Dual Fallible Consensus in a Scale Free Network

S.C. Wang **S.C. Yang** **K.Q. Yan**
scwang@cyut.edu.tw **s9430619@cyut.edu.tw** **kqyan@cyut.edu.tw**
Chaoyang University of Technology

摘要

在分散式系統中，當要執行一些特定的任務時，必須事先達成一個共同的合議，當合議達成後分散式系統可以對抗損毀元件的影響，而這種問題稱為合議問題。合議問題可以使運作健全的處理器排除掉損毀元件的影響而達成協議，並且增加工作執行的可靠度。過去有關合議問題的研究中，大都是在隨機式的網路中進行，然而現今實際的網路環境卻是異常複雜且呈現無尺度之架構。因此本研究將在一個更符合目前以及未來需求的無尺度網路架構下，尋找出最快以及容錯能力最好且可達成合議的方法。透過無尺度網路所具有的成長性和優先連結性，將使合議問題的實用性及適用性大增。

關鍵詞：合議，無尺度網路，分散式系統，容錯

Abstract

Generally, the task in a distributed system must achieve consensus. It requires a set of processors to agree on a common value even if some components are corrupted. There are significant studies on this consensus 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 consensus protocols and results cannot cope with the new network environment and the consensus problem thus needs to be revisited. In this paper, we propose a new consensus protocol to adapt to the scale-free network environment and derive its bound of allowable faulty transmission media (TMs) with two rounds of message exchange. It is observed that the scale-free network with the proposed consensus protocol can tolerate more faulty TMs than the networks based on previous studies.

Keywords: Consensus, Scale-free network, Distributed system, Fault tolerance.

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 [4]. It is an important topic for can represent the structure of highly fault-tolerant ability in the Scale-Free Network (SFN) [9] seems to provide a new topic. SFN is universal in real world or in operation of the human body function [2]. In SFN, each processor can infinite link other processors. If delete these processors of smaller connectivity until 20% of the processors exist, the network still normal operation in SFN environment [7].

However, under a large amount of calculation 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 [3]. To confirm the consensus 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.

To achieve a consensus on a predefined value in a distributed system, protocols are necessary so that the system will run even if certain components in the distributed system were failed. Such a unanimity problem is named as consensus problem, has been studied extensively in the literature [5,10]. The definition of the problem is to make the correct processors in an n processor distributed system to reach consensus. Each processor chooses an initial value to start with, and communicates each other by means of message exchange. The desired protocol is to solve the consensus problem if it satisfies the following constraints:

(Agreement):All correct processors agree on the same value.

(Validity): If the initial value of all processors is v_i then all correct processors shall agree on v_i.

Conventionally, many results in consensus problem are based on the assumption of processor failure in a fail-safe network [6-8]. Based on this assumption, a TM fault is unfairly treated as a processor fault [8], regardless the correctness of an innocent processor; therefore an innocent processor does not involve a consensus. This contradicts with

the definition of consensus problem that requires all correct processors to reach consensus. Yan et al. [10] had solved this problem but they treated all TM failures as malicious. Actually, the symptom of a faulty TM can be classified into two types: dormant and malicious. The dormant faults of a TM always can be identified by the receiver if the transmitted message was encoded appropriately (i.e. by NRZ-code, Manchester code [10]) before transmission. On the other hand, the malicious faulty TMs are unpredictable. We revisit consensus problem to enlarge the fault tolerant capability by allowing both dormant faults and malicious faults exist in the system simultaneously.

As the same with Yan et al. [10], we consider a distributed system whose processors are reliable during the consensus execution; while message TMs may be disturbed by some faults, break down, stuck-at, noise or an intruder. A new efficient and reliable protocol to achieve consensus in an unreliable communication environment is proposed first; then its efficiency and reliability are proved later. The common term round [7] is used to denote the interval of message exchange.

However, in this paper, the protocol Dual Fallible TM Protocol (DFTMP) is proposed, it can use the minimum number of message exchanges to make each correct processors in SFN reaches consensus.

The rest of this paper is organized as follows. Section 2 describes the characteristic of scale free network and the security technology our research used. Section 3 presents in detail a new protocol DFTMP for the SFN and gives an example of executing DFTMP. The analysis of fault tolerance capability is shown in Section 4. Subsequently, Section 5, the correctness and complexity of DFTMP is given. Finally, Section 6 concludes this paper.

2. Related Work

The design and development of the consensus protocol has several requirements that must be considered. The secure communication is one of the important topics to provide secure communication in the network. The network topology and the security technology that is used in our research have discussed in this section.

Fig. 1 shows a SFN where the network continually grows by adding a new processor. The newly introduced processor chooses to connect a processor with k TMs that is favoring highly connected processors. This phenomenon is called *preferential attachment*. According to this new observation, each processor in the SFN has different connectivity and follows the power law.

In the SFN, the processors are interconnected with the Internet; the network is assumed reliable and synchronous. Reaching consensus on a same value in a distributed system, even if certain components in distributed system were failed (inner damage or outer intruder), the protocols are required so that systems still can be executed correctly. However, there are two characteristics of Trusted Timely Computing Base (TTCB): security and synchronization [3]. Therefore, in this paper, the transmission with intact information by way of TTCB is used when the message is passing.

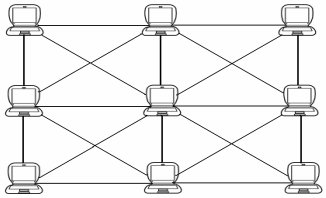

Fig. 1. SFN

3. The Consensus Protocol for SFN

The paper proposed a new protocol, called Dual Fallible TM Protocol (DFTMP), to solve the consensus problem due to faulty TMs, which may send wrong messages to influence the system to achieve consensus in a SFN. There are two phases in DFTMP, the message exchange phase and the decision making phase, as shown in Fig. 2.

Protocol DFTMP (for each processor P_i with initial value v_i)	
Definition	
V_i:	the vector of processor P_i
ϕ:	the default value
?:	no majority value
v_{ki}:	the k-th element in vector V_i
TM_{ij}:	the TM between processor P_i and P_j
V_{ij}:	the initial value v_i sending to P_j received by P_i
MAT_i:	all vector from other processor received by P_i
MAJ_k:	a majority function used to remove the influence of a faulty TM on messages stored in the vector V_i of MAT_i
DEC_i:	the decision value of processor P_i
Message Exchange Phase:	
Round 1:	Multicast the initial value v_i of P_i by TTCB through c_i TMs to other processors. Receive value v_j by TTCB from other processor P_j connecting to P_i via TM. Construct vector V_i. If a dormant TM, say TM_{ij}, was found, then $v_{ij}=\lambda$.
Round 2:	Multicast the vector V_i by TTCB to other processors. Receive the vector V_j by TTCB sent by other processors P_j and construct MAT_i. If a dormant TM was found then $V_k=[\lambda, \lambda, \ldots, \lambda, .. \lambda]$.
Decision Making Phase:	
Step 1:	Eliminate all λ to lessen the influence of faulty behavior, and take the majority value of each row k of MAT_i as MAJ_k.
Step 2:	If ($\exists MAJ_i = \neg v_i$), then $DEC_i=\phi$; else if ($\exists MAJ_k$ =?) and ($v_{ki}= v_i$) then $DEC_i=\phi$, else $DEC_i = v_i$ and halt.

Fig. 2. DFTMP

Moreover, DFTMP only needs two rounds of message exchange to solve the consensus problem. In the first round of the message exchange phase, each processor P_i transmits its initial value v_i through TMs by TTCB and then receives the initial value of other processors as well. In the second round, each processor P_i acts as the sender, sending the vector received in the first round, and receives the vector V_j by TTCB sent by other processors P_j to construct a matrix, called the MAT_i, $1 \le i,j \le n$. Finally, the decision making phase, each processor removes all λ in MAT_i to reduce the influence of faulty behaviors and takes the majority value of MAT_i to construct the matrix MAJ_i, and achieves the common value by MAJ_i. The proposed protocol DFTMP can achieve the maximum number of fault tolerance within the bound of $\lfloor (c_\delta+1)/2 \rfloor - 1 \le f_t \le \left\lfloor \sum_{i=1}^{n} (\lfloor (c_i+1)/2 \rfloor - 1)/2 \right\rfloor$ (f_t is the total number of allowable fault TMs, c_δ is the smallest connectivity) in a SFN.

Subsequently, an example of executing the DFTMP protocol based on the network configuration shown in Fig. 3(a). For illustration, we will assume that TM_{12}, TM_{36}, TM_{47}, and TM_{89} are dormant fault TMs and TM_{25}, TM_{45}, and TM_{56} are malicious fault TMs. We also assume that the initial value of P_1, P_3, P_7, and P_9 is 0, P_2, P_4, P_5, P_6, and P_8 is 1.

In the first round of the message exchange phase, each processor P_i uses the received message to construct vector V_i as shown in Fig. 3(b). In the second round, each processor P_i sending the vector received in the first round, and receives the vectors from other processors to constructs a matrix MAT_i as shown in Fig. 3(c). In the decision making phase, each processor removes all λ in MAT_i to reduce the influence of dormant faulty behaviors and to take the majority value of MAT_i to construct the matrix MAJ_i, as shown in Fig. 3(d), and achieves the common value 1 by MAJ_i.

4. Fault tolerance capability analysis

According to literatures [5-8], the number of allowable faulty TMs within the bound in a system is $f_t < \lfloor c/2 \rfloor$ and $c > 2m+d$, where f_t is the total number of allowable faulty TMs, c is the connectivity of network [10], m is the number of malicious faults, and d is the number of dormant faults. Moreover, the SFN has the property of virtual channel [9]; the system still executed message exchange through virtual channel when it is not direct connection between processors. So, the number of allowable fault TMs for each processor P_i can be analyzed according to the results for c-connectivity protocols.

Worst case:

The worst case occurs when faulty TMs focus on those processors that have the minimum connectivity. In other words, if the number of faulty TMs greater than the half of the processor's connectivity then the processor cannot catch enough information to reach consensus. Therefore, the number of allowable faulty TMs in the worst case is $f_t \ge \lfloor (c_\delta+1)/2 \rfloor - 1$ and $c > 2m+d$, where c_δ is the smallest connectivity of the system.

For example, such as processors P_1, P_3, P_7 and P_9 in Fig. 3(a) have the smallest connectivity $c_\delta = 3$. According to the constraints $f_t \ge \lfloor (c_\delta+1)/2 \rfloor - 1$ and $c > 2m+d$, the maximal number of allowable faulty TMs can be zero malicious fault and two dormant fault TMs (m=0, d=2), or one malicious fault and zero dormant fault TM (m=1, d=0).

Base case:

The best case occurs when faulty TMs concentrate on those processors that have the maximum connectivity and the faulty TMs disperse to each processor in the scale free network. Only each processor can catch enough information then the consensus will be reached. In other words, the number of allowable faulty TMs in the best case is $f_t \le \left\lfloor \sum_{i=1}^{n} (\lfloor (c_i+1)/2 \rfloor - 1)/2 \right\rfloor$, where c_i is the connectivity of P_i.

For example, the connectivity of P_5 is 8, P_2, P_4, P_6, and P_8 is 5, P_1, P_3, P_7, and P_9 is 3 in Fig. 3(a). The maximum number of allowable faulty TMs is $\left\lfloor \sum_{i=1}^{n} (\lfloor (c_i+1)/2 \rfloor - 1)/2 \right\rfloor = (1+2+1+2+3+2+1+2+1)/2 = 7$.

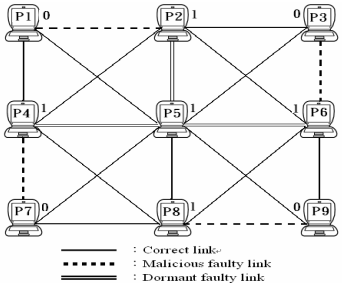

Fig. 3 (a). 9-processor SFN

V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	1	0	0	0	0	0	0	0
0	1	1	1	λ	1	1	1	1
0	0	0	0	0	1	0	0	0
1	1	1	1	λ	1	0	1	1
1	λ	1	λ	1	λ	1	1	1
1	1	0	1	λ	1	1	1	1
0	0	0	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	1	0

Fig. 3(b). The vector received in the first round

MAT_1								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	0	0	0	0	0	0	0	0
0	0	1	1	λ	1	1	1	1
0	1	0	0	0	1	0	0	0
1	0	1	1	λ	1	0	1	1
1	λ	1	λ	1	λ	1	1	1
1	0	0	1	λ	1	1	1	1
0	1	0	1	0	0	0	0	0
1	0	1	1	1	1	1	1	0
0	1	0	0	0	0	0	1	0

MAT_2								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
1	1	0	0	λ	0	0	0	0
1	1	1	1	λ	1	1	1	1
1	0	0	0	λ	1	0	0	0
0	1	1	1	λ	1	0	1	1
0	λ	1	λ	λ	λ	1	1	1
0	1	0	1	λ	1	1	1	1
1	0	0	1	λ	0	0	0	0
0	1	1	1	λ	1	1	1	0
1	0	0	0	λ	0	0	1	0

MAT_3								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	1	0	0	0	1	0	0	0
0	1	1	1	λ	0	1	1	1
0	0	0	0	0	0	0	0	0
1	1	1	1	λ	0	0	1	1
1	λ	1	λ	1	λ	1	1	1
1	1	0	1	λ	0	1	1	1
0	0	0	1	0	1	0	0	0
1	1	1	1	1	0	1	1	0
0	0	0	0	0	1	0	1	0

MAT_4								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	1	0	0	λ	0	1	0	0
0	1	1	1	λ	1	0	1	1
0	0	0	0	λ	1	1	0	0
1	1	1	1	λ	1	1	1	1
1	λ	1	λ	λ	λ	0	1	1
1	1	0	1	λ	1	0	1	1
0	0	0	1	λ	0	1	0	0
1	1	1	1	λ	1	0	1	0
0	0	0	0	λ	0	1	1	0

MAT_5								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	λ	0	λ	0	λ	0	0	0
0	λ	1	λ	λ	λ	1	1	1
0	λ	0	λ	0	λ	0	0	0
1	λ	1	λ	λ	λ	0	1	1
1	λ	1	λ	1	λ	1	1	1
1	λ	0	λ	λ	λ	1	1	1
0	λ	0	λ	0	λ	0	0	0
1	λ	1	λ	1	λ	1	1	0
0	λ	0	λ	0	λ	0	1	0

MAT_6								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	1	1	0	λ	0	0	0	0
0	1	0	1	λ	1	1	1	1
0	0	1	0	λ	1	0	0	0
1	1	0	1	λ	1	0	1	1
1	λ	0	λ	λ	λ	1	1	1
1	1	1	1	λ	1	1	1	1
0	0	1	1	λ	0	0	0	0
1	1	0	1	λ	1	1	1	0
0	0	1	0	λ	0	0	1	0

MAT_7								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	1	0	1	0	0	0	0	0
0	1	1	0	λ	1	1	1	1
0	0	0	1	0	1	0	0	0
1	1	1	0	λ	1	0	1	1
1	λ	1	λ	1	λ	1	1	1
1	1	0	0	λ	1	1	1	1
0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	0
0	0	0	1	0	0	0	1	0

MAT_8								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	1	0	0	0	0	0	0	1
0	1	1	1	λ	1	1	1	0
0	0	0	0	0	1	0	0	1
1	1	1	1	λ	1	0	1	0
1	λ	1	λ	1	λ	1	1	0
1	1	0	1	λ	1	1	1	0
0	0	0	1	0	0	0	0	1
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1

MAT_9								
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0	1	0	0	0	0	0	1	0
0	1	1	1	λ	1	1	0	1
0	0	0	0	0	1	0	1	0
1	1	1	1	λ	1	0	0	1
1	λ	1	λ	1	λ	1	0	1
1	1	0	1	λ	1	1	0	1
0	0	0	1	0	0	0	1	0
1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0

Fig. 3(c). Construct the matrix MAT_i

Majority value of each row of MAT_i (i=1~9) =

0
1
0
1
1
1
0
1
0

⇨ 1

Fig. 3(d). Take the majority value

Fig.3. An example of reaching consensus in a SFN

5. The Correctness and Complexity

According to the constraints on consensus problem, the protocol DFTMP must satisfy the agreement and validity condition if the system reaches a common value. The lemmas and theorems are used to prove the correctness and complexity of protocol DFTMP.

Lemma 1: If there is a majority value = $\neg v_i$ in MAT_i, then there is at least one processor with an initial value which disagrees with v_i in the SFN.

Proof: The majority value in the k-th row = $\neg v_i$ means that there are at least $\lceil (n-d+1)/2 \rceil$ v_i's in the k-th row (d is the number of dormant). Since the number of malicious faulty TMs is at most $\lfloor (c-d+1)/2 \rfloor$-1, and $(\lfloor (n-d)/2 \rfloor +1)-(\lfloor (c-d+1)/2 \rfloor -1)=\lfloor (n-c+3)/2 \rfloor$, there exists at least one value $\neg v_i$ received from a correct TM. In other words, a processor has a different initial value $\neg v_i$. ■

Lemma 2: Let the initial value of processor P_i be v_i and the TM_{ij} is correct or dormant, then the majority value at the i-th row in MAT_j should be v_i.

Proof:

Case 1: Since TM_{ij} is correct, the processor P_j will receive v_i from processor P_i in the first round and $v_{ij} = v_i$ in MAT_j. Meanwhile, the value v_i of processor P_i will be broadcast to the others. There are at most $\lfloor (c-d+1)/2 \rfloor$-1 malicious faulty TMs in the system. In the second round, processor P_j receives at least $(n-d-1)-(\lfloor (c-d+1)/2 \rfloor -1)=\lfloor (2n-d-c-1)/2 \rfloor$ v_i's in the i-th row of MAT_j, where d is the number of λ which will be eliminated during the voting of majority. Hence, there are at least $\lfloor (2n-d-c-1)/2 \rfloor$ v_i's in the i-th row, and the majority value in the i-th row should be equal to v_i.

Case 2-1: TM_{ij} is dormant and c is an even number, the processor P_j will receive λ from processor P_i in the first round and v_{ij}=λ in MAT_j. Meanwhile, the value v_i of processor P_i will be broadcasted to the other processors. There are at most $\lfloor (c-d+1)/2 \rfloor$-1 malicious faulty TMs and d dormant TMs in the network. After the second round, processor P_j receives at least (d+1) λ's and at least $n-(d+1)-(\lfloor (c-d+1)/2 \rfloor -1) = \lfloor (2n-d-c-1)/2 \rfloor$ v_i's in the i-th row of MAT_j, where d is the number of λ which will be eliminated during the voting of majority. Hence, there (n-d-1) non-λ's value and least $\lfloor (2n-d-c-1)/2 \rfloor$ v_i's in the i-th row, so, the majority value in the i-th row should be equal to v_i.

Case 2-2: TM_{ij} is dormant and c is an odd number, the processor P_j will receive λ from processor P_i in the first round and v_{ij} = λ in MAT_j. Meanwhile, the value v_i of processor P_i will be broadcasted to the other processors. There are at most $\lfloor (c-d+1)/2 \rfloor$-1 malicious faulty TMs and d dormant TMs in the system. After the second round, processor j receives at least (d+1) λ's and at least $n-(d+1)-(\lfloor (c-d+1)/2 \rfloor -1) = \lfloor (2n-d-c-1)/2 \rfloor$ v_i's in the i-th row of MAT_j, where d is the number of λ which will be eliminated during the voting of majority. Hence, there are n-(d+1) non-λ's and at least $\lfloor (2n-d-c-1)/2 \rfloor$ v_i's in the i-th row, so, the majority value in the i-th row should be equal to v_i. ■

Lemma 3: If the initial value of processor P_i is v_i, whether the TM_{ij} is correct or dormant, the majority value at the i-th row of MAT_j, $1 \le j \le n$, should be either be v_i or not be able to be determined with $v_{ij} = \neg v_i$.

Proof: By Lemma 2, when TM_{ij} is correct or dormant, the majority value of the i-th row in processor P_j is v_i, for $1 \le j \le n$. When TM_{ij} is under the influence of malicious fault, we consider the following two cases after running the first round.

Case 1: $v_{ij} = v_i$. Since there are at most $\lfloor (c-d+1)/2 \rfloor$-1 malicious faulty TMs connected with processor P_j, at most $\lfloor (c-d+1)/2 \rfloor$-1 values that may be $\neg v_i$'s in the second round. The number of v_i's is $(n-d-1)-(\lfloor (c-d+1)/2 \rfloor -1)=\lfloor (n-d)/2 \rfloor +2$ in the i-th row where d is the number of λ which will be eliminated during the voting of majority; therefore, the majority of the i-th row in MAT_i is v_i.

Case 2: $v_{ij} = \neg v_i$. There are at most $\lfloor (c-d+1)/2 \rfloor$-1 malicious faulty TMs. Therefore, in the second round, the total number of $\neg v_i$'s does not exceed $(\lfloor (c-d+1)/2 \rfloor -1)+1=\lfloor (c-d+1)/2 \rfloor$ and the number of v_i's is at least $(n-d-1)-(\lfloor (n-d)/2+1 \rfloor)=\lfloor (n-d-1)/2 \rfloor$. If n-d is an even number, then $\lceil (n-d-1)/2 \rceil = \lfloor (n-d-1)/2 \rfloor$, the majority of the i-th row in MAT_j cannot be determined. If n-d is an odd number, then $\lceil (n-d-1)/2 \rceil > \lfloor (n-d-1)/2 \rfloor$. Hence, the majority of the i-th row in MAT_j is v_i. ■

Lemma 4: If $(\neg \exists MAJ_k = \neg v_i)$ AND $\{(\exists MAJ_k = ?)$ AND $(v_{ki} = v_i)\}$ in MAT_i, then $DEC_i = \phi$ is correct.

Proof: If there has a MAJ_k = ?, there are exactly (n-d)/2 v_i's and (n-d)/2 $\neg v_i$'s in the k-th row. If $v_{ki} = v_i$ in MAT_i, then all (n-d)/2 $\neg v_i$'s should be received in the second round. There are $\lfloor (c-d+1)/2 \rfloor$-1 malicious faulty TMs in the system. Therefore, in the second round, processor P_i at least receives $(n-d)/2-\lfloor (c-d+1)/2 \rfloor -1 \ge 1$ value $\neg v_i$ from processor P_k without disturbance. The initial value of processor P_k should disagree with the initial value of processor P_i; hence it is correct to choose $DEC_i = \phi$. If $v_{ki} = \neg v_i$, we claim that $\neg v_i$ ought to be passed through malicious TM from processor P_i, and the initial value of processor should be $\neg v_{ki} = v_i$. To prove, if TM_{ki} is correct, then the initial value of processor P_k should be $\neg v_i$. By Lemma 2, the majority value of the k-th row in MAT_i is $\neg v_i$. This is contradiction with the condition of $(\neg \exists MAJ_k = \neg v_i)$. If the initial value of processor P_k was $\neg v_i$, then by Lemma 3, MAJ_k should be either $\neg v_i$ or ? for $v_{ki} = v_i$. It is a contradiction. ■

Theorem 1:Protocol DFTMP is correct.

Proof: By Lemmas 1, 2, 3 and 4, the theorem is proved. ■

Theorem 2: Protocol DFTMP can reach a consensus.

Proof:

(1) Agreement:

Part 1: If a correct processor agrees on ϕ, all correct processors should agree on ϕ. If the correct processor P_p with initial value v_i agrees on ϕ, by Theorem 1, there is at least a correct processor P_k with initial value $\neg v_i$ in the network. By Lemma 4, the majority value in the k-th row of MAT_j, $1 \le j \le n$, should be either $\neg v_i$ or ? for $v_{kj} = v_i$. All correct processors with initial value v_i agree on ϕ. Similarly, for the correct processor P_p with initial value $\neg v_i$, the majority value of the p-th row in MAT_j, $1 \le j \le n$, either should be v_i or cannot be determined with $\neg v_{ij} = v_i$. All correct processors with initial value $\neg v_i$ agree on ϕ, too.

Part 2: If a correct processor agrees on v_i, all correct processor s should agree on v_i. If the correct processor P_i with initial value v_i and $DEC_i = v_i$, but there exists some correct processor P_i, $j \neq i$, has $DEC_j \neq v_i$, then that is impossible. To show this, if $DEC_j = \phi$, by Part 1, then $DEC_i = \phi$. This is a contradiction with the assumption as above. If $DEC_j = \neg v_i$, unless the initial value of processor P_i is $\neg v_i$, otherwise it is impossible according to the definition consensus problem. However, if the initial value of processor P_j is $\neg v_i$, by Lemma 4, MAJ_i is equal to $\neg v_i$ or ? with $v_{ij} = v_i$ in MAT_i; then, $DEC_i = \phi$, which is a contradiction. Hence, all correct processors should agree on the same value.

(2)Validity: The initial value of all processors should be the same. If there is a value $\neg v_i$ in MAT_j, $1 \le j \le n$, then the value must be caused by malicious faulty TM. There are at most $\lfloor (c\text{-}d+1)/2 \rfloor - 1$ malicious faulty TMs, hence there are at most $\lfloor (c\text{-}d)/2 \rfloor - 1$ $\neg v_i$'s in each row. Since the value received in the first round may be $\neg v_i$, the majority of each row for all MAT_j, should be MAJ_j= ? (If the value received in the first round is $\neg v_j$, $1 \le j \le n$); or v_j. Thus, by step 2 of the protocol DFTMP, all correct processors should agree on v_i. ■

Theorem 3: The amount of information exchange by DFTMP is $O(n^2)$.

Proof: In the first round, each processor sends out (n-1) copies of its initial value to other processors. In the second round, an n-element vector is sent to the other n-1 processors in the network; therefore, the total number of message exchange is (n-1)+(n * (n-1)). This finding implies that the complexity of information exchange is $O(n^2)$ ■.

6. Conclusion

Consensus problem is a fundamental problem in distributed system; there are many relative literatures in the past [2,12,13]. Network topology is an important cause when we discuss consensus problem. However, all past literatures investigate in random network, different from Internet network scale free characteristic appeared at present. In this paper, a new protocol DFTMP is proposed to solve consensus problem in a SFN. The DFTMP protocol redefines the consensus problem in a SFN and can achieve a common value if the condition $c_i > 2m_i + d_i$ is satisfied. However, the number of allowable faulty TMs is from $\lfloor (c_\delta+1)/2 \rfloor - 1$ to $\left\lfloor \sum_{i=1}^{n} (\lfloor (c_i+1)/2 \rfloor - 1)/2 \right\rfloor$.

References

[1] K. Aberer and M. Punceva "Improving Data Access in P2P Systems," IEEE Internet Computing, pp. 58-67, 2002.

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

[3] 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.

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

[5] M. Fischer and N. Lynchw, "A Lower Bound for the Assure Interactive Consistency," Information Processing Letters, vol. 14, no. 4, pp.183-186, 1982.

[6] Fred Halsall, Data Communications, Computer Networks and Open Systems, 4th, ed., Addison-Wesley Publishers Ltd., 1995, Ch.3, pp. 112-125.

[7] 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.

[8] F. J. Meyer and D. K. Pradhan, "Consensus with Duel Failure Modes," IEEE Transactions on Parallel and Distributed Systems, vol. 2, no. 2, pp. 214-222, 1991.

[9] 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.

[10] K. Q. Yan and S. C. Wang, "The Bounds of Faulty Components on Consensus with Dual Failure Modes," ACM Operating Systems Review, vol. 39, no.3, pp. 82-894, July 2005.