The Anatomy Study of Fallible Processes Agreement for Cloud Computing

Shu-Ching Wang
Chaoyang University of Technology
scwang@cyut.edu.tw

Shun-Sheng Wang*
Chaoyang University of Technology
sswang@cyut.edu.tw

Kuo-Qin Yan*
Kao Fong College of Digital Contents
kqyann@gmail.com

Chia-Ping Huang
National Cheng Kung University
blue_love520@hotmail.com

*: Corresponding author

Abstract

Cloud computing is Internet-based computing, whereby shared resources, software, and information are provided to users on demand. In general, the users of cloud computing do not own the physical infrastructure, instead avoiding capital expenditure by renting usage from a third-party provider. They consume resources as a service and pay only for resources that they use. However, there are many applications running synchronously in the service platform of cloud computing. The agreement problem is a fundamental issue of reliable distributed systems. Nevertheless, all previous studies of the agreement problem were visited in a network topology with faulty hardware components. However, in a cloud computing, there are a lot of application processes to provide the services of users. In addition, the influence of faulty process is different with the influence of faulty hardware component. Therefore, previous protocols for the agreement problem are not suitable for a cloud computing with fallible processes. To enhance the reliability, the agreement problem in a cloud computing with fallible processes is revisited in this study. The proposed protocol can solve the agreement problem with a minimal number of rounds of message exchange and tolerates a maximal number of faulty processes.

Keywords: Agreement problem; Byzantine agreement; Consensus problem; Interactive consistency; Distributed system; Fault tolerance; Cloud computing.

1. Introduction

Cloud computing is a new concept in distributed systems. It is currently used mainly in business applications in which computers cooperate to perform a specific service together. In addition, the Internet applications are continuously enhanced with multimedia, and vigorous development of the device quickly occurs in the network system [1,3,9,11,14,21]. As network bandwidth and quality outstrip computer performance, various communication and computing technologies previously regarded as being of different domains can now be integrated, such as telecommunication, multimedia, information technology, and construction simulation. Therefore, cloud computing is currently used many commodity nodes that can cooperate to perform a specific service together. Thus, applications associated with network integration have gradually attracted considerable attention.

The users can use the application platform of cloud computing to execute the personal software or program in their capacity of account. In a cloud computing, users can access the operational capability faster with Internet application [14], and the computer systems have the high stability to handle the service requests from many users in the environment. Today, a new application service of operation system is emerged and it changes the user's usage in the past. Originally, the Internet infrastructure is continuous grow that many application services

can be provided in the Internet. In a distributed computing system, components allocated to different places or in separate units are connected so that they may collectively be used to greater advantage [4]. The reliability is improved in a cloud computing by using the low-power hosts. In addition, cloud computing has greatly encouraged distributed system design and application to support user-oriented service applications [11]. Furthermore, many applications of cloud computing can increase user convenience, such as YouTube [14]. Component reliability is one of the most important aspects of cloud computing as it ensures overall reliability and fluency. Thus, the processes in a distributed system must be synchronously completed and all nodes of cloud computing must achieve common agreement. To ensure the cloud computing is reliable, a mechanism to ensure that all nodes can reach an agreed value is thus necessary.

The Internet platform of cloud computing provides many applications for users, just like video, music *et al.* Therefore, each node of a cloud computing needs to run many processes and needs to execute user's requests synchronously. In the cloud computing, each node passes messages through transmission media to other nodes to cooperatively complete user requests. Many users in the cloud computing can execute application services simultaneously. Therefore, the reliability issue of a cloud computing needs to be considered. However, the symptoms of faulty processes can influence the normal operation of a system. The cloud computing system can tolerate the faulty processes in the service environment because the system should respond to user requests quickly and completely the user requests as service. The requisite large number of nodes maybe meet some nodes will be fault to introduce faulty processes into the system. However, the system must allow for the tolerance of faults while maintaining functionality. Simultaneously, in the cloud computing, nodes receive the user's requests maybe influence by the faulty processes. Hence, to remove the affect of faulty processes needs to be mitigated. In a cloud computing, achieving perfect reliability must be accomplished by allowing a given set of nodes to reach a common agreement even in the presence of faulty processes. The agreement problem has been studied in the literature [5,7,10,12,13,15]. The agreement problem is one of the most important issues for designing a reliable distributed system [6,8,12]. Solving the agreement problem, many applications can be achieved [6,8,12]. Therefore, the agreement problem in a cloud computing with faulty processes is revised in this study. The proposed protocol is named **P**rotocol of **F**allible **P**rocesses **A**greement (PFPA in short) and can lead to an agreement of all healthy nodes in a cloud computing.

The rest of this paper is organized as follows. Section 2 discusses the applications of cloud computing and the topology of cloud computing. The related issues of agreement problem are illustrated in Section 3. Then, the proposed protocol PFPA is introduced and illustrated in detail in Section 4. In Section 5, an example of the execution of the proposed protocol is given. Section 6 demonstrates the correctness and complexity of PFPA. Section 7 concludes this paper.

2. Related Work

In previous literatures, the agreement problem has been solved in various network topologies with hardware fault. However, previous studies of the agreement problem [17] do not specifically address the cloud computing with faulty processes to order the application of Internet. Hence, in this section, the applications of cloud computing are illustrated first. Then, the network construction of cloud computing is discussed. Subsequently, three kinds of agreement problem are shown.

NATIONAL CENTRAL LIBRARY ROC

2.1 Practical Applications

Cloud computing is a kind of distributed computing where massively scalable IT-related capabilities are provided to multiple external customers "as a service" using Internet technologies [14]. The cloud providers have to achieve a large, general-purpose computing infrastructure; and virtualization of infrastructure for different customers and services to provide the multiple application services. The ZEUS Company has developed several types of software [17] that can create, manage, and deliver exceptional online services from physical and virtual datacenters or from any cloud environment, such as ZXTM [18] and ZEUS Web Server (ZWS) [20], as shown in Figure 1 [22].

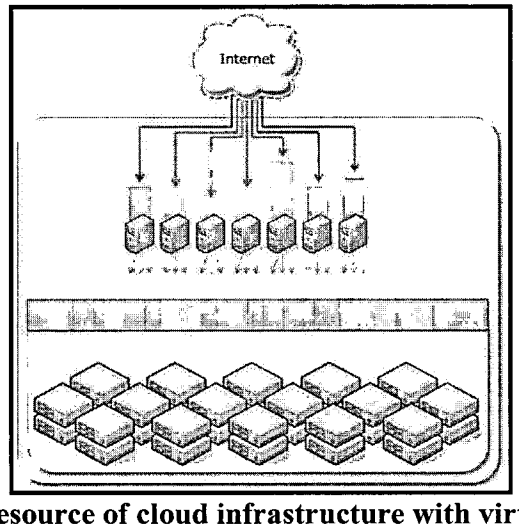


Figure 1 The resource of cloud infrastructure with virtualization [22]

A cloud infrastructure virtualizes large-scale computing resources and packages them up into smaller quantities [22]. Furthermore, the ZEUS Company develops software that can let the cloud provider easily and cost-effectively offer every customer a dedicated application delivery solution [23]. The ZXTM software is much more than a shared load balancing service and it offers a low-cost starting point in hardware development, with a smooth and cost-effective upgrade path to scale as your service grows. The concept is shown in Figure 2.

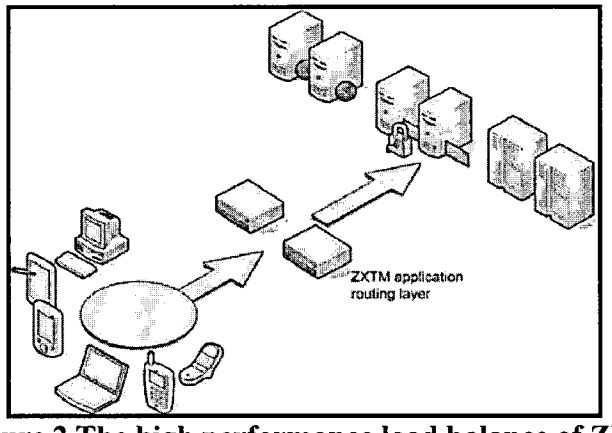


Figure 2 The high performance load balance of ZXTM

The ZEUS provided network framework can be utilized to develop new cloud computing methods [17,23], and is utilized in the current work. In this network composition that can support the network topology of cloud computing used in our study [20,22]. According to the

ZEUS network topology, a network topology of cloud computing is proposed to solve the agreement problem.

2.2 Network Topology

Cloud computing is a new distributed system concept that has been implemented by businesses such as Google [21] and Amazon [16]. Google provides various applications on their Internet platform such as Gmail and YouTube [21]. In addition, Google provides free storage capacity with gigabytes for each user. The big and powerful Google search engine allows users to find multiple results from different file types on the Internet. In previous literature, the agreement problem has been solved in various network topologies. However, previous studies of agreement problems [2,5,6,7,8,10,12,13,15] are not specifically address cloud computing to order the application of Internet. Hence, in this study, the topology of a cloud computing is applied. Subsequently, the agreement problem with fallible processes in the topology of a cloud computing is discussed. Cloud computing is a new distributed system computing concept in which nodes are interconnected with the Internet; the network is assumed reliable and synchronous. Figure 3 is the topology of cloud computing used in this study. The topology is composed of two levels, as follows:

(1) The nodes in A-Level group receive the service requests from users of different types of applications. Therefore, the nodes of A-Level group have higher computational capability than the nodes in B-Level group. In addition, nodes in A-Level group compute enormous amounts data and can communicate with other nodes in the same group directly through transmission media (TM).

(2) Some nodes are formed into a cluster in B-Level group, where each cluster provides a specific application service. According to the properties of nodes, the nodes are clustered to cluster B_i where $1 \le i \le C_{num}$ and C_{num} is the total number of clusters in B-Level group.

(3) For the reliable communication, multiple inter-transmission media (ITM) are used to connect the nodes between A-Level group and B-Level group. In A-Level group, each node must forward the message to all nodes in the corresponding cluster of the B-Level group.

NATIONAL CENTRAL LIBRARY ROC

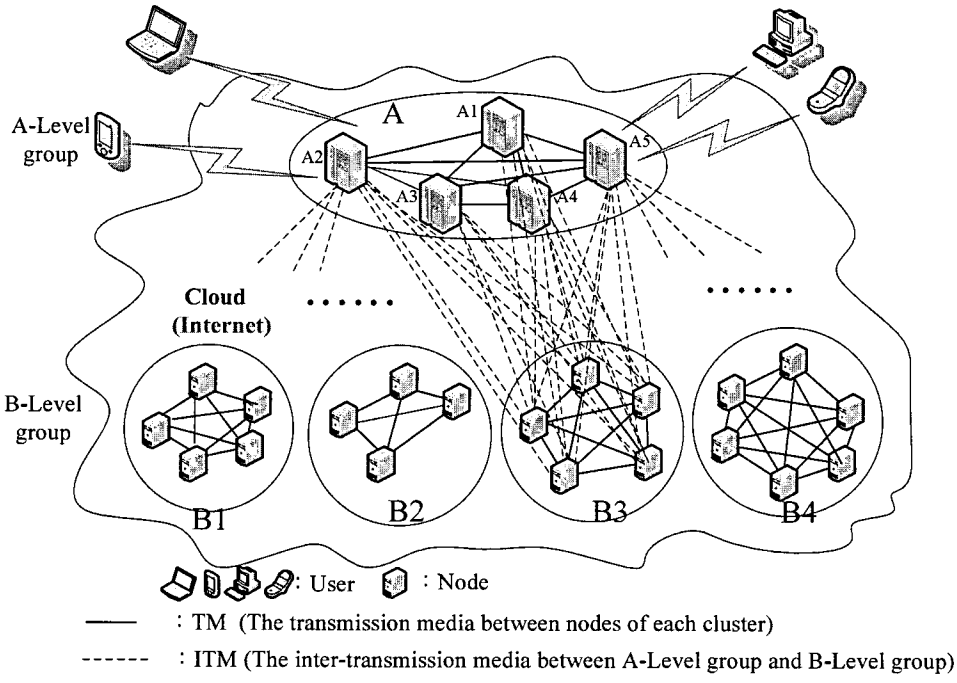


Figure 3 Example of topology of cloud computing

However, a node is said to be healthy if it follows the protocol specifications during the execution of a protocol. The fault symptom of process is called disorderly fault. The behavior of a disorderly fault is the fault can cause other components cannot complete work correctly and synchronously. A disorderly faulty process takes place when a node fails to transmit the fake messages and other nodes receive the same fake messages. However, the behavior of disorderly faulty process is unpredictable and to confuse other nodes to receive incorrect messages to complete a specific service for user. Therefore, the disorderly faulty processes will influence each node in the system finish user's request synchronization. Here, a solution of the agreement problem in the cloud computing with disorderly faulty processes is presented.

3. The Related Issues of Agreement Problem

In order to handle the applications more correctly in the cloud computing, the agreement problem is a very important topic. Simply, the cloud computing must achieve an agreement before any applications executing. Traditionally, the agreement problem is classified into three kinds: Byzantine Agreement (BA) problem, consensus problem and interactive consistency problem.

3.1 Agreement Problems

The BA problem is one of the most fundamental problems concerning reaching agreement in distributed systems [5]. First studied by Lamport, it is a well-known paradigm for achieving reliability in a distributed network of nodes [5]. According to the definition of the BA problem by Lamport:

1) There are n nodes, of which at most $\lfloor (n-1)/3 \rfloor$ nodes could fail without breaking down a workable network;
2) The nodes communicate with each other through message exchange in a fully connected network;

3) The message sender is always identifiable by the receiver;
4) A node is chosen as a source, and its initial value v_s is broadcasted to other nodes and itself to execute the protocol [5].

A closely related sub-problem of the BA problem, the consensus problem, has been studied extensively in the literature [6,8,12]. In this study, the consensus problem is revised in a cloud computing. The consensus problem requires a protocol to allow the components to exchange messages then the healthy components are to achieve consensus. Hence, the consensus is reached if the following constraints are met:

Agreement: All healthy nodes agree on a common value v_i.

Validity: If the initial value of each healthy node is v_i, then all healthy nodes shall agree on the initial value v the source node sends, $v=v_i$.

Another closely related sub-problem, the interactive consistency problem has been studied extensively [2]. The definition of the problem is to make the healthy nodes in an n-node distributed system reach interactive consistency. Each node chooses an initial value and communications with the others by exchanging messages. There is interactive consistency in that each node i has its initial value v_i and agrees on a set of common values. Therefore, interactive consistency has been achieved if the following conditions are met:

Agreement: Each healthy node agrees on a set of common values $V=[v_1,v_2,\ldots,v_n]$.

Validity: If the initial value of each healthy node is v_i, then the i-th value in the common vector V should be v_i.

In this study, the agreement problem with fallible processes in the cloud computing topology is revised. The problem requires all healthy nodes to reach agreement when some of components might be faulty. A distributed system can attain stable results without any influence from faulty components. However, in many cases, the faulty components will influence the system to reach agreement.

However, in this study, the solutions of the interactive consistency problem in A-Level group and the consensus problem in B-Level group are considered. Finally, the service applications of user's request can be completed.

3.2 The Types of Failure Process

The process fault is called the *disorderly fault*. The symptoms of disorderly fault that such fault always sends the constant value and it means process in the node-running overflow or procedure of operation system execute buffering. In the service platform of cloud computing where the nodes have to ensure all applications can be stable provided for users. If a process is in the abnormal state of the specific service cluster then the application cannot to be provided. When the operation system is running unstable or the memory capacity is not enough, or the interrupt of process is happened, then the disorderly faulty process is occurred in the node. However, in this study the disorderly faulty process that the general point is mean the software failure. In this definition, the disorderly faulty process will send the unreliable messages to other nodes and receive the same messages that have been changed. The disorderly fault represents the behavior of a disorderly faulty process is unstable. The healthy process can transmit messages on time or correctly and complete applications synchronously, but the disorderly faulty process may be inconsistent. In other words, the disorderly faulty process cannot send correct messages.

4. The Proposed Protocol

Cloud computing can provide multiple services [9,14,21]. In this study, the agreement problem is revisit in a cloud computing where disorderly faulty processes may influence services provide normally. In this study, a new protocol called **P**rotocol of **F**allible **P**rocesses **A**greement (PFPA in short) is proposed to solve the agreement problem when caused by disorderly faulty processes that may send incorrect messages to influence the cloud computing to reach agreement. When the disorderly faulty processes exist in the cloud computing then two rounds of message exchange required can be estimated to solve the agreement problem. For instance, if the faulty components are disorderly faulty processes, then PFPA can save some rounds required to remove the influence from the disorderly faulty processes. In the cloud computing topology, the main work of A-Level group's nodes is collecting user requests. Each node in A-Level group receives the various requests from users, while the nodes in B-Level group's cluster provide many services for users. Hence, all nodes may receive different initial values different two level groups. The protocol PFPA is executed by nodes in the X-Level groups, where X is the A or B-Level group. Therefore, the interactive consistency problem in A-Level group is discussed first, and then the consensus problem in B-Level group is explained.

In this study, the protocol PFPA is proposed to reach an agreement in a cloud computing. Each node in A-Level group that uses the service request as the initial value executes the PFPA to obtain the common vector DEC_A. Therefore, the PFPA is executed to solve the interactive consistency problem by nodes in A-Level group. After each node of A-Level group has been obtained the common vector value (DEC_A), then each node of the A-Level group forwards the element of vector DEC_A to the nodes in the B-Level group. However, the specific service request can be conformed by the nodes of the same group in the B-Level group.

Each node in the same cluster of B-Level group receives the element from the nodes of A-Level group. In the B-Level group, nodes may receive the fake value by the disorderly faulty processes in A-Level group. The nodes in B-Level group receive the fake value from failure processes of A-Level group through inter-transmission media. Therefore, the total number of allowable faulty processes in A-Level group must be less than the half of the total number of processes in A-Level group.

Sequentially, each node in the same cluster of B-Level group has to take majority value of the received element values (DEC_A). Hence, the initial value of each node can be obtained in the same cluster of B-Level group. Nodes in the same cluster of B-Level group must exchange and receive the initial value with other nodes by executing the *Agreement Process*. Finally, each node takes a majority value to get the DEC_B value. Then the agreement value can be obtained by the PFPA. PFPA is invoked to solve the agreement problem with disorderly faulty processes in cloud computing. Based on the network topology of cloud computing, PFPA can allow each node to transmit messages to other nodes without influence from disorderly faulty processes, the proposed protocol is shown in Figure 4. The PFPA executes the follow steps:

Step 1: The nodes of A-Level group execute the *Agreement Process* to obtain DEC_A (vector value) (for the node i in A-Level group with initial value v_i; $1 \le i \le n_A$, where n_A is the total number of nodes in the A-Level group).

Step 2: Each node of A-Level group sends the specific element of DEC_A to the nodes of cluster in B-Level group that provide the specific application.

Step 3: Each node k in the same cluster of the B-Level group takes a majority value MAJ_k ($1 \le k \le n_{Bj}$, where n_{Bj} is the total number of nodes in the cluster j of B-Level group) of the received elements, then the initial value v_k of each node k can be obtained.

Step 4: The nodes of cluster in B-Level group execute the *Agreement Process* (for the node i in the cluster j of B-Level group with common value v_i; $1 \le i \le n_{Bj}$).

Step 5: Each node of the same cluster in B-Level group takes a majority value from DEC_B, and then the agreement value v (single value) is obtained.

The node in cluster of B-Level group receives the initial value through the PFPA. The *Agreement Process* of PFPA requires two rounds to receive sufficient messages for nodes of A- and B-Level groups. In the first round of *Message Exchange Phase*, each node parallel transmits its initial value to other nodes in the same cluster, then receives the value, and stores it at the first level of its mg-tree. The mg-tree is a tree structure that is used to store the received messages [15]. A detailed description of the mg-tree is presented in Appendix I. Subsequently, each node in the same cluster transmits the received messages to other nodes and stores it at second level in its mg-tree. In the *Decision Making Phase* of *Agreement Process*, each node reorganizes its mg-tree into a corresponding ic-tree. The ic-tree is a tree structure that is used to store a received message without repeated node names [15]. The detailed description of the ic-tree is presented in Appendix II. The function VOTE is applied to the root of each corresponding ic-tree to take the majority value, and then a vector value DEC_A is obtained. Each element of DEC_A is mapped to a specific application that will be executed in the corresponding cluster of B-Level group. Each node of the same cluster in the B-Level group takes a majority value as the initial value from the vector. Sequentially, each node in the same cluster executes the *Message Exchange Phase* of *Agreement Process* and reorganizes its mg-tree into a corresponding ic-tree. Then, the function VOTE is applied to obtain the agreement value. Finally, all healthy nodes in the same cluster are achieved with an agreement value of DEC_B to reach agreement.

PFPA for node i in *X-Level Group*

if $X = A$,
  Do *Agreement Process*(i, n_A, *A-Level group*)
  Send DEC_A to the nodes in the cluster of B-Level Group
If $X=B$,
  Set the initial value v_i of node i = MAJ(v_1, ..., v_k, ..., v_{nx}) where v_k is the value received from the node k in A-Level Group.
  Do *Agreement Process*(i, n_B, *B-Level group*)
  Agree on v = MAJ(DEC_B), end.

Agreement Process(i, n_X, *X-Level group*)

***Message Exchange Phase*:**

$r = 1$, do:

A) Each node i parallel broadcasts its initial value v_i to other nodes in the cluster of an X-Level group.

B) Each node receives and stores the n_X values sent from n_X nodes of the cluster in an X-Level group in the corresponding root of each mg-tree.

for $r = 2$

C) Each node parallel transmits the values at the first level of the corresponding mg-tree to other nodes in the cluster of X-Level group.

D) Each node receives values from other nodes and stores them in the second level of n_X corresponding mg-trees.

***Decision Making Phase*:**

Step 1: Reorganize each mg-tree into a corresponding ic-tree by deleting the vertices with repeated node names.

Step 2: VOTE(i, n_X) function is paralleled to apply to the root of each corresponding ic-tree, then a vector DEC_X as a common value with n_X elements has been obtained.

Function VOTE(i, n_X)

1. val(i), if i is a leaf.
2. The majority value in the set of {VOTE (αi, n_X)$|1 \le i \le n_X$, and vertex αi is a child of vertex α}, if such a majority value exists.
3. A default value ϕ is chosen otherwise.

Function MAJ

Step 1:Count the received values and take the majority, then set a majority value x.

Step 2:If the majority value is not existed, then output a majority value ϕ.

Step 3:Otherwise, output a majority value x. (where $x \in \{0,1\}$)

Figure 4 The proposed protocol PFPA

5. Examples of Executing PFPA

In this section, an example of executing the PFPA is given. In addition, an example of A-Level group is shown in Figure 5-1 and 5-4. The nodes in A-Level group receive service requests. The protocol, for this example, requires two rounds to exchange the messages. Each node in A-Level group can obtain the initial value as shown in Figure 5-2. The different requests are received from different users by each node, such as A1 receives the video service request and A5 receives the blog service request, etc. In each round of the *Message Exchange Phase*, each node parallel transmits the initial value to all nodes in the same cluster and stores the received values in the corresponding root of the mg-tree as shown in Figs. 5-3 and 5-5. Subsequently, in the *Decision Making Phase*, the mg-tree is reorganized into the ic-tree by

deleting the vertices with repeated node names as shown in Figure 5-6. The function VOTE is applied to each corresponding ic-tree of all nodes and then taking the majority value. Eventually, the common vector value DEC_A is obtained for all nodes in A-Level group as shown in Figure 5-7.

All nodes in the cluster of B-Level group receive the element of DEC_A from the nodes of A-Level group by inter-transmission media. All nodes in cluster B1 receive the element of DEC_A that transmits from the nodes in A-Level group for the specific application needing to be serviced. Subsequently, the received element of DEC_A is taken a majority value by each node in cluster B1 as shown in Figure 7.

The example of cluster B1 in B-Level group is presented in Figure 8-1 and 8-4. In this example, there are six nodes in cluster B1 and two rounds of message exchange are required. Figure 8-2 presents the initial value of each node. In the first round of *Message Exchange Phase*, the node sends the initial value (=1) to other nodes and receives the initial value from other nodes in the same cluster as shown in Figure 8-3. The execution of the second round of *Message Exchange Phase* by node B11 is shown in Figure 8-5. In the *Decision Making Phase*, the mg-tree of B13 is reorganized into the corresponding ic-tree as shown in Fig 8-6; and the function VOTE is applied on the ic-tree's root to take the majority value DEC_B, then an agreement value (=1) is obtained as shown in Figure 8-7. Hence, the agreement value has been obtained and all healthy nodes reach agreement.

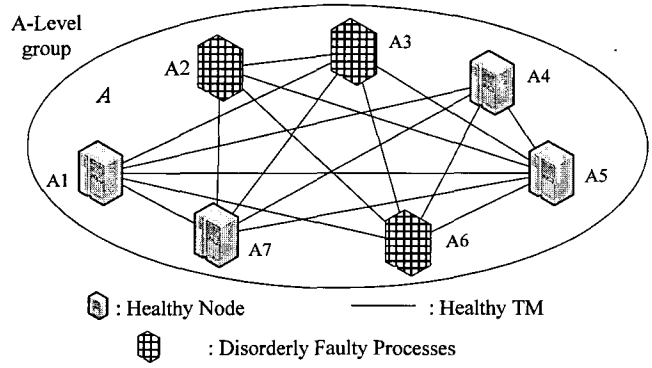


Figure5-1 Example of A-Level group in the 1st round

A1	A2	A3	A4	A5	A6	A7
0	1	0	1	0	1	0

Figure 5-2 The initial value of each node in A-Level group

level 0 Root	*level 1*	
A1	1	0
	2	*0*
	3	*1*
	4	1
	5	0
	6	*0*
	7	0

level 0 Root	*level 1*	
A2	1	0
	2	*0*
	3	*1*
	4	1
	5	0
	6	*0*
	7	0

level 0 Root	*level 1*	
A3	1	0
	2	*0*
	3	*1*
	4	1
	5	0
	6	*0*
	7	0

level 0 Root	*level 1*	
A4	1	0
	2	*0*
	3	*1*
	4	1
	5	0
	6	*0*
	7	0

level 0 Root	*level 1*	
A5	1	0
	2	*0*
	3	*1*
	4	1
	5	0
	6	*0*
	7	0

level 0 Root	*level 1*	
A6	1	0
	2	*0*
	3	*1*
	4	1
	5	0
	6	*0*
	7	0

level 0 Root	*level 1*	
A7	1	0
	2	*0*
	3	*1*
	4	1
	5	0
	6	*0*
	7	0

Figure 5-3 The mg-tree of each node in the A-Level group at the 1st round

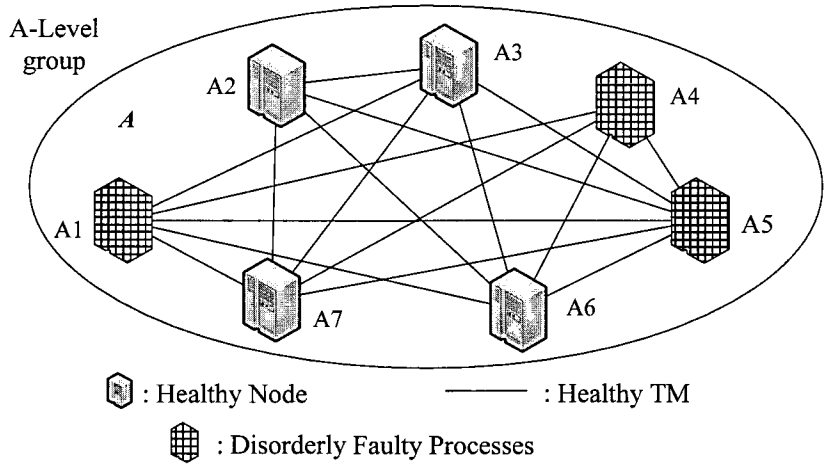


Figure 5-4 Example of A-Level group in the 2nd round

level 0 Root	*level 1*	*level 2*	
A1	Val(1)= 0	11	*1*
		12	0
		13	0
		14	*0*
		15	*1*
		16	0
		17	0
	Val(2)= *0*	21	*1*
		22	0
		23	0
		24	*0*
		25	*1*
		26	0
		27	0
	Val(3)= *1*	31	*1*
		32	1
		33	1
		34	*0*
		35	*1*
		36	1
		37	1
	Val(4)= 1	41	*1*
		42	1
		43	1
		44	*0*
		45	*1*
		46	0
		47	1
	Val(5)= 0	51	*1*
		52	0
		53	0
		54	*0*
		55	*1*
		56	0
		57	0
	Val(6)= *0*	61	*1*
		62	0
		63	0
		64	*0*
		65	*1*
		66	0
		67	0
	Val(7)= 0	71	*1*
		72	0
		73	0
		74	*0*
		75	*1*
		76	0
		77	0

Figure 6-5 The mg-tree of each node in the A-Level group at the 2nd round

level 0 Root	*Level 1*	*level 2*	
A1	Val(1)= 0		
		12	0
		13	0
		14	*0*
		15	*1*
		16	0
		17	0
	Val(2)= *0*	21	*1*
		23	0
		24	*0*
		25	*1*
		26	0
		27	0
	Val(3)= *1*	31	*1*
		32	1
		34	*0*
		35	*1*
		36	1
		37	1
	Val(4)= 1	41	*1*
		42	1
		43	1
		45	*1*
		46	0
		47	1
	Val(5)= 0	51	*1*
		52	0
		53	0
		54	*1*
		56	0
		57	0
	Val(6)= *0*	61	*1*
		62	0
		63	0
		64	*0*
		65	*1*
		67	0
	Val(7)= 0	71	*1*
		72	0
		73	0
		74	*0*
		75	*1*
		76	0

Figure 6-6 The ic-tree of each node in A-Level group

Level 1	*level 0 Root*
VOTE(1) =(0,0,*0*,*1*,0,0)= 0	VOTE(A1) =0,*0*,*1*,1,0,*0*,0
VOTE(2) =(*1*,0,*0*,*1*,0,0)= 0	
VOTE(3) =(*1*,1,*0*,*1*,1,1)= 1	
VOTE(4) =(*1*,1,1,*1*,1,1)= 1	
VOTE(5) =(*1*,0,0,*0*,0,0)= 0	
VOTE(6) =(*1*,0,0,*0*,*1*,0)= 0	
VOTE(7) =(*1*,0,0,*0*,*1*,0)= 0	

Figure 6-7 The agreement value of node A3

B11	B12	B13	...	B18
A1 1	A1 1	A1 1	...	A1 1
A2 1	A2 1	A2 1	...	A2 1
A3 ***0***	A3 ***0***	A3 ***1***	...	A3 ***0***
A4 ***1***	A4 ***0***	A4 ***0***	...	A4 ***0***
A5 1	A5 1	A5 1	...	A5 1
A6 ***0***	A6 ***0***	A6 ***0***	...	A6 ***0***
A7 1	A7 1	A7 1	...	A7 1
MAJ_1=1	MAJ_2=1	MAJ_3=1	...	MAJ_8=1

Figure 7 Each node of B1 cluster receives the element of DEC_A from A-Level group

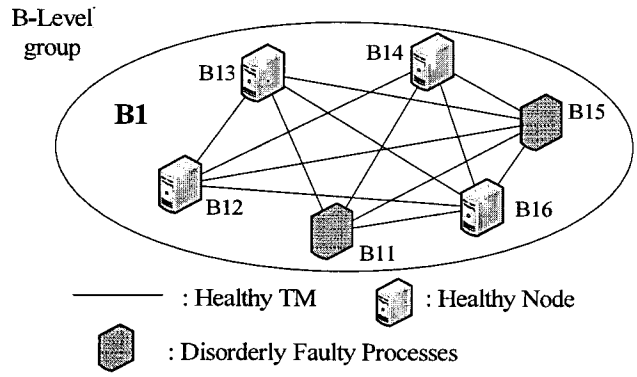


Figure 8-1 Example of B1 cluster in the B-Level group

B11	B12	B13	B14	B15	B16
1	1	1	1	1	1

Figure 8-2 The initial value of each node in B1 cluster

level 0 Root	*level 1*	*level 0 Root*	*level 1*	*level 0 Root*	*level 1*	*level 0 Root*	*level 1*	*level 0 Root*	*level 1*	...
Val(B11) =1	1 *0*	Val(B12) =1	1 *0*	Val(B13) =1	1 *0*	Val(B14) =1	1 *0*	Val(B15) =1	1 *0*	...
	2 1		2 1		2 1		2 1		2 1	...
	3 1		3 1		3 1		3 1		3 1	...
	4 1		4 1		4 1		4 1		4 1	...
	5 *0*		5 *0*		5 *0*		5 *0*		5 *0*	...
	6 1		6 1		6 1		6 1		6 1	...

Figure 8-3 The mg-tree of each node in B1 cluster at the 1st round

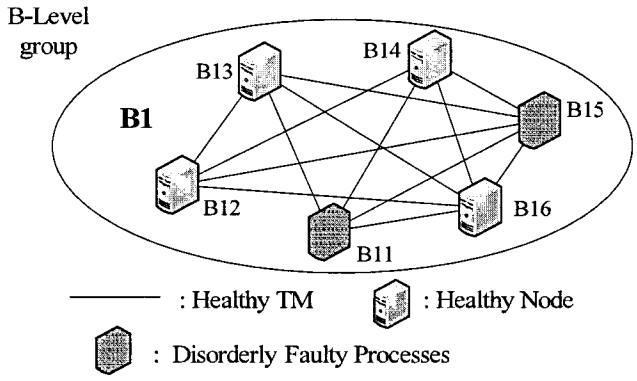


Figure 8-4 Example of cluster B1 in the 2nd round

level 0 Root	*level 1*	*level 2*	
B13	Val(1)= *0*	11	1
		12	1
		13	*0*
		14	1
		15	1
		16	*0*
	Val(2)= 1	21	1
		22	1
		23	*0*
		24	1
		25	1
		26	*0*
	Val(3)= 1	31	1
		32	1
		33	*0*
		34	1
		35	1
		36	*0*
	Val(4)= 1	41	1
		42	1
		43	*0*
		44	1
		45	1
		46	*0*
	Val(5)= *0*	51	1
		52	1
		53	*0*
		54	1
		55	1
		56	*0*
	Val(6)= 1	61	1
		62	1
		63	*0*
		64	1
		65	1
		66	*0*

Figure 8-5 The mg-tree of node B13 at the 2nd round

The ic-tree erased the vertices with repeated names from the mg-tree

level 0 Root	*level 1*	*level 2*	
B13	Val(1)= *0*		
		12	1
		13	*0*
		14	1
		15	1
		16	*0*
	Val(2)= 1	21	1
		22	1
		23	*0*
		24	1
		25	1
		26	*0*
	Val(3)= 1	31	1
		32	1
		33	*0*
		34	1
		35	1
		36	*0*
	Val(4)= 1	41	1
		42	1
		43	*0*
		45	1
		46	*0*
	Val(5)= *0*	51	1
		52	1
		53	*0*
		54	1
		56	*0*
	Val(6)= 1	61	1
		62	1
		63	*0*
		64	1
		65	1

Figure 8-6 The ic-tree of node B13

level 1	*level 0 Root*
VOTE(1) =(1,0,1,1,0)= 1	VOTE(B1)=(1,1,1,1,1,1)= 1
VOTE(2) =(1,0,1,1,0)= 1	
VOTE(3) =(1,1,1,1,0)= 1	
VOTE(4) =(1,1,0,1,0)= 1	
VOTE(5) =(1,1,0,1,0)= 1	
VOTE(6) =(1,1,0,1,1)= 1	

Figure 8-7 The agreement value of node B13

6. The Correctness and Complexity

According to the literature, a protocol is obtained and the following proofs for the agreement and validity property are given in this section. The following lemmas and theorems are used to prove the correctness and complexity of the **P**rotocol of **F**allible **P**rocesses **A**greement (PFPA in short). The notations of PFPA proving are shown as follows:

Notation	Description
n:	The number of nodes in the cloud computing.
TM_{ij}:	The transmission media between node i and node j.
ITM:	The inter-transmission media between A-level group and B-level group.
c:	The connectivity of network topology.
c_A:	The connectivity in A-Level group.
c_{Bj}:	The connectivity in the cluster j of B-Level group.
ITM_{Bj}:	The connectivity with each node of the j cluster in B-Level group between A-Level group.
ITM_{Bjc}:	The connectivity with each node in the cluster j of B-Level group.
n_A:	The number of nodes in A-Level group.
n_{Bj}:	The number of nodes in the cluster j of B-Level group.
Nfp_A:	The number of allowable disorderly faulty processes in A-Level group.
Nfp_{Bj}:	The number of allowable disorderly faulty processes in the cluster j of B-Level group.
Nfp:	The number of allowable disorderly faulty processes in the cloud computing.
tf_A:	The number of allowable disorderly faulty processes in A-Level group.
tf_B:	The number of allowable disorderly faulty processes in B-Level group.
T_f:	The total number of allowable disorderly faulty processes.
σ:	The number of rounds required in the *Agreement Process*.

6.1 Correctness of PFPA

To prove that vertex α is common; the term common frontier [2] is defined as follows: When every root-to-leaf path of the mg-tree contains a common vertex, the collection of the common vertices forms a common frontier. In addition, the constraints, Agreement and Validity, can be rewritten as:

- *Agreement*: Root i is common
- *Validity*: VOTE(i)=v_i for each healthy node, if the node i is healthy

Every healthy node has the same values collected in the common frontier if a common frontier does exist in a healthy node's mg-tree. Subsequently, using the same function VOTE

to compute the root value of the tree structure, every healthy node can compute the same root value because the same input (the same collected values in the common frontier) and the same computing function will produce the same output (the root value). Since PFPA can solve the agreement problem, the correctness of PFPA should be examined in the following two ways [2].

(1)Correct vertex: Vertex αi of a tree is a correct vertex if node i (the last node name in vertex αi's node name list) is healthy. In other words, a correct vertex is a place to store the value received from a healthy node.

(2)True value: For a correct vertex αi in the tree of a healthy node, val(αi) is the true value of vertex αi if TM_{ij} is fault-free. In other words, a correct vertex is a place to store the value received from a healthy node. In other words, the stored value is called the true value of a vertex if the value stored in such a vertex is correct from the influence of a faulty transmission media.

Lemma 1. The healthy destination node can detect the influence of the values through disorderly faulty processes.

Proof: The message(s) send by disorderly faulty processes can be detected if the sending node sends the same values that are not following the initial value passed.

Lemma 2. The healthy nodes can receive message from healthy node, if the number of c_A and c_{Bj} and ITM_{Bj} is maximal.

Proof: A healthy sender node broadcasts a message to others and itself. In the worst case, a healthy node can receive c_A-Nfp_A, c_{Bj}-Nfp_{Bj} and ITM_{Bjc}-Nfp_A messages transmitted in each round of the message exchange because the disorderly faulty processes can be detected. If c_A-Nfp_A>2Nfp_A, c_{Bj}-Nfp_{Bj}>Nfp_{Bj} and $\sum_{Bj}\lceil ITM_{Bj}/2\rceil$-1≥[$ITM_{Bjc}$≥ (2$Nfp_A$+1)], a healthy node can determine messages from sender nodes by taking the majority value from the values received in each message exchange.

Theorem 1. The influences from disorderly faulty processes can be removed by a healthy node.

Proof: By Lemma 1 and Lemma 2, the theorem is proved.

Theorem 2. Each node can receive the values without influences of any disorderly faulty processes from the sender node via PFPA in each round, then n_A>2Nfp_A+1 in A-Level group and n_{Bj}> $\max_{j=1}^{C_{num}}$ 2Nfp_{Bj}+1 in the cluster j of B-Level group.

Proof: The influences of disorderly faulty processes between any pairs of nodes can be ignored in each round of message exchange and n_A>2Nfp_A+1 in A-Level group; and n_{Bj}>2Nfp_{Bj}+1 in the cluster j of B-Level group. The reason is that the healthy sender nodes n_A (n_{Bj}) copies of message to all destination nodes. In the worst case, a healthy destination node receives n_A-Nfp_A messages transmitted via the healthy sender node in A-Level group; and receives n_{Bj}-Nfp_{Bj} messages transmitted via the healthy sender node in the cluster j of B-Level group.

Theorem 3. The healthy node can detect the disorderly faulty processes in the network.

Proof: In the proposed protocol PFPA, there are two rounds of message exchange in *Agreement Process*, where Nfp≥(⌊(n-1)/3⌋) and n>3, so there are two rounds of message exchange in the *Message Exchange Phase*. Each healthy node receives the message from the sending node in the first round of message exchange and receives other nodes messages in the second round of message exchange. In terms of the Lemma 1, each healthy node can detect the disorderly faulty processes in the cloud computing.

Lemma 3. In an ic-tree, all correct vertices are common.

Proof: The tree structure has conversed from mg-tree to ic-tree. At the level σ or upon of ic-tree, the correct vertex i has at least 2σ-1 children, in which at least σ children are correct. The real value of these σ correct vertices is common, and the majority value of

vertex α is common. For this reason, all correct vertices of the ic-tree are common.

Lemma 4. The common frontier exists in the ic-tree.

Proof: There are σ vertices along each root-to-leaf path of an ic-tree, so that though most σ-1 nodes have failed, at least one vertex is correct along each root-to-leaf path of the ic-tree. The correct vertex is common, and the common frontier exists in each healthy node ic-tree by Lemma 1.

Lemma 5. Let α be a vertex, and α is common if there is a common frontier in the sub-tree rooted at *i*.

Proof: When the height of α is 0, and the common frontier exists, α is common. If the height of α is σ, the children of α are all in common by induction hypothesis with the height of the children at σ-1. Then the vertex α is common.

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

Theorem 4. The root of a healthy node's ic-tree is common.

Proof: By Lemmas 1, 2 and the Corollary, the theorem is proved.

Theorem 5. The proposed protocol PFPA solves the agreement problem in a cloud computing.

Proof: Inasmuch as the theorem must be described that PFPA meets the constraints *Agreement'* and *Validity'*.

Agreement': Root *i* is common, and by Theorem 3, *Agreement'* is satisfied.

Validity': VOTE(i)=v_i for each healthy node, if the initial value of the node i is v_i.

Whereas all nodes are healthy, the nodes use PFPA to communicate with all others. The message of correct vertices for all healthy nodes' mg-trees is v_i. When the tree structure has converted from mg-tree to ic-tree, the correct vertices still exist. Therefore, every correct vertex of the ic-tree is common (refer to Lemma 4), and its true value is v_i. This root is common by Theorem 4. The computed value VOTE(i)=v_i is stored in the root of the ic-tree for all healthy nodes. (Validity') is satisfied.

6.2. Complexity of PFPA

The complexity of PFPA is evaluated in terms of: 1) the maximum number of allowable disorderly faulty processes; and 2) the minimum number of rounds to exchange messages. Theorems 6 and 7 show that the optimal solution is reached.

Theorem 6. The number of allowable disorderly faulty processes is T_f.

Proof: According to the past literatures of the agreement problem, the influence of disorderly faulty processes is similar as faulty transmission media; hence, the constraint of the maximum number of allowable faulty (n>2Nfp+1) can be applied to our study.

In a cloud computing, PFPA can tolerate tf_A ($\geq 2Nfp_A+1$) disorderly faulty processes in A-Level group and the fault tolerant capability of B-Level group is tf_B ($\geq \max_{j=1}^{C_{num}} 2Nfp_{Bj}+1$). The total number of allowable disorderly faulty processes by PFPA is T_f ($\geq (2Nfp_A+1)+ \max_{j=1}^{C_{num}} (2Nfp_A+1)$), and the number of disorderly faulty processes is maximal in the cloud computing.

Theorem 7. PFPA requires σ rounds of message exchange to solve the agreement in a cloud computing and σ is minimum number of rounds.

Proof: The message passing is required only in the *Message Exchange Phase*; two rounds are used to send the sufficient messages to achieve agreement in an n-nodes distributed system [15]. In a cloud computing, each node needs to exchange messages with other nodes. Therefore, the constraint of the minimum number with two rounds can be applied to the study. However, in a cloud computing, two rounds of exchange messages in the A and B-level group are required. In addition, each node in the same cluster of B-Level group needs to receive messages from A-Level group's nodes; therefore, one

round is required. In conclusion, the minimum number rounds to exchange message required is optimal.

As a result, PFPA requires a minimal number of rounds and tolerates a maximal number of disorderly faulty processes to reach a common agreement with all healthy nodes. The optimality of the protocol is proven.

7. Conclusions

Cloud computing is a new concept of distributed systems [1,3,9,11,14,21]. It has greatly encouraged distributed system design and practice to support user-oriented services with application [3,9,14,21]. In the Internet platform of cloud computing where each node needs to complete the user's requests synchronously and to reach the common agreement as specific service. Fault-tolerance is an important research topic in the study of distributed systems and it is a fundamental problem in distributed systems; there are many relative literatures in the past [2,5,6,7,8,10,12,13,15]. According to previous studies, network topology plays an important role in the agreement problem, but the results cannot cope with a cloud computing with fallible processes and the agreement problem thus needs to be reinvestigated. Moreover, in this study, the agreement problem with disorderly faulty processes in a cloud computing has been solved by the proposed protocol.

The proposed protocol, **P**rotocol of **F**allible **P**rocesses **A**greement (PFPA in short), ensures that all healthy nodes in the cloud computing can reach a common value. Moreover, the new protocol PFPA is adapted to the cloud computing and the solution of PFPA is applied to a cloud computing with fallible processes. Nevertheless, the interactive consistency problem in A-Level group and the consensus problem in B-Level group have been solved. PFPA can derive the bound of allowable disorderly faulty processes. PFPA uses the minimum number of rounds of message exchange and tolerates the maximum number of allowable disorderly faulty processes in a cloud computing. Furthermore, the fault-tolerance capacity is enhanced by PFPA.

Appendix

I. The Message Gathering Tree (mg-tree)

Each healthy node maintains such an mg-tree during the execution of PFPA. In the first round of *Message Exchange Phase* in *Agreement Process*, the node i broadcasts its initial value to other nodes. When the healthy node receives the value sent from the sending node i, it stores the received value, denoted as val(i) at the root of its mg-tree, as shown in Figure 5-3. In the second round of *Message Exchange Phase* in *Agreement Process*, each node multicasts the value stored in the root of the mg-tree to all nodes. If the node j sends message val(i) to node k, then node k stores the received value from node j, denoted as val(ij), in vertex ij of its mg-tree, shown in Figs. 5-4. The val(αk) shows that the message is sent to a series of receivers, denoted as α, and that node k is the latest receivers. For instance, the message val($ij\ldots\ k$) is stored in the vertex $ij\ldots\ k$ of an mg-tree, which implies that the message just received was sent through the node i, node j, .., and that node k (the node k is the latest node to pass the message). In addition, it is denoted as val(αk). When the message is transmitted through a node more than once, the name of the node will be repeated correspondingly. For example, val(11), stored in vertex 11 of Figure 5-4, indicates that the message is sent from node A1, then through node A1 again.

In summary, the root of an mg-tree is always named i to denote that the stored message is sent from the node i in the first round, and the vertex of an mg-tree is labeled by a list of node

names. The node name list contains the names of the nodes through which the stored message was transferred.

Appendix II. The Information-Collecting tree (ic-tree).

An ic-tree is reorganized from a corresponding mg-tree by removing the vertices with repeated node names in order to reduce the influence from a faulty node repeatedly in an ic-tree. In Figure 6-6, we show an example of reorganizing an mg-tree to an ic-tree by deleting the repeated node names of mg-tree.

Acknowledgment

This work was supported in part by the Taiwan National Science Council under Grants NSC96- 2221-E-324-021 and NSC97-2221-E-324–007 -MY3.

Reference

[1] Aymerich, F.M., Fenu, G., and Surcis, S., "An Approach to a Cloud Computing Network," the ***First International Conference on the Applications of Digital Information and Web Technologies***, pp. 113-118, August 2008.

[2] Fischer, M., and Lynch, N., "A Lower Bound for the Assure Interactive Consistency," ***Information Processing Letters***, Vol. 14, No.4, pp. 183-186, 1982.

[3] Grossman, R.L., Gu, Y., Sabala, M., and Zhang, W., "Compute and Storage Clouds Using Wide Area High Performance Networks," ***Future Generation Computer Systems***, Vol. 25, No. 2, pp. 179-183, February 2009.

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

[5] Lamport, L., Shostak, R., and Pease, M., "The Byzantine General Problem," A***CM Transactions on Programming Language and Systems***, Vol. 4, No. 3, pp. 382-401, July 1982.

[6] Meyer, F.J., and Pardhan, D.K., "Consensus with Dual Failure Modes," ***IEEE Transactions on Parallel and Distributed System***, Vol. 2, No. 2, pp. 214-222, 1991.

[7] Pease, M., Shostak, R., and Lamport, L., "Reaching Agreement in Presence of Faults," ***Journal of ACM***, Vol. 27, No. 2, pp. 228-234, April 1980.

[8] Siu, H.S., Chin, Y.H., and Yang, W.P., "A Note on Consensus on Dual Failure Modes," ***IEEE Transactions on Parallel and Distributed System***s, Vol. 7, No. 3, pp. 225-230, 1996.

[9] Vouk, M.A., "Cloud Computing- Issues, Research and Implementations," ***Information Technology Interfaces***, pp. 31-40, June 2008.

[10] Wang, S.C., Chin, Y.H., Yan, K.Q., and Chen, C., "Achieving Byzantine Agreement in a Generalized Network Model," ***CompEuro '89***, Vol. 4, pp. 139-145, 1989.

[11] Wang, L.H., Tao, J., and Kunze, M., "Scientific Cloud Computing: Early Definition and Experience," the ***10th IEEE International Conference on High Performance Computing and Communications***, pp. 825-830, 2008.

[12] Wang, S.C., and Yan, K.Q., "Revisit Consensus Problem on Dual Link Failure Modes," the ***International Computer Software & Applications Conference***, pp. 84-89, August 1998.

[13] Wang, S.C., Yan, K.Q., Wang, S.S., and Zheng, G.Y., "Reaching Agreement Among Virtual Subnets in Hybrid Failure Mode," ***IEEE Transactions on Parallel and Distributed Systems***, Vol. 19, No. 9, pp. 1252-1262, September 2008.

[14] Weiss ,A., "Computing in The Clouds," ***netWorker***, Vol. 11, No. 4, pp. 16-25, 2007.

NATIONAL CENTRAL LIBRARY ROC

[15] Yan, K.Q., Chin, Y.H., and Wang, S.C., "Optimal agreement protocol in malicious faulty processors and faulty links," ***IEEE Transactions on Knowledge and Data Engineering***, Vol. 4, No. 3, pp. 266-280, June 1992.
[16] "Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more," http://www.amazon.com/, January 2010.
[17] "Application Delivery Networking, Application Acceleration, Internet Traffic Management System: Zeus.com," http://www.zeus.com/, January 2010.
[18] "Application Traffic Management, Application Security," http://www.zeus.com/products/traffic-manager/index.html, January 2010.
[19] "Cloud Computing," http://www.zeus.com/cloud_computing/, January 2010.
[20] "Load Balancing, Load Balancer," http://www.zeus.com/products/zxtmlb/index.html, January 2010.
[21] "More Google Product," http://www.google.com/options/, January 2010.
[22] "What is Cloud Computing?," http://www.zeus.com/cloud_computing/cloud.html, January 2010.
[23] "ZXTM for Cloud Hosting Providers," http://www.zeus.com/cloud_computing/for_cloud_providers.html, January 2010.