two 2-FSMs left remain a decomposition of the original FSM.

It is difficult to build error control codes that are binary and optimal (every redundancy bit increases the Hamming distance by 1). Furthermore, Reed-Solomon codes [2, 9] are codes with symbols coded on several bits but that are easy to build for variable Hamming distances. The use of such codes enables us to tolerate several faulty observers. The disadvantage of these codes is that the symbol size requires the decomposition into FSMs of k states, with $k = 2^{(\text{size (symbol)})}$. For example, decomposing into 4-FSMs allows us to use a Reed-Solomon code that detects up to 3 errors or corrects 1 error, which allows us to tolerate the breakdown of up to 3 observers or one malicious observer.

5.3 Illustration: Transport protocol TP4

We restrain ourselves to a simplified version of the transport protocol TP4 to illustrate our decomposition strategy based on coding theory.

The original finite state machine of TP4 is shown in Fig. 8. In order to decompose this FSM, we arbitrarily code the states as follows:

Closed	=	000	*WFCC*	=	100
WFTRESP	=	001	*WFCC'*	=	101
AKWAIT	=	010	*Closing*	=	110
Open	=	011			

This coding yields the following decomposition, shown in Fig. 8:

$$Q_1^1 = \{Closed, WFTRESP, AKWAIT, Open\}$$
$$Q_2^1 = \{WFCC, WFCC', Closing\}$$

[5] In a code of size n and Hamming distance d, any two codewords differ in at least d bits.

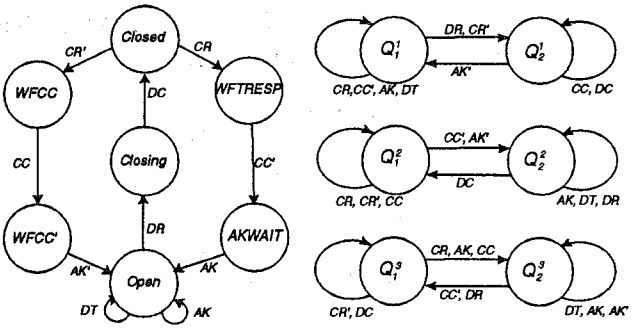

Figure 8. *Decomposition of TP4.*

$$
\begin{aligned}
Q_1^2 &= \{Closed, WFTRESP, WFCC, WFCC'\} \\
Q_2^2 &= \{AKWAIT, Open, Closing\} \\
Q_1^3 &= \{Closed, AKWAIT, WFCC, Closing\} \\
Q_2^3 &= \{WFTRESP, Open, WFCC'\}
\end{aligned}
$$

In order to be able to replace any one of the three 2-FSMs of TP4 or to detect that it is faulty, it is possible to add one additional 2-FSM by using a parity code. The states of TP4 are coded as follows:

Closed	=	0000	*WFCC*	=	1001
WFTRESP	=	0011	*WFCC'*	=	1010
AKWAIT	=	0101	*Closing*	=	1100
Open	=	0110			

The states of the 2-FSM added for tolerance are defined as follows:

$$
\begin{aligned}
Q_1^4 &= \{Closed, Open, WFCC', Closing\} \\
Q_2^4 &= \{WFTRESP, AKWAIT, WFCC\}
\end{aligned}
$$

The new 2-FSM is shown in Fig. 9.

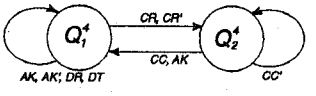

Figure 9. *A parity 2-FSM to replace any 2-FSM in the decomposition of TP4.*

6 Hints for resolving non-determinism

As mentioned earlier, an arbitrary coding of the states may yield more non-determinism in the sub-FSMs than if the codes for states were chosen in a well-thought manner.

In [14], Vijayananda discusses fault coverage of a given decomposition. In this section, we give two indications on how to code states such that the resulting decomposition exhibits a reduced 'amount' of non-determinism. First we define the source and destination sets as follows:

Definition 1 The *source set* E^- of an event E is the set of states that accept the event E:

$$E^- = \{S \in Q \mid \exists S' \in Q, \delta(S, E) = S'\}$$

Definition 2 The *destination set* E^+ of an event E is the set of states that can be reached with the event E:

$$E^+ = \{S \in Q \mid \exists S' \in Q, \delta(S', E) = S\}$$

One way to reduce the non-determinism and to increase the fault coverage of a decomposition is to encode the states of the initial FSM such that:

1. For every event E, the Hamming distance between two states in the source set E^- is minimized.

$$\forall E \in \Sigma; \{\text{Hamming-distance}(S_1, S_2) | S_1, S_2 \in E^-\} \text{ is minimal.} \quad (1)$$

2. For every event E, the Hamming distance between two states in the destination set E^+ is minimized.

$$\forall E \in \Sigma; \{\text{Hamming-distance}(S_1, S_2) | S_1, S_2 \in E^+\} \text{ is minimal.} \quad (2)$$

Intuitively, the condition expressed in Expression (1) increases the fault coverage of the decomposition; whereas the condition expressed in Expression (2) reduces potential non-determinism.

The states of the initial FSM are placed on the vertices of a hypercube of dimension $\lceil \log_2 |Q| \rceil$. When the states of the source set (respectively, destination set) of an event E are placed (coded) on the same edge, face, cube, etc. of the hypercube, the maximal Hamming distance between any two states in the source set (respectively, destination set) is reduced. The size of the source set (respectively, destination set) of E is thus reduced.

Consider the FSM corresponding to the ABP protocol shown in Fig. 6 (left). The source and destination sets of the events $\Sigma = \{a, b, c, d, e\}$ of this FSM are as follows:

$$
\begin{aligned}
a^- &= \{S_1, S_4\} & a^+ &= \{S_2, S_5\} \\
b^- &= \{S_2\} & b^+ &= \{S_3\} \\
c^- &= \{S_3, S_6\} & c^+ &= \{S_2, S_5\} \\
d^- &= \{S_3, S_6\} & d^+ &= \{S_1, S_4\} \\
e^- &= \{S_5\} & e^+ &= \{S_6\}
\end{aligned} \quad (3)
$$

The only events that have a source or destination set that is not a singleton are events a, c, and d.

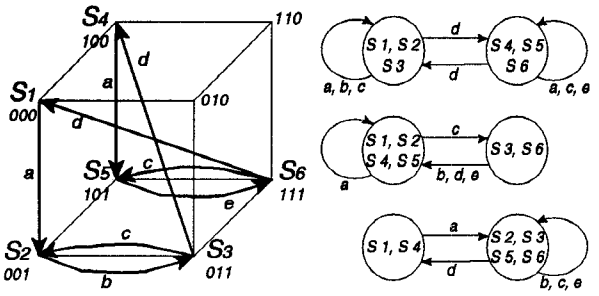

Figure 10. *Left:* The states of the initial FSM of ABP are placed on the vertices of a hypercube. The hard lines represent the transitions in the initial FSM. *Right:* Decomposition into deterministic sub-machines.

In Fig. 10 (left), the states of the ABP protocol are coded in a hypercube of dimension 3 (i.e., a cube). The states in the source set of each of the events a, c, and d are placed on the same edge in the cube. This is also true for the states in the destination set of these events. This assignment results in a deterministic full-coverage decomposition. This decomposition must be contrasted with the decomposition obtained by the perfect shuffle algorithm [16, 17], which does not cover all faults.

However, automating this decomposition technique is still an open problem and requires further investigations.

7 Conclusion

In this paper, we address the problem of detecting execution errors in communication protocols. We base our approach on the multiple observers strategy proposed by Wang and Schwartz [16]. We show that this problem can be studied from the perspective of coding theory. This allows us to generalize the technique of multiple observers into a fault tolerant one.

As a future research direction, we intend to develop an adequate encoding of the states in order to solve the problem of non-determinism of the sub-FSMs resulting from the decomposition.

References

[1] K. Barlett, R. Scantlebury, and W. Wilkinson. A Note on Reliable Full-duplex Transmission over Half-duplex Links. *Communications of the ACM*, 12(5):260–261, 1969.

[2] R. E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley, Reading, MA, 1983.

[3] Hermann Kopetz and Gunter Grunsteidl. TTP - A Protocol for Fault-Tolerant Real-Time Systems. *Computer*, 37:14–23, 1994.

[4] David Lee, Arun N. Netravali, and Krishan K. Sabnani. Protocol Pruning. *Proceedings of the IEEE*, 83:1357–1372, 1995.

[5] Régis Leveugle. *Analyse de Signature et Test en Ligne Integré sur Silicium.* PhD thesis, Institut National Polytechnique de Grenoble, January 1990.

[6] Guevara Noubir. *Nouvelles Techniques pour la Tolérance aux Pannes Basées sur l'Algèbre des Polynômes.* PhD thesis, Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland, 1996.

[7] Guevara Noubir, Kateel Vijayananda, and Henri J. Nussbaumer. A Robust Transport Protocol for Run-Time Fault Detection. In *Proceedings of ICNP'95: IEEE International Conference on Network Protocols*, pages 164–171, Tokyo, Japan, November 7-10, 1995.

[8] Guevara Noubir, Kateel Vijayananda, and Prasad Raja. Signature Based Technique for Fault Detection in Communication Protocols. In *Proceedings of the IEEE International Symposium on Information Theory*, page 43, Whistler, BC, Canada, September 17-22, 1995.

[9] Henri J. Nussbaumer. *Téléinformatique I.* Presses Polytechniques Romandes, 1987.

[10] Henri J. Nussbaumer. *Computer Communication Systems*, volume 1 & 2. Wiley Chichester, 1989-1990.

[11] Kostas N. Oikonomou. Abstractions of Finite-State Machines Optimal with Respect to Single Undetectable Output Faults. *IEEE Transactions on Computers*, C-36(2):185–200, 1987.

[12] Kostas N. Oikonomou. Abstractions of Finite-State Machines and Immediately-Detectable Output Faults. *IEEE Transactions on Computers*, 41(3):325–338, 1992.

[13] Marc Riese. *Model-Based Diagnosis of Communication Protocols.* PhD thesis, Swiss Federal Institute of Technology, Lausanne, Switzerland, 1993.

[14] Kateel Vijayananda. *A Framework for Diagnosis of Communication Protocols.* PhD thesis, Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland, 1996.

[15] Kateel Vijayananda. Distributed Fault Detection in Communication Protocols Using Extended Finite State Machine. In *ICPADS, International Conference on Parallel and Distributed System*, Tokyo, June 3-6, 1996.

[16] Clark Wang and Misha Schwartz. Fault Detection with Multiple Observers. In *IEEE INFOCOM'92, Florence*, pages 2187–2196, 1992.

[17] Clark Wang and Misha Schwartz. Fault Detection with Multiple Observers. *IEEE Transactions on Networking*, 1(1):48–55, 1993.