Towards Malicious Agreement in a Scale Free Network

S.C. Wang scwang@cyut.edu.tw
S. C. Yang s9430619@cyut.edu.tw
K.Q. Yan kqyan@cyut.edu.tw
Chaoyang University of Technology, Taiwan, R.O.C

Abstract

The fault-tolerance is an important research topic in the study of distributed systems. To cope with the influence from faulty processors, reaching a common agreement at the presence of faults before performing some special tasks is essential. Therefore, the Byzantine Agreement (BA) problem has drawn more and more of the researchers' attention as they explore deeper and deeper into the world of distributed systems. There are significant studies on this BA problem in a random network. Recently, many large complex networks have emerged and displayed a scale-free feature, which influences the system to reach a common value differently. Unfortunately, existing BA problem and results cannot cope with the new network environment and the BA problem thus needs to be revisited. In this paper, we propose a new protocol to adapt to the scale free network and derive its bound of allowable faulty components.

Keywords: Byzantine agreement, fault-tolerance, distributed system, scale free network, random network

1. Introduction

With the fast development of Internet, in order to increase systematic operation ability, the distributed system has replaced the traditional large-scale computer system gradually. In a distributed computing system, processors allocated in different places and connected together to create greater power and ability [10]. It is an important topic for represent the structure of highly fault-tolerant ability in the Scale Free Network (SFN) [17] seems to provide a new topic. SFN is universal in real world and each processor can infinite link other processors. [4].

However, under a large amount of computation resource requirement, the single processor has not been enough for the requirement. The distributed system is mostly used in order to increase systematic operation ability, the processors that disperses to every place is considered as a virtual group, the distributed system is the communication through different ways each other and exchanges information [7]. To achieve a common agreement when the distributed system is applied and to confirm each other reference information of need when transmits the file each other. It is necessity to develop a highly fault-tolerant protocol of improve system security and dependability.

The Byzantine Agreement (BA) problem is one of the most fundamental problems to reach a common value in a distributed system. The Byzantine Agreement problem was first proposed by Lamport [14]. Traditionally, BA problem defined by Lamport assumes [14]:

(1) There are n processors, of which at most t_p ($t_p \leq \lfloor (n-1)/3 \rfloor$) processors could fail without breaking down a workable network;

(2) The processors communicate with each other through message exchange in a fully connected network;

(3) The message's sender is always identifiable by the receiver;

(4) An arbitrary processor is chosen as a source, and its initial value v_s is transmitted to other processors and itself for executing the protocol.

Based on these assumptions, the BA requirement can be satisfied when the following constraints are met:

(Agreement): All correct processors agree on a common value.

(Validity): If the source processor is correct, then all correct processors agree on the initial value sent by the source processor.

Under such assumptions, several protocols [1,5,9,12] have been proposed for solving the BA problem. Lamport has also proved that the solution is impossible if the number of faulty processors exceeds one-third of the total number of processors in the network [14]. Further, Fischer and Lynch [12] pointed out that t＋1 rounds are the minimum number of rounds to get enough messages to achieve BA. However, most of the distributed computing systems would not be fully connected. The SFN has the property of virtual channel [17]; the system still executed message exchange through virtual channel when it is not direct connection between processors. How the processors reach an agreement in the SFN must also be concerned deeply.

The symptom of a faulty processor is usually unrestrained, and is commonly called malicious fault [6]. In such a fault, a processor can withhold the messages to be sent and send irregular message or collide with other faulty processors. A malicious fault is unpredictable, and the behaviors of the other failure types can be treated as special cases of a malicious fault. However, if the malicious fault, which is the thorniest fault, can be solved, then the other fault types [6] can surely be solved. Therefore, in this paper we just investigate the malicious fault and explore how processors reach agreement in the SFN.

The rest of this paper is organized as follows: Section 2 describes the characteristic of SFN. Then, our new protocol will be brought up and illustrated in detail in Section 3. Section 4 gives an example of executing the proposed protocols. Section 5 is responsible for proving the correctness and complexity of our new protocols. Finally, in Section 6, we shall come to the conclusion.

2. Network topology

In the past, the complex network is regarded as a random network [8]. Random network theory maintains that though the processor linking is random, but will present the regular system finally, in other words the number of most processor connective are similar, the way of processor distribution will be present Poisson Distribution [8] in Fig. 1.

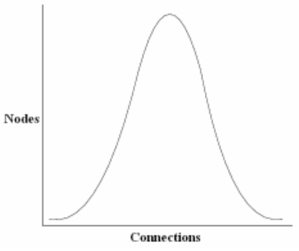


Fig. 1 Poisson Distribution

There are some processors can link unrestrictedly any other processors in the Scale Free Network (SFN). If has deleted gradually processor of smaller connection until 20% of the processors exist in the scale free network topology, the system was still normal operate, in other word, in the condition of the processor damage 80%, the system was also still normal operate [3]. Basically, the WWW and Internet is linked together by web servers of several high connection, it connectivity distribution conform to Power Law within characteristic of exponential decay [2]. Any processors and other *k* processors connectively probabilities are proportional and described as function has shown decrease continually.

When most processors were attacked by virus in the random network, the system would be paralysis. However, 80% of the processors in the SFN were attacked, the system still normal working. SFN owns virtual cannel property; even if it is not keep connect between any two processors. SFN explained very strong ability to support for accidental breakdown.

If new processors unlimited increase which linear speed in SFN model, **Growth** that is SFN own property New processor will be connected other exist processors that high connectivity,

Preferential Attachment that is SFN own property. SFN is shown in Fig. 2. The rule of SFN own processor connective quantity as shown in Formula (1). The purport of processors of own much connection degree in network could more connect with new processors.

$$\prod(K_i) = k_i \big/ \sum_j k_j \qquad \text{……….. Formula (1)}$$

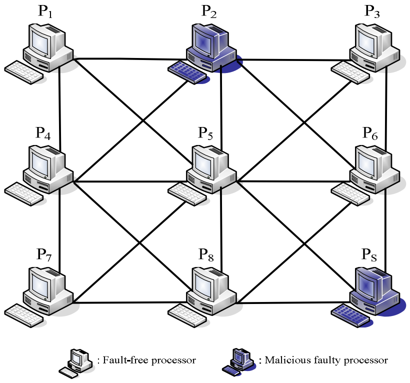


Fig. 2 SFN

3. The definitions and condition for BA problem

In this study, the BA problem is discussed in a synchronous environment. In order to solve the BA problem, the system model, the number of rounds of message exchange required, and the number of tolerable faulty processors should be considered.

An example of SFN is shown in Fig. 2. In a synchronous network, the bounds of delay for each correct component are finite [11,13,16]. The assumptions and parameters of our protocols are listed as follows:

- ♦ The underlying network is synchronous.
- ♦ Each processor in the network can be identified uniquely.
- ♦ Let N be the set of all processors in the network and |N|=*n*, where *n* is the number of processors in the underlying network.
- ♦ The processors of the underlying network are assumed to be fallible.
- ♦ A processor that transmits messages is called a source processor. There is only one source processor who transmits the message at the first round in the BA problem.
- ♦ Let f_m be the maximum number of malicious faulty processors.
- ♦ Let *c* be the connectivity of the SFN.
- ♦ A processor does not know the faulty status of other processors.

In Lamport's protocol [14], the fallible component is processor only, the failure type of the fallible processor is malicious and network topology is fully connected. So that, the constraints of Lamport [14] is $n>3f_m$ and $c=n$-1. In Meyer [15], the assumption of failure types of the fallible processor is malicious faults, and the underlying network topology may not be fully connected. So, the constraint of Meyer [18] is $n>3f_m$ and $c>2f_m$. However, Siu et al. [18] find that the correct constraint on number of processors required should be $n > \lfloor(n\text{-}1)/3\rfloor+2f_m$.

In this paper, our protocol is used to solve the BA problem in a SFN with fallible processors and the assumption of the fallible components is malicious faults. Therefore, the constraints are as follows:

(Constraint 1): $f_m\leq\lfloor(n\text{-}1)/3\rfloor$.
(Constraint 2): $c>2\,f_m$.

The constraint with the connectivity of an SFN is based on the number of malicious faults, In addition, the total number of malicious faults must be smaller than half of c. Hence, the constraint as to the connectivity of a SFN is $c>2f_m$.

There are two phases in our protocol SFNP$_P$ (Scale Free Network Protocol for processor): the message exchange phase and the decision making phase. The SFNP$_p$ is shown in Fig. 3.

Protocol SFNP$_P$	
Definition	
n: the number of processors in the SFN γ: the number of rounds required v_s: the initial value of source ϕ: the default value	
Message Exchange Phase:	
γ=1 do:	1) The source processor transmits its value v_s to other processors and itself through virtual channel. 2) Each processor stores v_s received through virtual channel in the root of its ms-tree.
For γ>1, do:	1) Each processor transmits the value at (γ-1)-th of its ms-tree to other processors and itself through virtual channel. 2) Each processor stores the received values through virtual channel in the corresponding vertices at level of γ its ms-tree.
Decision Making Phase:	
Step 1:	Reorganizing the ms-tree into a corresponding ic-tree. (The vertices with repeated names are deleted)
Step 2:	The root s of each processor's ic-tree and obtaining the common value VOTE(s) by using function VOTE.
Function VOTE(α)	
1. val(α), if the α is a leaf. 2. The majority value in the set of {VOTE(αi)|1≤i≤n, and vertex αi is a child of vertex α}, if such a majority value exists. 3. A default value ϕ is chosen, otherwise.	

Fig. 3 The proposed protocol SFNP$_P$

In the message exchange phase, we collect enough messages through virtual channel, which needs f_m+1 rounds of message exchange, where $f_m=\lfloor(\text{n-1})/3\rfloor$. In the first round of message exchange (γ=1), the source processor transmits its initial value to each processor through virtual channel and then each receiver processor stores the value from in the root s of its ms-tree [19]. The ms-tree is a tree structure that is used to store the received message. After the first round of message exchange (γ>1), each processor transmits the value at (γ-1)-th of its ms-tree to other processors and itself, than stores the received values at level of γ its ms-tree.

In the decision making phase, all correct processors reorganize the ms-tree into a corresponding ic-tree by deleting vertices with repeated group names [19]. Finally, all correct processors use function VOTE to remove the faulty influence from malicious faulty processors to obtain the common value. Since VOTE is a common value, each correct processor can agree on the value, and the agreement is reached. An example of executing SFNP$_P$ is in Section 4.

4. An example of executing Protocol SFNP$_P$

An example for executing SFNP$_P$ is given in this section. An SFN is shown in Fig. 2, there are 9 processors falling in an SFN. The malicious faulty processors are processor P$_S$ and P$_2$, others are correct. The worst case of the BA problem is that the source processor is a malicious faulty processor. If the BA problem can be solved in the worst case, the BA problem also can be solved in other cases. In this example, we suppose processor P$_S$ is the source processor that is a malicious faulty processor. In this case, the number of rounds of messages exchange is 3 ($\lfloor(n\text{-}1)/3\rfloor+1=\lfloor(9\text{-}1)/3\rfloor+1$).

In the first round of the message exchange phase, the source processor P$_S$ transmits messages to other processors through virtual channel. The messages sent by the source processor P$_S$ are shown in Fig. 4(a). The message stored by each correct processor in the first round of the message exchange phase is illustrated in Fig. 4(b). In the γ-th (γ>1) round of message exchange, each processor transmits values at the (γ-1)-th level in its ms-tree to the others and itself through virtual channel. Then, Each processor stores the received values at level of γ its ms-tree. The ms-tree of correct processor P$_1$ at the second and third round in the message exchange phase is shown in Fig. 4(c) and Fig. 4(d).

In the decision making phase, each correct processor turns its ms-tree into a corresponding ic-tree by deleting the vertices with duplicated names. An example of processor P$_1$ reorganizes its ms-tree into the corresponding ic-tree is illustrated in Fig. 4(c) and Fig. 4(d). Finally, using function VOTE to root s of each processor's ic-tree and the common value 1 is obtained. An example of correct processor P$_1$ uses function VOTE to root s is shown in Fig. 4(f).

5. The correctness and complexity of SFNP$_P$

In this section, the correctness and complexity will be proved. The first subsection will prove the correctness of SFNP$_P$, and the complexity will be proved in the next subsection.

To prove protocol's correctness, a vertex α is called common [9] if α of each correct processor has the same value. Thus the agreements, (Agreement) and (Validity), can be rewritten as:

(**Agreement'**): Root *s* is common, and
(**Validity'**): VOTE(*s*)=v_s for each correct processor, if the commander is correct.

To prove that a vertex is common, the term common frontier [7] is defined as: When every root-to-leaf path of the mg-tree contains a common vertex, then the collection of the common vertices forms a common frontier. Based on these two terms, the correctness of SFNP$_P$ can be examined as follows. Before proving the correctness of SFNP$_P$, the term correct vertex is defined as: Vertex αi is a correct vertex if P_i is correct.

Lemma 1 All correct vertices of the ic-tree are common.
Proof: After reorganization, no repeatable vertices are in an ic-tree. At the level f_m+1 or above, the correct vertex has at least 2f_m+1 children, in which at least f_m+1 children are correct. The true value of these f_m+1 correct vertices is common, and the majority value of vertex is common. The correct vertex is common in the ic-tree if the level is less than f_m+1. As a result, all correct vertices of the ic-tree are common.

Lemma 2 The common frontier exists in the ic-tree.
Proof: There are f_m+1 vertices along each root-to-leaf path of an ic-tree in which the root is labeled by the source name, and the others are labeled by a sequence of group names. Inasmuch as most f_m processors can be failed, at least one vertex is correct along each root-to-leaf path of the ic-tree. By Lemma 1, the correct vertex is common, and the common frontier exists in each correct processor's ic-tree.

Lemma 3 Let α be a vertex, if there is a common frontier in the subtree rooted at α, then α is common.
Proof: If the height of α is 0 and the common frontier (α itself) exists, then α is common. If the height of α is *r*, the children of α are all in common by using induction hypothesis with the height of the children at *r*-1, and then the vertex α is common.

Corollary 1 If the common frontier exists in the ic-tree, then the root is common.

Theorem 1 The root of a correct processor's ic-tree is common.
Proof: By Lemma 2 and Corollary 1, the theorem is proved.

Theorem 2 Protocol SFNP$_P$ can solve the BA problem in the SFN.
Proof: To prove the theorem, it has to show that SFNP$_P$ meets the constraints (Agreement') and (Validity').

(**Agreement'**): Root *s* is common. By Theorem 1, it is satisfied.
(**Validity'**): VOTE(*s*) = *v* for all correct processors, if the initial value of the source is v_s, say v=v_s.

Since most of processors are correct, they transmit the message to all others. The value of correct vertices for all correct processors' ms-tree is *v*. When the ms-tree is reorganized to an ic-tree, the correct vertices still exist. As a result, each correct vertices of the ic-tree is common (Lemma 1), and its true value is *v*. By Theorem 1, this root is common. The computed value VOTE(*s*) = *v* is stored in the root for all correct processors. (Validity') is satisfied.

The complexity of SFNP$_P$ is evaluated in terms of (1) the number of rounds required, and (2) the number of allowable faulty processors.

Theorem 3 SFNP$_P$ requires (f_m+1) of message exchanges and can tolerate f_m ≤⌊(n-1)/3⌋ faulty processors.
Proof: By Lemma 1, the ambiguity due to at most f_m (≤⌊(n-1)/3⌋) faulty processors can be resolved. Hence, the theorem is proved.

Theorem 4 Protocol SFNP$_P$ solves BA problem by using the number of message exchanges and it is minimum.
Proof: Fischer [9] pointed out that f_m+1 rounds are the minimum number of rounds to get enough messages to achieve BA. The unit of Fischer is same with SFNP$_P$. Thus, the number of required rounds of message exchange in the SFNP$_P$ is (f_m+1) and this number is the minimum.

Theorem 5 The number of allowable faulty processors f_m (⌊(n-1)/3⌋) in SFNP$_{P\,I}$ is the maximum.
Proof: According to Lamport [14], the total number of allowable faulty processors cannot exceeds ⌊(n-1)/3⌋.

6. Conclusion

BA is a fundamental problem in distributed system; there are many relative literatures in the past [1,7,14]. Network topology is an important cause when we discuss BA. However, all past literatures investigate in random network, different from Internet network scale free characteristic appeared at present. In this paper, a new protocol SFNP$_P$ is proposed to solve BA in a SFN. The SFNP$_P$ protocol redefines the BA in a SFN and can achieve a common value if the constraint on number of processors required should be $f_m \le \lfloor (n-1)/3 \rfloor$ is satisfied.

Reference

[1] A. Bar-Noy, et al., "Shifting gears: changing algorithms on the fly to expedite byzantine agreement," Proceedings of the Symposium on Principles of Distributed Computing, 1987, pp. 42–51.

[2] A. L. Barabàsi, R. Albert, and H. Jeong, "Mean-field Theory for Scale-free Random Networks," Physica A, vol.272, pp.173-187, 1999.

[3] A. L. Barabàsi, R. Albert, and H. Jeong, "Scale-free Characteristics of Random Network: the Topology of the World-wide Web," Physica A, vol.281, pp.69-77, 2000.

[4] A. L. Barabàsi and R. Albert, "Statistic Mechanics of Complex Networks," Reviews of Modern Physics, vol.48-94, 2002.

[5] M. Barborak, M. Malek, A. Dahbura, "The consensus problem in fault-tolerant computing," ACM Computing Surveys 25 (1993) 171– 220.

[6] O. Babaoglu, R. Drummond, "Streets of Byzantium: network architectures for fast reliable broadcasts," IEEE Transactions on Software Engineering SE-11 (6) (1985 June) 546– 554.

[7] M. Correia, L. C. Lung, N. F. Neves, and P. Verıssimo, "Efficient Byzantine-resilient reliable multicast on a hybrid failure model," In Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems, pp. 2–11, 2002.

[8] M. Correia, N. F. Neves, L. C. Lung and P. Verıssimo, "Low Complexity Byzantine-Resilient Consensus," Distributed Computing, vol. 17, pp. 237-249, 2005.

[9] P. Dasgupta, Agreement under faulty interfaces, Information Processing Letters 65 (1998) 125– 129.

[10] P. Erdos and A. Renyi, "On the Evolution of Random Graphs," Pub. Math. Inst. Hung. Acad. Sci., vol. 5, pp. 17-60, 1960.

[11] M. Fischer, M. Paterson, N. Lynch, "Impossibility of distributed consensus with one faulty process," Journal of ACM 32 (1985) 374–382.

[12] M. Fischer, N. Lynch, "A lower bound for the assure interactive consistency," Information Processing Letters 14 (4) (1982 June) 183–186.

[13] F. Halsall, "Data Links, Computer Networks and Open Systems," 4th ed., Addison-Wesley Publishers, 1995, pp. 112–125 Ch. 3.

[14] L. Lamport, R. Shostak and M. Pease, "The Byzantine Generals Problem," ACM Transactions on Programming Languages and System, vol. 4, no. 3, pp.384-401, 1982.

[15] F.J. Meyer, D.K. Pradhan, "Consensus with dual failure modes," IEEE Transactions on Parallel and Distributed Systems 2 (2)(1991 April) 214– 222.

[16] A. Silberschatz, P.B. Galvin, G. Gagne, Operating System Concepts, 6th ed., John Wiley & Sons, 2002.

[17] D. S. Suk, and S. M. Reddy, "Test Procedures for a Class of Pattern-Sensitive Faults in Semiconductor Random-Access Memories," IEEE Transactions on Computers, vol. C-29, no. 6, pp. 419-429, 1980.

[18] H.S. Siu, Y.H. Chin, W.P. Yang, A note on consensus on dual failure modes, IEEE Transactions on Parallel and Distributed System 7 (3) (1996) 225–230.

[19] K.Q. Yan and S.C. Wang, "The Bounds of Faulty Components on Consensus with Dual Failure Modes," in *ACM Operating Systems Review*, Vol. 39, No.3, July 2004, pp. 82-89.

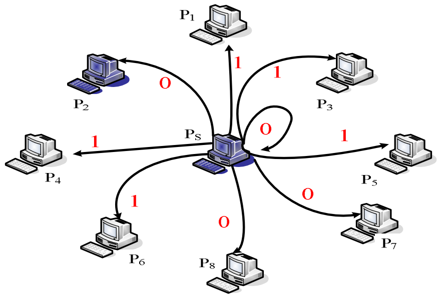


Fig. 4(a) The message sent from the source processor

	Level 1 root S
Fault-free processor P$_1$	1
Fault-free processor P$_3$	1
Fault-free processor P$_4$	1
Fault-free processor P$_5$	1
Fault-free processor P$_6$	1
Fault-free processor P$_7$	0
Fault-free processor P$_8$	0

Fig. 4(b) The message received by each correct processor in the first round

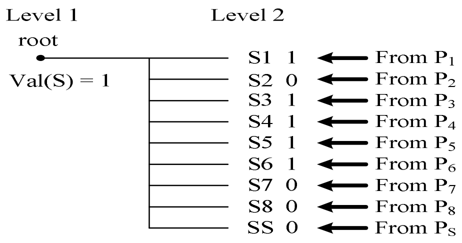
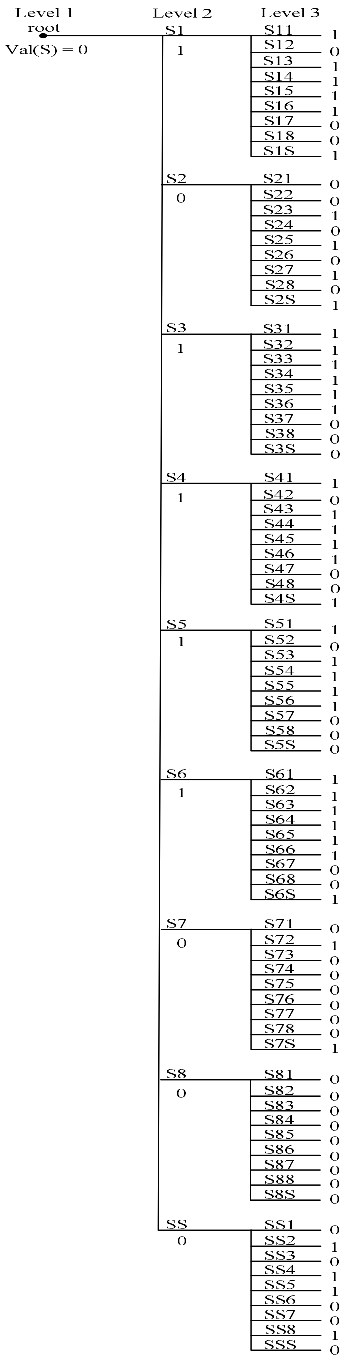


Fig. 4(c) The ms-tree of processor P$_1$ at the second round of message exchange phase



Turn the ms-tree into the corresponding ic-tree by deleting the vertices with repeated names

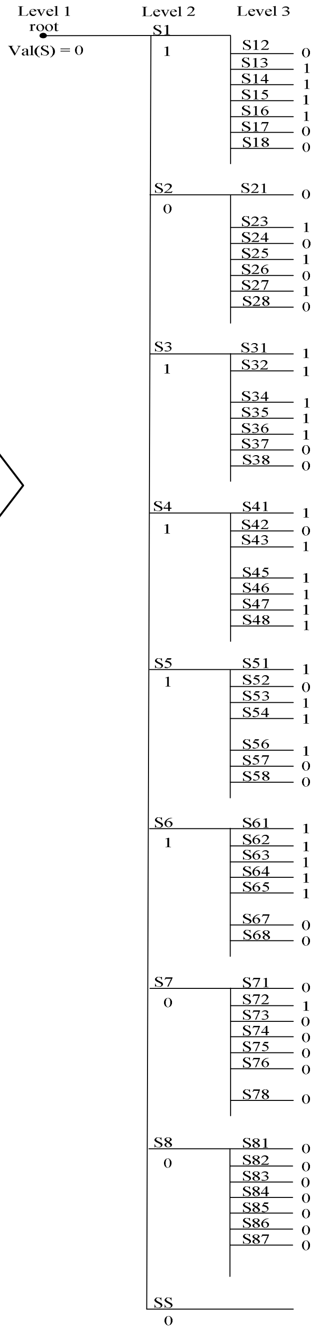


Fig. 4(d) The final ms-tree of processor P_1 after the message exchange phase Fig. 4(e) The ic-tree of processor P_1

VOTE(s1)= VOTE(0,1,1,1,1,1,0)=1 VOTE(s4)= VOTE(1,0,1,1,1,1,1)=1 VOTE(s7)= VOTE(0,1,0,0,0,0,0)=0
VOTE(s2)= VOTE(0,1,0,1,0,1,0)=0 VOTE(s5)= VOTE(1,0,1,1,1,0,0)=1 VOTE(s8)= VOTE(0,0,0,0,0,0,0)=0
VOTE(s3)= VOTE(1,1,1,1,1,0,0)=1 VOTE(s6)= VOTE(1,1,1,1,1,0,0)=1

VOTE(s)=VOTE(1,0,1,1,1,1,0,0)=1

Fig. 4(f) The common value VOTE(s) by correct processor P_1

Fig. 4. An example of reaching an agreement in SFN