Workshop on Computer Systems

Reaching Real Agreement in an Unknown Network Environment

K.Q. Yan
Department of Business Administration
kqyan@mail.cyut.edu.tw

S.C. Wang (corresponding author)
Department of Information Management
scwang@mail.cyut.edu.tw

M.L. Chiang
Department of Information Management
s9014614@mail.cyut.edu.tw

Chaoyang University of Technology
168 Gifeng E. Rd., Wufeng,
Taichung County, Taiwan 413, R.O.C.
TEL: 886-4-2332-3000 ext. 3071
Fax: 886-4-2374-2319

Abstract

The Byzantine Agreement problem is one of the most fundamental problems in the field of distributed environment. It requires a set of the processors to agree on a common value even if some processors are corrupted. There are many significant studies about these problems in a well-defined network environment. These network like the FCN (Fully Connected Network), GCN (Generalized Connected Network), and SGCN (Super Generalized Connected Network) discussed about vary faults of processors and transmission media. Thus, we can understand the faulty tolerant capability of these network environments. But it is computationally infeasible for ignoring the network structure in a real environment. In general, the processors will turnover the network arbitrary in any time. Thus, the network will become an instable state that influences the system to reach a common value. Subsequently, the fault tolerant capability will unable to be computed due to each processor doesn't know any connectivity of other processors. Thus, this paper will discuss about this serious problem that vary network environment. This kind of the network, we call it as an Unknown Network (UNet) structure. And this paper proposes the solution for Byzantine Agreement in a UNet environment. These assumptions will generate some different results with prior literatures. Similarly, our proposed protocol can solve the Byzantine Agreement using minimum rounds of message exchange, and tolerates the maximum number of transmission media failure.

Keywords: Byzantine Agreement, Consensus, Unknown Network, Fault Tolerant, and Distributed Network.

1. Introduction

The Byzantine Agreement (BA) problem is one of fundamental problems in a distributed fault tolerant environment. This problem first studied by Lamport in 1982 [7] was solved to make system run and agree on a common value even if certain processors in the distributed system were failed. The main model of the problem describes the system contains *n* communicate processors, at most *t* of which are corrupted [2,3,4,9]. Each processor will agree on a common value if the fault numbers less than fault tolerant boundary. The goal will be achieved if an initial value needs to be set in a source processor before reaching a common value among healthy processors. We call this agreement as a Byzantine Agreement problem. A closely related sub-problem, the *consensus* problem, has been extensively studied [18] as well. The main operates as follow, each processor *j* has its own initial value v_j before executing the protocol, $1 \leq j \leq n$. To start the protocol, each processor broadcasts its initial value to all processors in the first round, subsequently, each processor broadcasts the messages that received in the first round and makes the decision. The problem will be solved and achieved if they satisfy the following constraints:

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

(**Validity**): If the initial value of each healthy processor is v_i then all healthy processors shall agree on the value v_i.

In practice, the symptom of faults in a distributed system can distinguish between the processors and transmission media (TM). These faults can be classified as dormant fault (include crash, omission, stuck-at, or timing faults) and malicious fault (Byzantine faults). According to above discussion, we know the network topology is a very important factor. It will assist us to distinguish from these faults, if the network structure is known. But, each processor doesn't know any connectivity of other processors in a real network environment. Thus, we will discuss unknown transmission media of UNet in this paper.

Consequently, this paper considers a vary network environment structure due to the network states will influence the fault tolerant capability deeply. However, many kinds of the discussions in FCN [11], GCN [9], and SGCN [10], but they considered the BA problem or *consensus* problem in a well-defined network. Thus, prior papers will generate some problems that network structure will change if each processor appears or disappears in any time. Although, each processor can achieve a common value under the well-known network, but it is infeasible in a UNet environment. Thus, this paper will enquire about UNet Protocol (UNP) to achieve agreement in a UNet environment.

The rest of the paper is organized as follows. Section 2 describes the previous results. The UNet Protocol (UNP) is explained in section 3. The correctness and complexity of UNP is proved in section 4. Finally, the conclusion is showed in section 5.

2. Previous Results

According to the prior literatures, the allowable faulty transmission media (TMs) are ($\lfloor n/2 \rfloor$-1) in a (n-1)-connected environment [5,6,12,13]. These protocols need two rounds of message exchange and make the healthy processors to reach a common value in the last round [11,12]. Similarly, the influence of a faulty TM produced in the (k-1)th round can be removed by k-th round of message exchange [9,10,11]. Based on the results of Wang [9], we can know that only two rounds required in the message exchange phase for each healthy processor to reach an agreement if the number of faulty TMs are less than ($\lfloor n/2 \rfloor$-1). The messages were influenced by a faulty TM in the first round, but this fault would be cleared in the next round (second round), so we can clearly know the exactly messages and agree on a common value. In this assumption, prior literatures mention c-connectivity protocol [3,4], it allows c-connectivity network can tolerant $\lfloor (c+1)/2 \rfloor$-1 [3,4] fault TMs in general case. But this assumption will decrease the fault tolerant capability. Only the faulty numbers spread equally. Subsequently, next section we will discuss uses our proposed protocol can improve the tolerant numbers from $\lfloor (c+1)/2 \rfloor$-1 to $\left\lfloor \sum_{i=1}^{n} \left(\lfloor (c_i+1)/2 \rfloor - 1 \right) \Big/ 2 \right\rfloor$.

3. The UNet Protocol (UNP)

Meanwhile, prior protocols do not consider about the fault results from the vary processors or TMs [9,10,11,12]. These problems will decrease accuracy of messages and fault tolerant capability. Thus, this paper will confer on each processor step by step to fit the real network environment and find the fault tolerant capability.

Due to prior literatures [3,4], we can gain the protocol which can tolerate any transmission media faults in a system provided $n > 3m + d$ and $c > 2m + d$ [3,4], where n is the total number of processors in the network, c is the system connectivity, m is the number of malicious faults, and d is the number of dormant faults. However, we find the protocol we proposed that it is more flexible to fit an unknown network (UNet) environment. The network will extend or reduce continuously if any processor gets in or out offline in any time. Thus, each processor maybe understands the connection state of the neighbor only via the message exchange. Unfortunately, the network changes soon again and each processor doesn't distinguish from all of the network structures basically. On other hand, each processor only knows itself connection condition at most. Thus, we will take this point to generate our protocol that each processor can know itself connectivity only in an unknown distributed environment.

This paper has different result than before [3,10,13]. Our protocol UNP (UNet Protocol) is introduced to solve the consensus problem in a UNet. The faulty transmission media will send the wrong messages which are changed or delayed and influence the processors to achieve the agreement. Thus, the consensus problem is achieved if the number of faulty transmission media within $\left\lfloor \sum_{i=1}^{n}(\lfloor (c_i+1)/2\rfloor -1)\Big/2 \right\rfloor$. Similarly, each processor will execute the protocol UNP to reach a common value as the same as prior protocols. The protocol UNP have two phases that message exchange phase and decision making phase. In the prior literatures, it needs two rounds to solve the consensus problem if the network has the faulty transmission media only [11,12].

Thus, under the protocol UNP, we can solve the consensus problem using two rounds message exchange in a UNet. In the first round of message exchange, each processor multicasts its initial value v_i through connected transmission media which been knew and then receive other processors' initial value as well. In UNP, each processor is not aware of other processors' connections. If there has no connection exist, the processor won't know that. Each processor will receive the c_i messages according to its connection state and construct the vector through the received messages. In the second round of message exchange, each processor acts as the sender and sends the vectors which be received in the first round and constructs the matrix MAT$_i$. Finally, the decision making phase will reach an agreement among the processors. The proposed protocol UNP is defined in Figure 1.

In the prior literatures [3,4], we can gain a fault tolerant capability protocol $c > 2m + d$ and find the result will be changed from some variables, like dormant fault and malicious fault. It is feasible protocol in a distinct network structure. But the previous protocols will occur a serious problem that those don't solve each processor knows its connection only in a UNet structure. Namely, the connection is indeterminate state. In this restriction, each processor doesn't know all connections capability except itself. Thus, the protocol $c > 2m + d$ will not display the real fault tolerant capability that we needed. So, this paper uses the protocol $c_i > 2m_i + d_i$ ($for\ 1 \leqq i \leqq n$) to replace previous protocol $c > 2m + d$ for solving fault tolerant capability in a UNet. This protocol was gained by accumulating information of each processor tolerant situation. Thus, the protocol will achieve a common value via corresponded with each processor condition. Similarly, the malicious fault and dormant fault were computed by equation (1) and (2) among the processors. The protocol UNP uses the equation (1) and (2) to compute the fault tolerant capability.

m: the total number of malicious faults

d : the total number of dormant faults

$$m = \sum_{i=1}^{n} \mathrm{m}_i/2 \quad \ldots (\mathit{for}\ 1 \leqq i \leqq n) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

$$d = \sum_{i=1}^{n} d_i/2 \quad \ldots (\mathit{for}\ 1 \leqq i \leqq n) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

According to above equations, the protocol $c_i > 2m_i + d_i$ will be acquired and analyzed of the fault transmission media. The relationships between malicious and dormant faults are analyzed in Table 1.

```
Protocol UNP (For each processor i with initial value $v_i$)

Definition:   n      : the number of processors
              $TM_{ij}$ : the transmission media is connecting with the processor i and processor j.
              v      : the initial message
              $V_i$    : vector
              $c_i$    : the connectivity of processor i
Message Exchange Phase:
Round 1:      Multicast the initial value through ($c_i$) TMs. Receive other processors' (v) values via ($c_i$)
              TMs. Then, construct vector $V_i$.
Round 2:      Multicast the vector ($V_i$).
              Receive the vectors sent by other processors. Then, construct $MAT_i$.
Decision Making Phase:
    Step 1:   Taking the majority value of each row k of $MAT_i$ to be $MAJ_k$.
    Step 2:   If ($\exists MAJ_i = \neg v_i$), then $DEC_i = \phi$;
    Step 3:   Else if ($\exists MAJ_k = ?$) AND ($v_{ki} = v_i$), then $DEC_i = \phi$, else $DEC_i = v_i$ and halt.
```

Figure 1. The proposed protocol UNP to solve consensus problem

Table 1 The fault tolerant number of a UNet

	Meyer & Prahan			UNP		
	$c > 2m + d$			$c_i > 2m_i + d_i$		(1) $m=\sum_{i=1}^{n}\frac{m_i}{2}$ (2) $d=\sum_{i=1}^{n}\frac{d_i}{2}$
n	m	d	c	d_i	c_i	Remark
n = 6	0	4	5	4	5	If m_i = 0 & c= 5, then d_i = 0~4
				3	4,5	If m_i = 0 & c= 4, then d_i = 0~3
				2	3~5	If m_i = 0 & c= 3, then d_i = 0~2
				1	2~5	If m_i = 0 & c= 2, then d_i = 0~1
				0	1~5	If m_i = 0 & c= 1, then d_i = 0
	1	2	5	2	5	If m_i = 1 & c= 5, then d_i = 0~2
				1	4,5	If m_i = 1 & c= 4, then d_i = 0~2
				0	3~5	If m_i = 1 & c= 3, then d_i = 0
	2	0	5			If m_i = 0 & c= 5, then d_i = 0~4

According to the Table 1, this protocol will generate many different results by the vary fault dormant fault and malicious fault. And the malicious fault influences heavily more than the dormant fault in faulty tolerant capability

According to above protocol, we give an example for improving the connection in a UNet structure. This example will use the worst case and best case to explain the different states. The worst case is representing the prior protocols $c > 2m + d$ and $\lfloor(c+1)/2\rfloor$-1. Similarly, the best case we proposed protocol would increase the fault tolerant capability. Thus, our protocol based on the $c_i > 2m_i + d_i$ and the allowable number of faulty TMs are $\left\lfloor \sum_{i=1}^{n}(\lfloor(c_i+1)/2\rfloor-1)\Big/2 \right\rfloor$.

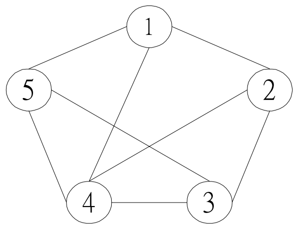

Figure 2.The connectivity of five-processor

Giving an example for executing our protocol and show the structure as Figure 2. Firstly, we don't know any other connectivity state except itself neighbor, like processor 1, it just knows the processor 2, 4, and 5 due to connect each other. Meanwhile, we have $\sum_{i=1}^{5} c_i \big/ 2$ =8 TMs in this case.

In general, each processor just knows his connection of neighbors. Thus, each processor uses the $\lfloor (c_i+1)/2 \rfloor -1$ to compute his connection fault tolerant capability. Subsequently, we will explain the worst case and best as follow.

Worst Case:

If the faulty numbers are focus on the particular processor that has minimum connectivity, likes processor 1 and 3. The computing as follow:

f_t : the total numbers of allowable faulty TMs in a UNet

The number of allowable faulty TMs in processor 1:

$\lfloor (c_1+1)/2 \rfloor -1 = \lfloor (1+1)/2 \rfloor -1 = 1$, $c_i > 2m_i + d_i$

or

The number of allowable faulty TMs in processor 2:

$\lfloor (c_2+1)/2 \rfloor -1 = \lfloor (1+1)/2 \rfloor -1 = 1$, $c_i > 2m_i + d_i$

Thus, the worst case is $f_t = \lfloor (c_\delta +1)/2 \rfloor -1 = 1$, where c_δ is the smallest connectivity of the system.

Best Case:

According to worst case, we can understand the connectivity is a most important factor in faulty tolerant capability. Thus, the faulty numbers of best case will disperse to each processor. Our example is showing in Figure 3(a), the faulty numbers appear in processor 4. The computing as follow:

$$\left\lfloor \sum_{i=1}^{n} \left(\lfloor (c_i+1)/2 \rfloor - 1 \right) \Big/ 2 \right\rfloor = \lfloor (1+1+1+1+1)/2 \rfloor = 2$$

The result shows the fault tolerant capability is 2 more than the prior worst case. It is distinct from worst case and best case. Thus, we can gain the conclusion that all of the case in a UNet environment will be bounded in equation 3.

f_t : the total numbers of allowable faulty TMs in a UNet

c_δ : the smallest connectivity of the system

$$\lfloor (c_\delta +1)/2 \rfloor -1 \leqq f_t \geqq \left\lfloor \sum_{i=1}^{n} \left(\lfloor (c_i+1)/2 \rfloor - 1 \right) \Big/ 2 \right\rfloor \text{......................(3)}$$

According the protocol UNP, the consensus problem is shown in Figure 3. The UNP contains two phases, message exchange phase and decision making phase. In the first round of message exchange, each processor *i* multicast its initial value v_i through connected TM, $1 \leqq i \leqq n$, and then receive other processors initial value as well. If there is no connection exists, replace it as λ. In the second round of message exchange message, and receives the vectors to construct the matrix MAT$_i$. After that, take majority value to construct the matrix MAJ$_i$. The last step is to reach an agreement in the decision making phase.

The messages received by processor *i* in the first round of message exchange are illustrated in Figure 3(b). In second round of message exchange, each processor does the same thing to construct the matrix MAT$_i$ in Figure 3(c). Subsequently, take majority value to construct the MAT$_i$ is in Figure 3(d). Finally, it gain the agreement value in the decision making phase.

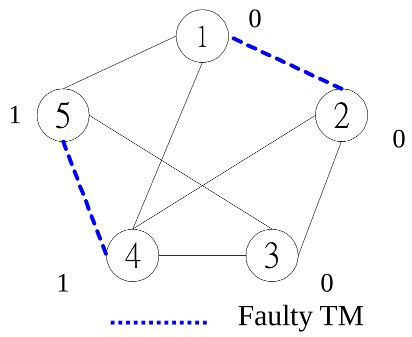

Figure 3(a) A UNet

V$_1$	V$_2$	V$_3$	V$_4$	V$_5$
0	1	λ	0	0
1	0	0	0	λ
λ	0	0	0	0
1	1	1	1	0
1	λ	λ	0	1

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

❑ Messages received by Processor *1*

1	***2***	***3***	***4***	***5***				
0	0	λ	0	0		0		
0	0	0	0	λ		0		
λ	1	λ	0	0	⇨	0	⇨	0
1	0	λ	1	0		?		
1	λ	λ	0	λ		1		

❑ Messages received by Processor *2*

1	***2***	***3***	***4***	***5***				
1	1	λ	0	λ		1		
0	0	0	0	λ	⇨	0		
λ	0	0	0	λ		0	⇨	0
0	1	1	1	λ		1		
0	λ	λ	0	λ		0		

❑ Messages received by Processor *3*

1	***2***	***3***	***4***	***5***				
λ	1	λ	0	0		0		
λ	0	0	0	λ		0		
λ	0	0	0	0	⇨	0	⇨	0
λ	1	1	1	0		1		
λ	λ	λ	0	1		?		

Figure 3 (c). Figure 3 (d)

Construct the matrix MAT$_\mathrm{i}$ Take the majority value

❑ Messages received by Processor *4*

1	***2***	***3***	***4***	***5***				
0	1	λ	0	1		?		
1	0	0	0	λ		0		
λ	0	0	0	1	⇨	0	⇨	0
1	1	1	1	1		1		
1	λ	λ	0	0		0		

❑ Messages received by Processor *5*

1	***2***	***3***	***4***	***5***				
0	λ	λ	1	0		0		
1	λ	0	0	λ	⇨	0		
λ	λ	0	1	0		0	⇨	0
1	λ	1	0	0		?		
1	λ	λ	1	1		1		

Figure 3(c). Figure 3(d)

Construct the matrix MAT$_\mathrm{i}$ Take the majority value

Figure 3. An example of reaching a common agreement in a UNet

4.Correctness and Complexity

The following lemmas and theorems are used to prove the correctness and complexity of protocol UNP.

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

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

Lemma 2: Let the initial value of processor *i* be v_i and the TM_{ij} is healthy (not faulty), then the majority value at the i-th row in MAT_j should be v_i.

Proof: Since TM_{ij} is healthy, the processor *j* will receive v_i from processor *i* in the first round and $v_{ij} = v_i$ in MAT_j. Meanwhile, the value v_i of processor *i* will be broadcasted to the others. There are at most $\lfloor n/2\rfloor$-1 malicious faulty TMs in the system. In the second round, processor *j* receives at least $(n-1)-(\lfloor n/2\rfloor-1) = \lceil n/2\rceil$ v_i's in the i-th row of MAT_j. Hence, there are at least $\lceil n/2\rceil+1$ v_i's in the i-th row, and the majority value in the i-th row should be equal to v_i. ■

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

Proof: By Lemma 2, when TM_{ij} is healthy, the majority value of the i-th row in processor *j* is v_i, for $1\leq j \leq n$. 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 $\lfloor n/2\rfloor$-1 malicious faulty TMs connected with processor *j*, at most $\lfloor n/2\rfloor$-1 values that may be $\neg v_i$'s in the second round. The number of v_i's is $[(n-1)-(\lfloor n/2\rfloor-1)]+1=\lceil n/2\rceil+1$ in the i-th row; 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 $\lfloor n/2\rfloor$-1 malicious faulty TMs. Therefore, in the second round, the total number of $\neg v_i$'s does not exceed $(\lfloor n/2\rfloor-1)+1=\lfloor n/2\rfloor$ and the number of v_i's is at least $[(n-1)-(\lfloor n/2\rfloor-1)]=\lceil n/2\rceil$. If n is an even number, then $\lfloor n/2\rfloor=\lceil n/2\rceil$, the majority of the i-th row in MAT_j cannot be determined. If n is an odd number, then $\lfloor n/2 \rfloor < \lceil n/2 \rceil$. 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 healthy.

Proof: If MAJ_k does not exist or cannot be determined, there are exactly n/2 *v*'s and n/2 $\neg v$'s in the k-th row. Let $v_{ki} = v$ in MAT_i, then, all n/2 $\neg v$'s should be received in the second round. Notably, $\lfloor n/2 \rfloor$-1 malicious faulty TMs are in the system. Therefore, in the second round, processor *i* at least receives a value from processor k without any disturbance. The initial value of processor *k* should disagree with the initial value of processor *i*; hence it is healthy to choose $DEC_i = \phi$

If $v_{ki} = \neg v_i$, we claim that $\neg v_i$ ought to be passed by a faulty TM from processor *k*, and the initial value of processor be $\neg v_{ki} = v_i$.

To prove, if TM_{ki} is healthy, then the initial value of processor *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 *k* was $\neg v_i$, then, by Lemma 3, MAJ_k should be either $\neg v_i$ or cannot be determined for $v_{ki} = v_i$. It is a contradiction. ■

Theorem 1: UNP is valid.

Proof: According to Lemma 1, 2, 3,and 4, the validity of UNP is confirmed. ■

Theorem 2: Protocol UNP can reach a consensus.

Proof:

(1) **Agreement:**

Part 1: If a healthy processor agrees on ϕ, then all healthy processors should agree on ϕ.

If the healthy processor *p* with initial value v_i agrees on ϕ, by Theorem 1, at least there is a healthy processor *k* with initial value $\neg v_i$ in the environment. 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 ? (can not be determined) for $v_{kj} = v_i$. All healthy processors with initial value v_i agree on. Similarly, for the healthy processor *p* with initial value $\neg v_i$, the majority value of the p-th row in MAT_j, $1 \le j \le n$, should be either v_i or cannot be determined with $\neg v_{ij} = v_i$. All healthy processors with initial value $\neg v_i$ agree on ϕ, too.

Part 2: If a healthy processor agrees on v_i, then all healthy processors should agree on v_i.

If the healthy processor *i* with initial value v_i and $DEC_i = v_i$, but there exists some healthy processor *j*,

$j \neq i$, which has $DEC_j \neq v_i$. Therefore, such a situation is impossible. To demonstrate this impossibility, if $DEC_j = \phi$, by Part 1, then $DEC_i = \phi$. This contradicts the above assumption.

If $DEC_j = \neg v_i$, unless the initial value of processor j is $\neg v_i$; otherwise, it is impossible according to the definition of a consensus problem. However, if the initial value of processor j is $\neg v_i$, by Lemma 4, MAJ_i equals to $\neg v_i$ or cannot be determined with $v_{ji} = v_i$ in MAT_i. Then, $DEC_j = \phi$, it is a contradiction. Hence, all healthy processors should agree on the same value by the definition of consensus problem.

(2) Validity:

To prove this case, the initial value of all processors should be the same. If there is a value $\neg v_i$ in MAT_j, $1 \leq j \leq n$, then the value must be attributed to a malicious faulty communication medium. There are at most $\lfloor n/2 \rfloor$-1 malicious faulty communication medias; hence, there is at most $\lfloor n/2 \rfloor$-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 v_i. Therefore, all healthy processors should agree on v_i. ■

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

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

Theorem 4: The number of allowable faulty components of UNP is $\lfloor (c_\delta+1)/2 \rfloor - 1 \leqq f_t \leqq \left\lfloor \sum_{i=1}^{n} (\lfloor (c_i+1)/2 \rfloor - 1) \big/ 2 \right\rfloor$. ■

Proof: According to the prior literatures, we can know the allowable faulty TMs are ($\lfloor n/2 \rfloor$-1), but we can tolerate the number of faulty TMs in $\lfloor (c_\delta+1)/2 \rfloor - 1 \leqq f_t \leqq \left\lfloor \sum_{i=1}^{n} (\lfloor (c_i+1)/2 \rfloor - 1) \big/ 2 \right\rfloor$. ■

5. Conclusion

Byzantine Agreement is a fundamental problem in a distributed environment. These problems are studied by any kinds of the network model in the past. These protocols enlarge faulty tolerant capability gradually. However, it is inflexible in a real network of distributed environment. Each component in the intranet, extranet and Internet can be changed, likes appearing or disappearing. Thus, each processor maybe joins this agreement in any time. In addition, the component in faulty model will also generate different influences via a vary network. Similarly, the protocol must compute tolerant capability again when the network configuration is changed.

Finally, the protocol UNP we proposed will redefine the distributed network environment in an unknowable model and achieve the common value if the condition $c_i > 2m_i + d_i$ are satisfied. Similarly, we prove the equation $\left\lfloor \sum_{i=1}^{n} (\lfloor (c_i + 1)/2 \rfloor - 1) \big/ 2 \right\rfloor$ to increase faulty tolerant capability. Thus, according to our computing, the faulty tolerant capability will be improved. This method is better than prior literatures.

Reference:

[1] A. Bar-Noy, and D.Dolev, "Consensus Algorithms with One-bit Messages," *Distributed Computing*, pp. 205-233.

[2] M. Barborak, M. Malek, and A. Dahbura, "The Consensus Problem in Fault Tolerant Computing," *ACM Computing Surveys*, vol. 25, no. 2, pp.171-220, June 1993.

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

[4] Hin Sing Siu, Yeh Hao Chin, and Wei Pang Yang "A Note on Consensus on Dual Failure Modes," *IEEE Trans. on Parallel and Distributed Systems*, vol. 7, NO. 3, March 1996.

[5] P. Dasgupta, "Agreement under faulty interfaces," *Information Processing Letters*, vol. 65, 1998, pp. 125-129.

[6] D. Dolev, and R. Reischuk, "Bounds on Information Exchange for Byzantine Agreement, " *Journal of ACM*, vol. 32, no. 1, 1985, pp. 191-204.

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

[8] K. Shima, H. Higaki, and M. Takizawa, "Fault-Tolerant Causal Delivery In Group Communication," *in the Proceedings of International Conference of Parallel and Distributed Systems*, 1996, pp. 302 –309.

[9] S. C. Wang, Y. H. Chin, and K. Q. Yan, "Byzantine Agreement in a Generalized Connected Environment Model," *in IEEE Transactions on Parallel and Distributed System*, 6(4), 1995, pp. 420-427.

[10] S.C. Wang, K. Q. Yan., S.H. Kao and L. Y. Tseng, "Consensus with Dual Link Failure Modes on Generalized Network," *in CY journal*, ISSN 1026-244X, Aug. 2000, pp. 35-52.

[11] K. Q. Yan, Y. H Chin, and S. C. Wang " A Optimal Solution for Consensus Problem in an Unreliable Communication System, " *in Proceedings of International Conference on Parallel Processing*, University Park, Pak, pp. 388-391, Aug, 1988.

[12] K. Q. Yan, Y.H. Chin and S. C. Wang, "Optimal Agreement Algorithm in Malicious Faulty Processors and Faulty Links," *IEEE Trans. on Data and Knowledge Engineering*, pp.266-280, 1992.

[13] K. Q. Yan, S. C. Wang and Y.H. Chin, "Consensus Under Unreliable Communication," *Information Processing Letters*, vol. 69, no. 5, March 12. 1999, pp. 243-248.