行政院國家科學委員會專題研究計畫　成果報告

# 一致性問題在多重錯誤同步分散式計算環境下的最佳解

計畫類別：個別型計畫
計畫編號：NSC91-2213-E-009-081-
執行期間：91 年 08 月 01 日至 92 年 07 月 31 日
執行單位：國立交通大學資訊工程學系

計畫主持人：黃廷祿

計畫參與人員：陳勝雄

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 92 年 10 月 30 日

# 行政院國家科學委員會專題研究計畫成果報告

## 一致性問題在多重錯誤同步分散式計算環境下的最佳解

## A Tight Bound of Consensus Problem in Hybrid Fault Synchronous Distributed Computing

計畫編號：NSC 91-2213-E-009-081

執行期限：91 年 8 月 1 日至 92 年 7 月 31 日

主持人：黃廷祿　　國立交通大學資訊工程學系

E-mail: tlhuang@csie.nctu.edu.tw

計畫參與人員：陳勝雄　　國立交通大學資訊工程學系

E-mail: chenss@csie.nctu.edu.tw

## 一、中文摘要

一致性問題是分散式系統中容錯處理必須面臨的重要課題。 Siu 等人於 1998 年提出一致性問題在多重錯誤模式下容錯能力的必要條件，並且提出一個演算法嘗試證明此條件亦為充分條件。然而，我們提出一個反例來說明他們的演算法違反了一致性問題的基本要求。因此，一致性問題容錯能力的最佳解是否存在仍然未定。根據 Garay 與 Perry 的演算法加上 Siu 等人在其演算法中所用的溝通機制（FTVC），我們提出一個演算法證明 Siu 等人所提的必要條件亦為充分條件，亦即此演算法為一致性問題容錯能力的最佳解。

**關鍵詞**： 一致性問題、容錯系統、分散式系統

## Abstract

Consensus problem is one of the most important issues in literature of fault tolerant distributed computing. Siu et al. in 1998 proposed a necessary condition of failures in the hybrid fault model, and an algorithm called GPBA to show that this condition is also sufficient. However, we present a counterexample to show that the algorithm violates the agreement condition, the most important safety property that was claimed. The necessary and sufficient condition of failures should be re-examined. Based on Garay and Perry's algorithm in addition to the reliable communication protocol, FTVC, used in GPBA, we propose an algorithm showing that the necessary condition of Siu et al. is indeed also sufficient.

**Keywords**: Consensus Problem, Fault-tolerant Systems, Distributed Systems

## 二、緣由與目的

Achieving consensus is a key problem in distributed computing systems tolerant of failures. Many algorithms have been proposed in different failure models [1-8]. Each algorithm attempted to maximize the failure resilience.

Siu, Chin and Yang [1] in 1998 proposed a byzantine agreement algorithm, *generalized protocol for the BA problem* (GPBA), in a general network whose topology is connected but may not be fully connected. In such a network, each processor can be subject to either *arbitrary fault* or *dormant fault*, so can each link. An arbitrary fault can exhibit arbitrary behavior, also known as byzantine fault; while a dormant fault consists merely of omission of messages or delay in sending or relaying messages. We follow the convention that a dormant processor cannot stop receiving messages since message reception is an action regarded as not locally controllable.

Let $n$ be the number of processors, $V$ the set of all possible values, $c$ the system connectivity, $Pa$ ($Pd$) the number of processors subject to the arbitrary (dormant) fault, and *La* (*Ld*) the number of links subject to the arbitrary (dormant) fault. According to Theorem 5 of Siu et al. [1], if failures are constrained by conditions (i) $n > Pa + Pd$, and (ii) $c > 2Pa + Pd + 2(La + Ld)$, then GPBA satisfies the following correctness conditions, using $\lfloor (n-1)/3 \rfloor +1$ rounds.

**Agreement**: All fault-free processors agree on the same common value *v*.

**Validity**: If the source is fault-free, then the common value *v* should be the initial value $v_s$ of the source.

However, we discovered a counterexample that satisfies the constraints on failures of GPBA but violates the agreement condition. Siu et al. [1] have shown that the conditions of failures $n > Pa + Pd$ and $c > 2Pa + Pd + 2(La + Ld)$ are necessary. Unfortunately, their algorithm is wrong and therefore the tight bound of failure resilience is still unknown. In this project, we aim to determine the tight bound.

## 三、結果與討論

First, we propose a counterexample for GPBA, showing that the tight bound of failure resilience should be re-examined. Then, we show that there exists an algorithm satisfying the conditions of failures proposed by Siu et al. Thus, the necessary conditions of failures are also sufficient, determining the optimal resilience of the hybrid fault model.

### A Counterexample for GPBA

A network and an execution of GPBA are presented in this section to show that the algorithm violates the agreement condition.

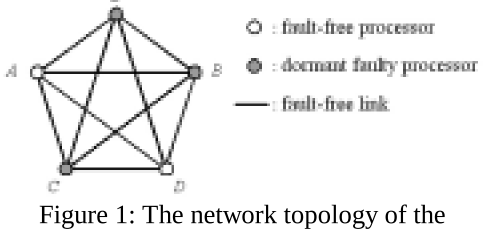

Figure 1: The network topology of the counterexample.

Fig. 1 shows the network of this counterexample with five processors, for which the connectivity is four. Let *S* be the source and $v_s$ its initial value. The value set $V=\{0,1\}$ and $v_s = 1$. Also, suppose that processors *S*, *B* and *C* are subject to dormant fault, and all the other components are fault-free. Finally, we assume *default value* is equal to 0. Processors will select the *default value* as the initial value of the source when receiving no message broadcast by the source. That is, suppose $n=5$, $c=4$, $v_s=1$, $Pa=0$, $Pd=3$, $La=0$ and $Ld=0$. It is clear that the constraints on failures are satisfied. Therefore, applying GPBA on this case, an agreement should be achieved by $\lfloor (n-1)/3 \rfloor +1 = 2$ rounds.

During the execution of GPBA on the above network, each fault-free processor maintains a tree structure having 2 levels to collect the messages. After the first round, each fault-free receiver stores the message received from the source, denoted as *val(S)*, at the root *S* of this tree. In the second round, each processor broadcasts the root's value to all receivers except *S*. If sender *B* sends a message *val(S)* to receiver *i*, *i* will store the message received from *B*, denoted as *val(SB)*, at vertex *SB* of its tree. The value of vertex *SB* in processor *i*'s tree is meant to be the initial value of *S* that *B* has conveyed to *i*; thus, the vertex *SB* is said to *correspond* to *B*. Value ∅ for a vertex indicates that a processor receives no message from the processor to which the vertex corresponds. It will be replaced by the default value in the first round and by value **A** in the second round after applying the absent rule in each round. **A** will be ignored in the VOTE function.

Based on the network in Fig. 1, Fig. 4 shows an execution of GPBA in which fault-free processors *A* and *D* are unable to decide the same value, and exhibits the states of all non-faulty processors after each round. In the first round, source *S* should use FTVC protocol to broadcast its initial value “1” to all other processors, but *S* sent its initial value only to *A*, *B* and *C* and then became dormant, omitting the sending to *D*. Since processor *D* didn't receive the message from source *S*, it selected the default value, 0, as the initial value of *S*, as shown in Fig. 4(a). In the second round, all processors (except *S*) should exchange the message received from *S*. Assume processors *B* and *C* suffered dormant faults in this round. After sending the message to *A* using FTVC protocol, processor *B* stopped sending and stopped relaying messages. Similarly, *C* stopped after sending the message to *A*. Fig. 4(b) shows the messages received at each non-faulty processor after the second round. Finally, after applying the function VOTE onto the received messages, processors *A* and *D* (the remaining fault-free processors) decided on 1 and 0, respectively, as shown in Fig. 4(c). This is a disagreement error.

## A resilience-optimal Algorithm

The resilience-optimal algorithm is based on the Frangible Consensus Protocol proposed by Garay and Perry [2]. The original protocol is designed for a failure model in which all links are reliable and the underlying network is complete. In the hybrid failure model, each link can be subject to either arbitrary fault or dormant fault. If we can find out a protocol making the underlying network as if a complete reliable network, the Frangible Consensus Protocol works on the network.

The *fault-tolerance virtual channel* (FTVC) protocol proposed by Siu et al. [1] can be used to provide a reliable communication between any two processors if the connectivity of the network satisfies the condition $c > 2Pa + Pd + 2(La + Ld)$. We combine the FTVC protocol and the Frangible Consensus Protocol to obtain the resilience-optimal algorithm. The resulting algorithm is correct if failures are constrained by conditions (i) $n > Pa + Pd$, and (ii) $c > 2Pa + Pd + 2(La + Ld)$. As a result, the conditions of failures are tight.

The resilient-optimal algorithm is briefly described as follows. First, the **MakeUnique** protocol ensures that all processors that accept a value accept the same value, but some processors may not accept any value (indicating by "accepting" the value 2).

```
MakeUnique($v$)
    FTVC($v$) to all processors;
    $C[0]$ := number of 0's received;
    $C[1]$ := number of 1's received;
    if      $C[0] \geq n-(Pa+Pd) \wedge C[1] \leq Pa$
            then $v$:=0
    elseif  $C[1] \geq n-(Pa+Pd) \wedge C[0] \leq Pa$
            then $v$:=1
    else    $v$ :=2
    fi;
```

Figure 2**: MakeUnique** Protocol.

Lemma 1 (Garay and Perry [2]) *If p and q are nonfaulty processors such that, in* **MakeUnique**, *p assigns* $r \neq 2$ *to* $v_p$ *and q assigns* $s \neq 2$ *to* $v_q$, *then* $r = s$.

```
$v$ := "initial value";
for $K$ := 1 to $(Pa+Pd) + 1$ do
    /* Universal Exchange 1 */
    MakeUnique($v$);

    /* Universal Exchange 2 */
    FTVC($v$) to all processors;
    $D[0]$ := number of 0's received;
    $D[1]$ := number of 1's received;
    $D[2]$ := number of 2's received;
    if      $D[0] \geq Pa$ then $v$: = 0
    elseif  $D[1] \geq Pa$ then $v$: = 1
    fi;

    /* King's Broadcast*/
    if $i = K$ then FTVC($v$) to all processors fi;
    $W$ := value received from processor $K$;
    if   ($v = 2 \vee D[v] \leq Pa \vee D[2] > Pa$ )
         then $v$ := min(1,$W$)
    fi;
od;
"final value" := $v$
```

Figure 3: The optimal algorithm for each processor *i*.

Lemma 2 (Garay and Perry [2]) *At the end of phase* $(Pa + Pd) + 1$ *for any nonfaulty processors p and q*, $v_p = v_q$.

Fig. 3 shows the resilient-optimal algorithm. It follows the Phase King paradigm of [3] in which the computation proceeds in phases, each of which has a processor designated as the phase king. Each phase $K$ of the resilient-optimal algorithm consists of 3 rounds of communication. In the first round, each processor executes the **MakeUnique** protocol. Then, each processor communicates with one another by the FTVC protocol. In the third round, only the phase king sends messages to all processors. According to Lemma 2, the resilient-optimal algorithm satisfies the agreement condition.

We assume that our model is synchronous round-based message-passing model and assume that a processor can communicate with each processor by using FTVC once during a period of one round. The resilient-optimal algorithm consists of $(Pa+Pd)+1$ phases, and each phase consists of 3 rounds of communication. The time complexity of the algorithm is $3(Pa+Pd+1)$ rounds.

## 四、計畫成果自評

本計畫首先指出 Siu 等人所提出具有最大容錯能力的一致性演算法有錯誤。這也指出一致性問題的容錯能力極限必須再重新探討。此部分成果已經投稿至 IEEE Transactions on Parallel and Distributed Systems.

其次，我們改進 Garay 與 Perry 所提出的演算法確認 Siu 等人所提容錯能力的必要條件的確也是充分條件。因此，Siu 等人追求最大容錯能力的方向是非常正確的。然而，此演算法主要依循 Garay 與 Perry 所提出的演算法再加上 Siu 等人所提的溝通機制（FTVC protocol），因此尚不具太多學術價值。可以再突破的地方為目前尚無關於多重錯誤模式下時間複雜度的下限。

## 五、參考文獻

[1] H.-S Siu, Y.-H. Chin and W.-P. Yang, “Byzantine Agreement in the Presence of Mixed Faults on Processors and Links,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 9, no. 4, pp. 335-345, Apr. 1998.

[2] J.A. Garay and K.J. Perry, “A Continuum of Failure Models for Distributed Computing,” In *Proceedings of the 6th International Workshop on Distributed Algorithms*, volume 647 of Lecture Notes in Computer Science, pp. 153-165, Haifa, Israel, November 1992.

[3] P. Berman, J.A. Garay and K.J. Perry, “Towards Optimal Distributed Consensus,” In *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*, pp. 410-415, 1989.

[4] F.J. Meyer and D.K. Pradhan, “Consensus with Dual Failure Modes,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 2, no. 2, pp. 214-222, Apr. 1991.

[5] H.-S. Siu, Y.-H. Chin and W.-P. Yang, “A Note on Consensus on Dual Failure Modes,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 7, no. 3, pp. 225-230, Mar. 1996.

[6] K.-Q. Yan, Y.-H. Chin and S.-C. Wang, “Optimal Agreement Protocol in Malicious Faulty Processors and Faulty Links,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 4, no. 3, pp. 266-280, June 1992.

[7] P. Lincoln and J. Rushby, “A Formally Verified Algorithm for Interactive Consistency Under a Hybrid Fault Model,” In *Proceedings of the Symposium on Fault-tolerant Computing*, pp. 402-411, 1993.

[8] P. Thambidurai and Y.K. Park, “Interactive Consistency with Multiple Failure Modes,” In *Proceedings of the Symposium on Reliable Distributed Systems*, pp. 93-100, Oct. 1988.

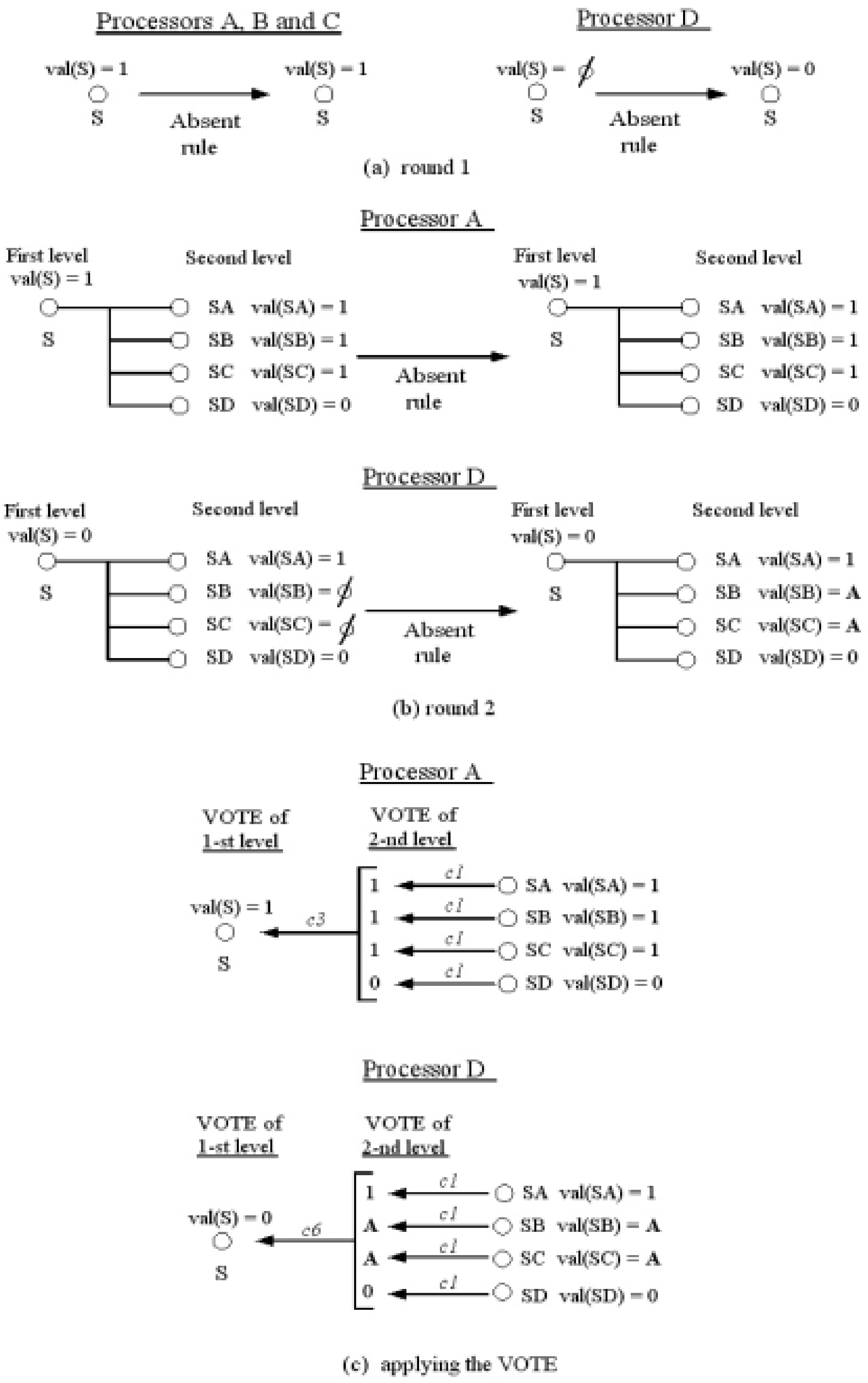

Figure 4: The states of nonfaulty processors after each round.