Open access • Journal Article • DOI:10.1109/71.491575

A note on consensus on dual failure modes — Source link

Hin-Sing Siu, Y. H. Chin, Wei-Pang Yang

Institutions: National Chiao Tung University, National Tsing Hua University

Published on: 01 Mar 1996 - IEEE Transactions on Parallel and Distributed Systems (IEEE Computer Society)

A Note on Consensus on Dual Failure Modes

Hin-Sing Siu, Yeh-Hao Chin, *Senior Member, IEEE*, and Wei-Pang Yang, *Senior Member, IEEE*

Abstract—Meyer and Pradhan proposed the *MS* (for "mixed-sum") algorithm to solve the Byzantine Agreement (BA) problem with *dual failure modes*: *arbitrary faults* (Byzantine faults) and *dormant faults* (essentially omission faults and timing faults) [3]. Our study indicates that this algorithm uses an inappropriate method to eliminate the effects of dormant faults and that the bound on the number of allowable faulty processors is overestimated. This paper corrects the algorithm and gives a new bound for the allowable faulty processors.

Index Terms—Byzantine Agreement, consensus problem, distributed systems, dual failure modes, fault tolerance, hybrid fault model.

1 Introduction

In practice, the processors of a distributed system may be subjected to different types of failure simultaneously. Examples of processor failure include *crash, omission, timing, incorrect computation*, or *arbitrary faults* (also called *malicious* or *Byzantine faults*). For such *multiple failure modes* (also referred to as the *hybrid fault model*), several algorithms have been proposed for solving the Byzantine Agreement (BA) problem [2], [3], [5]. The major limitation of these algorithms is that the number of *arbitrary* faults must be known prior to execution of the algorithm. However, this requirement violates the general assumption of the *BA* problem—that a fault-free processor cannot ascertain another processor's faulty status [1]. Moreover, Shin and Ramanathan [4] found that it is practically impossible to run diagnostics to detect all malicious faults in a network; so some malicious faulty processors will remain undetected even after a diagnostic. Also, Meyer and Pradhan indicated that this kind of algorithm is unable to reach an agreement between processors when the number of arbitrary faults is overestimated (or underestimated) [3].

To remove the above limitation, Meyer and Pradhan proposed the *MS* algorithm for the *BA* problem with dual failure modes [3]. The algorithm can tolerate any fault in a system provided $n > 3m + b$ and $c > 2m + b$, where n is the total number of processors, c is the system connectivity, m is the number of arbitrary faults, and b is the number of dormant faults. It is a recursive algorithm modified from the *OM* (oral message) algorithm in Lamport et al. [1]. The difference between the *OM* algorithm and the *MS* algorithm is that the latter has a specific method for handling dormant faults. In order to eliminate the effects of dormant faults, a fault-free processor selects a specific value, *val*, to replace the incoming message in the *last round* of the *MS* algorithm if it receives no message or a nonsensical message. The majority vote used in *OM* is replaced in the *MS* algorithm by a majority vote *maj* with all null messages eliminated.

The *MS* algorithm appears to be reliable and reasonable; however, we have discovered two problems with the algorithm. The first is that the method for handling dormant faults is inappropriate. The second is that the bound on the number of allowable faulty processors is overestimated. In Section 2, the first problem is discussed, and a revised approach for handling dormant faults is presented. Section 3 derives the correct bound on the number of allowable faulty processors in the *MS* algorithm.

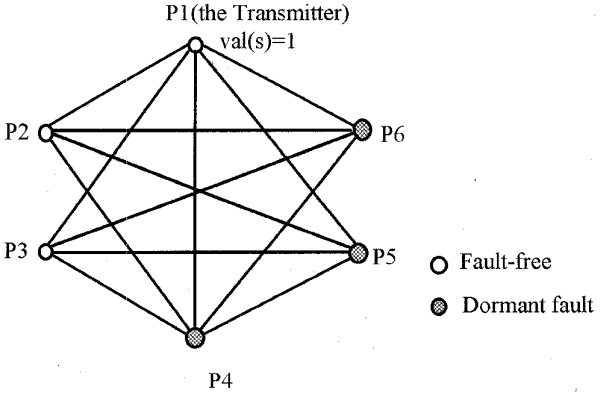


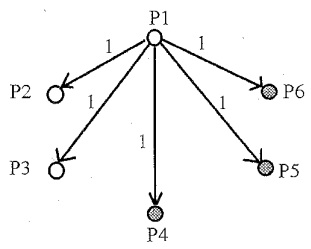
Fig. 1. The first counterexample to the *MS* algorithm.

2 The First Problem With the *MS* Algorithm

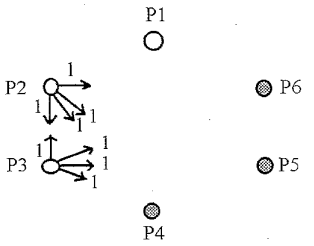
The following example shows that the method used by the *MS* algorithm to eliminate the effect of dormant faults is inappropriate. Fig. 1 shows a network with six processors, for which the system connectivity c is four. Suppose that processors P4, P5, and P6 are subjected to dormant faults. That is, suppose $n = 6$, $c = 4$, $m = 0$, and $b = 3$. According to the constraints on failures, namely $n > 3m + b$ and

• *H.-S. Siu and W.-P. Yang are with the Institute of Computer and Information Science, National Chiao-Tung University, Hsin-Chu, Taiwan 30050, Republic of China.*
• *Y.-H. Chin is with the Institute of Computer Science, National Tsing-Hua University, Hsin-Chu, Taiwan 30043, Republic of China. E-mail: yhchin@cs.nthu.edu.tw.*

Manuscript received Sept. 25, 1994; revised July 17, 1995.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number D95091.



(a) MS^1

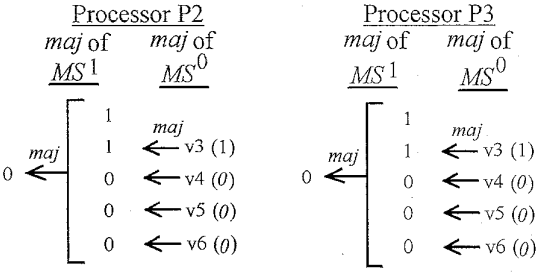


(b) MS^0

Processor P2	
MS^1	MS^0
v1 (1)	
	v3 (1)
	v4 (0)
	v5 (0)
	v6 (0)

Processor P3	
MS^1	MS^0
v1 (1)	
	v2 (1)
	v4 (0)
	v5 (0)
	v6 (0)

(c) The messages received at each fault-free processor



(d) After majority vote *maj*

Fig. 2. The step-by-step procedure of the *MS* algorithm.

$c > 2m + b$, the bound holds, because $6 > 3$ and $4 > 3$; however, the fault-free processors P1, P2, and P3 are unable to reach an agreement when the *MS* algorithm is applied.

Fig. 2 shows the step-by-step procedure of the *MS* algorithm. In the first round, denoted by MS^1, Transmitter P1 broadcasts its initial value "1" to all other processors, as shown in Fig. 2a. In second round MS^0 (the last round), every processor (excluding P1) broadcasts the message received from P1 to all four other processors (excluding P1 and itself), as shown in Fig. 2b. Fig. 2c shows the messages received at each fault-free processor after MS^0 (vi represents the message received from processor *i*). Assume that the dormant-faulty processors P4, P5, and P6 send no messages during the entire execution of the algorithm. In the *MS* algorithm, a specific value *val*, say "0," is selected by the receiving processor when no message is received from a sender during the last round (MS^0) of the algorithm. As shown in Fig. 2d, when the majority vote *maj* is applied to the received messages, the two fault-free processors, P2 and P3, are unable to agree on the common value "1" that was sent by Transmitter P1.

The cause of the result is that the *MS* algorithm does not accurately reflect the status of a dormant fault. In the last round of the algorithm, a fault-free processor p will select the specific value, *val*, to replace the missing message from the faulty processor q. In the way, *val* is semantically still counted as a valid message during the majority vote; q is treated as a *present voter* and its messages are counted as *present votes*. Consequently, the faulty processor q still affects the final result.

The correct principle for handling a "no message" situation is that *an absentee's vote should not be counted*. In the first round of the *MS* algorithm, the transmitter should broadcast its initial value *val*(*s*), which belongs to set of possible values, V, to all other $n-1$ processors; thus if no message is received from the transmitter, a fault-free processor can determine that the transmitter has failed. In order to satisfy the *agreement condition* of the *BA* problem [1], agreement should be reached by every fault-free processor even if no message was sent from the transmitter and p selects the default value, say 0, to replace the transmitter's message if no message was received from the transmitter.

Subsequently, the $n-1$ processors will execute $\lfloor (n-1)/3 \rfloor$ message exchange rounds to verify the message received from the transmitter [3]. In these message exchange rounds, each processor (excluding the transmitter) acts as the transmitter to broadcast its messages to other processors and *itself*. Since these $n-1$ processors exchange their messages with each other in every message exchange round, a fault-free processor p can detect that another processor q has failed if no message is received from q. If p received no message from q in the rth round, all messages received from q (directly) in the rth round and subsequent rounds of the algorithm are replaced by the value A, and this value will be relayed to the other processors as value RA_1. In each subsequent round, the value RA_j will be relayed to the other processors as value RA_{j+1} (A and $Ra_j \notin V$, where $1 \le j \le \lfloor (n-1)/3 \rfloor$.

Semantically, the value A is represented as an *absentee vote*, and processor q is treated as an *absentee*; hence, q's ticket is ignored during the majority vote. The value RA_j will be interpreted as *the j*th *time an absent vote is reported*. Processor p will report to all the other processors that q is an absentee, and the faulty processor q will be forced out of the game; thus, q has no influence on the others when

the majority vote is taken. Our approach can be formalized as follows.

Absent rule: When processor p receives no message directly from another processor q in the MS^k algorithm,

1) Processor p selects the *default value* to replace the incoming message from q (the transmitter) if $k = \lfloor (n-1)/3 \rfloor$ (the first round); or
2) All messages received from q in MS^k and subsequently rounds (if any) are replaced by A, and the value RA_1 is relayed to all the other processors; the value RA_j will be relayed to the other processors as value RA_{j+1} $(1 \le j \le \lfloor (n-1)/3 \rfloor)$ if $0 \le k < \lfloor (n-1)/3 \rfloor$.

After the messages are exchanged, the new recursive majority vote, *n-maj*, used in our approach counts only the non-A values. The majority value returned from the MS^l algorithm depends on the following four conditions:

c1: The original value of MS^l, if $l = 0$.
c2: The default value, if the majority value doesn't exist.
c3: v, if the majority value of MS^l is v and $v \neq RA_j$ where $1 \le j \le \lfloor (n-1)/3 \rfloor$.
c4: A, if the majority value of MS^l is RA_1.
c5: RA_{j-1}, if the majority value of MS^l is RA_j where $1 < j \le \lfloor (n-1)/3 \rfloor$.
c6: The original value of MS^l, if $l > 0$ and the majority value is Null.

Note that the conditions, *c1*, *c2*, and *c3* (excluding the condition $v \neq RA_j$), are similar to Meyer and Pradhan's majority vote, *maj*. The additional conditions, *c4*, *c5*, and *c6*, are used to correctly handle dormant faults. Semantically, conditions *c4* and *c5* are used to report the existence of an absentee. When a majority of processors report that an absentee exists, *n-maj* returns the value A or RA_{j-1} to represent the event. On the other hand, when the majority value is Null (i.e., the values returned from all MS^{l-1} are A), it means that all other processors are absentees; therefore, the votes from these absentees shall be ignored, and the original value of MS^l shall be used as the majority value.

When this approach is used, as shown in Fig. 3, every fault-free processor in the network shown in Fig. 1 agrees on the common value "1" that was sent by Transmitter P1.

3 The Second Problem With the *MS* Algorithm

The second problem with Meyer and Pradhan's results [3] is that the bound on the number of allowable faulty processors is incorrect. The constraint on connectivity, $c > 2m + b$, is indeed a necessary condition for reaching an agreement under dual failure modes (i.e., the constraint is correct); however, the constraint on the number of processors required, $n > 3m + b$, is incorrect. The following

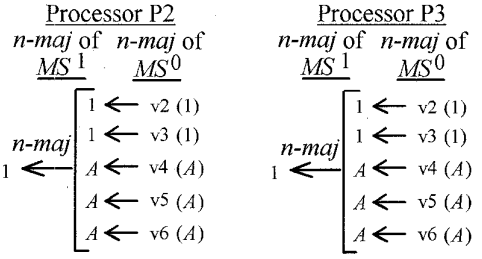


Fig. 3. Removing the effects of dormant faults in Fig. 1.

example shows that this constraint is incorrect. Fig. 4a shows a fully connected network with seven processors, for which the system connectivity c is six. Suppose that the processors P4, P5, and P6 are subjected to dormant faults and the processor P7 is subjected to an arbitrary fault. That is, suppose $n = 7$, $c = 6$, $m = 1$, and $b = 3$. According to the constraints on failures, namely $n > 3m + b$ and $c > 2m + b$, the bound holds, because $7 > 3 + 3$ and $6 > 2 + 3$; however, the fault-free processors P2 and P3 are unable to reach an agreement with respect to Transmitter P1's initial value "1" when the *MS* algorithm is applied, as shown in Fig. 4b.

The inaccurate results can be explained as follows. In the first round of the *MS* algorithm, denoted by MS^l $(l = \lfloor (n-1)/3 \rfloor)$, the transmitter, says P1, broadcasts its initial value to all other $n-1$ processors, each of which then executes $n-1$ separate execution of the MS^{l-1} algorithm (the second round) to exchange the message received from the transmitter in the first round. In MS^{l-1}, every processor (excluding the transmitter) invokes $n-2$ separate execution of the MS^{l-2} algorithm to exchange the message received in the second round. This recursive procedure will be executed $\lfloor (n-1)/3 \rfloor + 1$ times, and then the majority vote *maj* is applied to compute the common agreement. The MS^i algorithm, $i > 0$, creates a vector of $n - \lfloor (n-1)/3 \rfloor + i - 1$ values, where each vector value is the output of the majority vote from the MS^{i-1} algorithm, and each MS^0 outputs one message. When a majority vote is used to compute the final common value for agreement, the number of messages collected in the MS^i algorithm must be greater than $2m + b$, namely $n > \lfloor (n-1)/3 \rfloor - i + 1 + 2m + b$. Because the number of messages collected in MS^1 is the least, MS^1 can compute a correct value from MS^0, and each MS^i can also compute a correct value from MS^{i-1}, $i > 0$. Therefore, the correct constraint on failure in the *MS* algorithm for dual failure modes is $n > \lfloor (n-1)/3 \rfloor + 2m + b$. Meyer and Pradhan claimed that the number of processors required to tolerate $m + b$ faults is $n > 3m + b$ [3]. However, m should not be greater than $\lfloor (n-1)/3 \rfloor$, namely $m \le \lfloor (n-1)/3 \rfloor$. This implies that $3m + b \le \lfloor (n-1)/3 \rfloor + 2m + b$; therefore, the bound on the number of faulty processors allowed in the algorithm is too high. Table 1 gives an example to compare the number of faulty processors allowed under these two bounds.

TABLE 1
THE EXAMPLE OF THE NUMBER OF FAULTY PROCESSORS ALLOWED UNDER THE ORIGINAL BOUND AND THE CORRECT BOUND

	Original bound $n > 3m + b$		Correct bound $n > \lfloor (n-1)/3 \rfloor + 2m + b$	
n	m	b	m	b
7	1	3*	1	2
8	2	1	2	1
	1	4*	1	3
9	2	2	2	2
	1	5*	1	4
10	2	3*	2	2
	1	6*	1	4
11	3	1	3	1
	2	4*	2	3
	1	7*	1	6

* *Exceeds the tolerable bound*

To prove the correctness of the new constraint on failures, we need to prove that the modified-*MS* algorithm can meet the requirements of *validity* and *agreement* under the new bound.

LEMMA 1. *The modified-MS (MMS) algorithm satisfies the validity condition if* $n > \lfloor (n-1)/3 \rfloor + 2m + b$.

PROOF. The proof is by induction on l. Note that the validity condition applies only when the transmitter is fault-free. Since the transmitter is fault-free, all other fault-free processors receive the transmitter's value of MMS^0, so the lemma is true when $l = 0$. Assuming that the Lemma is true for $l - 1$, $l > 0$, we show that it is true for l.

The fault-free transmitter of MMS^l sends a value $val(s)$ to all $n-1$ other processors and each of the $n-1$ processors executes the MMS^{l-1} algorithm. By hypothesis $n > \lfloor (n-1)/3 \rfloor + 2m + b$, we have $(n-1) > (\lfloor (n-1)/3 \rfloor - 1) + 2m + b$. Thus, by the induction hypothesis, we conclude that every fault-free processor gets $val_p(s) = val(s)$ for each fault-free processor p.

Since each of the $n-1$ processors acts as the transmitter in MMS^{l-1}, each of these processors has a vector of $n-1$ elements. Each element in this vector can take on one of two possible values: *invalid* (value A) and *valid* (non-A). Let k denote the total number of invalid values and v_l the total number of valid values in this vector. Then $v_l = n - 1 - k$. We can write $k = \alpha + \beta$, where α is the number of invalid votes with respect to the dormant faults, $\alpha \geq 0$, and β is the number of invalid votes with respect to the arbitrary faults, $\beta \geq 0$. Therefore, we can write $b = b' + \alpha$,

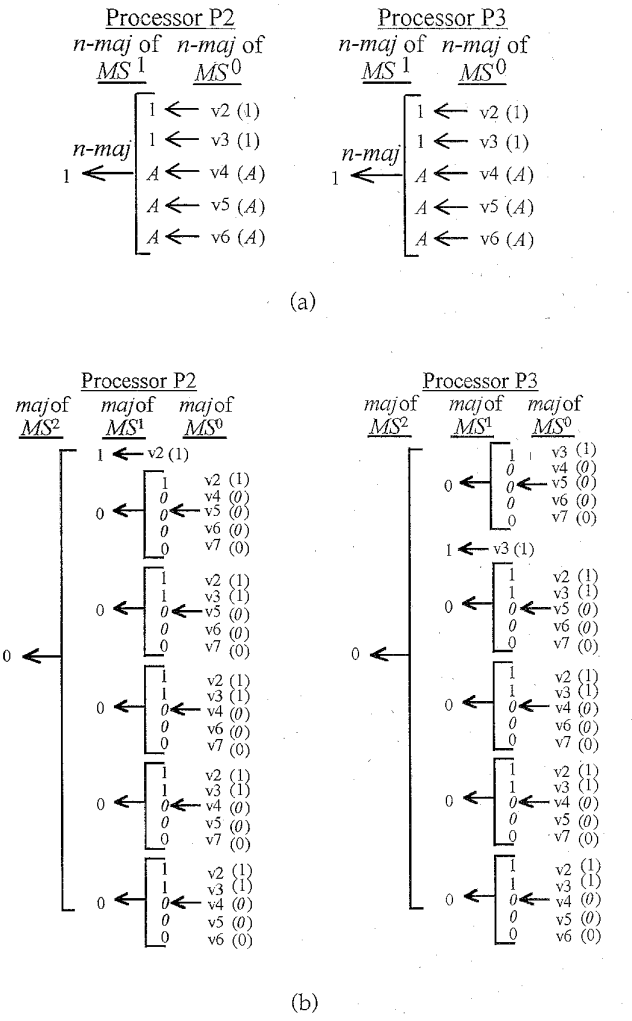


Fig. 4. The second counterexample to the *MS* algorithm: a) the network with seven processors; b) after majority vote *maj*.

where $b' \geq 0$ and $m = m' + \beta$, where $m' \geq 0$. Note that the set of b' elements (each element is correct) results from crash faults or omission faults (these faulty links always send correct messages in the protocol). To compute the correct value from these valid messages by using a majority vote, the number of valid messages must be greater than $2m' + b'$; namely $v_l > 2m' + b'$. By hypothesis,

$$n > \lfloor (n-1)/3 \rfloor + 2m + b,$$

so that

$$v_l > \lfloor (n-1)/3 \rfloor + 2m + b - 1 - k,$$

and we have

$$v_l > \lfloor (n-1)/3 \rfloor + 2m + b - 1 - (\alpha + \beta),$$

$$> \lfloor (n-1)/3 \rfloor + 2(m' + \beta) + (b' + \alpha) - 1 - (\alpha + \beta),$$

$$> \lfloor (n-1)/3 \rfloor + 2m' + \beta + b' - 1.$$

Since $\beta \geq 0$ and $\lfloor (n-1)/3 \rfloor > 0$, we can write $v_l > 2m' + b'$. Therefore, a majority of the v_l values are correct. Thus, each fault-free processor selects *val*(s) as the common value by the majority vote of the algorithm, which satisfies the validity condition. □

The following proves that the modified-*MS* algorithm satisfies the agreement condition. If the transmitter is fault-free, Lemma 1 implies the agreement condition. We only have to prove that the Lemma is true when the transmitter is faulty. If the transmitter is subjected to a dormant fault, all fault-free processors receive the same value (either the default value or *val*(s)). By Lemma 1, each fault-free processor receives the same values for all fault-free processors in the MMS^{l-1} algorithm. Hence, the MMS^{l} algorithm satisfies the agreement condition if the transmitter's faulty is dormant. When the faulty in a transmitter is arbitrary. There are at most m processors with arbitrary faults, and the transmitter is one of them, so at most $m-1$ other processors have arbitrary faults. By hypothesis,

$$n > \lfloor (n-1)/3 \rfloor + 2m + b,$$

we have,

$$\begin{aligned}(n-1) &> (\lfloor (n-1)/3 \rfloor - 1) + 2m + b \\ &> (\lfloor (n-1)/3 \rfloor - 1) + 2(m-1) + b.\end{aligned}$$

Consequently, each fault-free processor receives the same values. Thus, each fault-free processor selects the common value that is the majority value of v_l valid values $(v_l = n - 1 - k)$. Hence, the agreement condition is satisfied. Since the modified-*MS* algorithm works the same way as the *OMH* algorithm [2] does, the correctness of the following lemma can be proven through the above discussion and the arguments of [2].

LEMMA 2. *The modified-MS algorithm satisfies the agreement condition if* $n > \lfloor (n-1)/3 \rfloor + 2m + b$.

THEOREM 1. *The modified-*MS^l $(l = \lfloor (n-1)/3 \rfloor)$ *algorithm solves the BA problem with dual failure modes if* $n > \lfloor (n-1)/3 \rfloor + 2m + b$.

PROOF. By Lemma 1 and Lemma 2, both conditions are necessary for the *BA* problem; thus, the theorem is proven. □

The constraint on failures, $n > \lfloor (n-1)/3 \rfloor + 2m + b$, not only presents a correct bound for the modified *MS* algorithm, but also can be extended for any algorithm based on the "oral messages" approach [1]. Hence it is a general constraint on failures for the *BA* problem with dual failure modes. When only arbitrary faults are presented in a network, the constraint on failures by modified-*MS* algorithm is $n > 3m$; and the algorithm provides the same fault-masking ability as the traditional oral messages algorithm has. On the other hand, the constraint on failures by the modified-*MS* algorithm is $n > b$ if only dormant faults are considered. The original bound, namely $n > \lfloor (n-1) / 3 \rfloor + b$, is enlarged to $n > b$ for the following reason. According to the absent rule stated in Section 2, a dormant fault can drop at most one message. Therefore, the influence of dormant faults can be removed by using the voting function, *n-maj*, if $n - b > 0$, namely $n > b$. Hence, the maximum number of tolerable dormant faults is $n - 1$, and the bounds for these two types of failure can be stated below.

COROLLARY 1. *When the network has processors with arbitrary faults only* $(b = 0)$, *the modified-MS algorithm solves the BA problem if* $n > 3m$; *if the network has processors with dormant faults only* $(m = 0)$, *the modified-MS algorithm solves the BA problem if* $n > b$.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous referees for their critical reading of the original manuscript and their valuable suggestions that improved the quality and readability of this paper.

This work was supported in part by the National Science Council of the Republic of China under Grant No. NSC84-2213-E-007-004.

REFERENCES

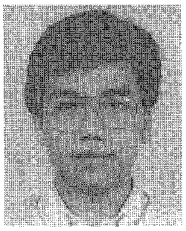
[1] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Programming, Languages, Systems*, vol. 4, no. 3, pp. 382-401, July 1982.

[2] P. Lincoln and J. Rushby, "A Formally Verified Algorithm for Interactive Consistency Under a Hybrid Fault Model," *IEEE Proc. Symp. Fault-Tolerate Computing*, pp. 402-411, 1993.

[3] F.J. Meyer and D.K. Pradhan, "Consensus With Dual Failure Modes," *IEEE Trans. Parallel and Distributed Systems*, vol. 2, no. 2, pp. 214-222, Apr. 1991.

[4] K. Shin and P. Ramanathan, "Diagnosis of Processors With Byzantine Faults in a Distributed Computing System," *IEEE Proc. Symp. Fault-Tolerate Computing*, pp. 55-60, 1987.

[5] P. Thambidurai and Y.-K. Park, "Interactive Consistency With Multiple Failure Modes," *IEEE Proc. Symp. Reliable Distributed Systems*, pp. 93-100, 1988.



Hin-Sing Siu received the BM degree from the Fu-Jen University, and the MS degree in computer and information science from the National Chiao-Tung University, Hsin-Chu, Taiwan, where he is now a PhD candidate in computer and information sciences.

His research interests include fault-tolerant distributed systems, computer networking, and database management systems.

Yeh-Hao Chin (S'69-M'72-SM'95) received the BSEE degree from the National Taiwan University, and the MS and PhD degrees in electrical engineering from the University of Texas at Austin in 1970 and 1972, respectively.

Dr. Chin has been a member of the faculty at Northwestern University, Cleveland State University, and the National Chiao-Tung University. He has also worked for the Control Data Corporation, Sunnyvale, California, and AT&T Bell Laboratories, Holmdel, New Jersey. Currently, he is a professor with the Institute of Computer Sciences, National Tsing-Hua University, Hsin-Chu, Taiwan.

His research interests include database management systems, operation systems, design and analysis of algorithms, software engineering, and VLSI design. He is a senior member of the IEEE Computer Society.

Wei-Pang Yang received the BS degree in mathematics from the National Taiwan Normal University in 1974, and MS and PhD degrees in computer engineering from the National Chaio-Tung University, Hsiu-Chu, Taiwan, in 1979 and 1984, respectively.

Since August 1979, he has been on the faculty of the Department of Computer Science and Information Engineering at the National Chiao-Tung University. In the 1985-1986 academic year, he was awarded the National Postdoctoral Research Fellowship and was a visiting scholar at Harvard University. In August 1988, he joined the Department of Computer and Information Science at the National Chiao-Tung University and acted as head of the department for one year. Next, he joined the IBM Almaden Research Center in San Jose, California, for a year as a visiting scientist. From 1990 to 1992, he again served as head of the Department of Computer and Information Science at the National Chiao-Tung University. His research interests include database theory, database security, object-oriented databases, image databases, and Chinese database-retrieval systems.

Dr. Yang is a senior member of the IEEE and a member of ACM. He won the 1988 and 1992 Acer Term Award for Outstanding MS Thesis Supervision, the 1993 Acer Long Term Award for Outstanding PhD Dissertation Supervision, and the 1990 Outstanding Paper Award of the Computer Society of the Republic of China. He also obtained the 1991-1995 Outstanding Research Award of the National Science Council of the Republic of China.