JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 25, 1267-1281 (2009)

Short Paper

Investigation of Consensus Problem over Combined Wired/Wireless Network

CHIEN-FU CHENG, SHU-CHING WANG+ AND TYNE LIANG
Department of Computer Science
National Chiao Tung University
Hsinchu, 300 Taiwan
+*Graduate Institute of Informatics*
Chaoyang University of Technology
Taichung, 413 Taiwan

Wireless networks have become ubiquitous, making combined wired/wireless network a popular trend of development in nowadays. Therefore, the Consensus problem in combined wired/wireless network is an important topic. Over the past few years, a considerable number of studies have been made on pure wired networks. However, no studies have ever tried to solve the Consensus problem in combined wired/wireless networks. In order to meet the characteristics of combine wired/wireless networks (the limited resources have made the computation ability of mobile processors often weaker than that of stationary processors) and reduce the number of rounds of message exchange required, most of the communications and computation overhead must be fulfilled within by the consensus-servers. Therefore, we introduce a hierarchical concept in our system model. Only consensus-servers need to exchange messages and compute the common value. In this paper, we will investigate the Consensus problem in combined wired/wireless network to enhance fault-tolerance and reliability. Besides, we also prove our protocol is able to tolerate a maximum number of allowable faulty components with minimum rounds of message exchange required.

Keywords: consensus, byzantine agreement, fault-tolerance, dormant fault, malicious fault, combined wired/wireless network, secure communication

1. INTRODUCTION

The Byzantine Agreement (BA) [10, 11, 15, 17] and Consensus [1, 8, 14, 17] are two closely related fundamental problems in agreement on a common value in distributed system. The BA problem was first described and solved by Pease, Shostak, and Lamport [11]. In the BA problem, there are n ($n \geq 4$) processors in the network, where one processor is designated as the commander that holds an initial value v_s. The commander first sends the initial value v_s to all other processors. On receipt of the value v_s, each processor (without the commander) exchanges the received value with other processors. In addition, there is an adversary that controls up to p_m ($n \geq 3p_m + 1$) of the processors and can arbitrarily deviate from the designated protocol specification. After $\lfloor (n - 1)/3 \rfloor + 1$ rounds of message-exchange, a common value can be obtained. More precisely, the BA problem is

Received August 15, 2007; revised January 7 & March 3 & April 22, 2008; accepted May 8, 2008.
Communicated by Chin-Laung Lei.

defined by the two properties: (1) Agreement: All fault-free processors agree on a common value. (2) Validity: If the source (commander) processor is fault-free, then all fault-free processors agree on the initial value that the source processor sends. That is, only one processor has initial value in the BA problem. After executing BA protocol, each fault-free processor can make the same decision (common value), the common value is necessary to be selected from the initial value or the default value (the default value is predefined).

The difference between the BA problem and Consensus problem is that each processor has its own initial value in the Consensus problem [17]. Hence, the Consensus problem is defined by these two properties: (1) Consensus: All fault-free processors agree on a common value; (2) Validity: If the initial value of all processors is v_i, then all fault-free processors shall agree on v_i. That is, each processor has its own initial value and the initial value of each processor may be different in the Consensus problem. After executing the Consensus protocol, each fault-free processor can make the same decision (common value), so the common value is necessary to be selected from one of the initial values or the default value (the default value is predefined). In view of the definition of the initial value and the common value, the Consensus problem is solved if n copies of the BA protocol are run in parallel [17]. Through these properties, it can be clearly understood that the BA problem is a special case of the Consensus problem in which only one processor's initial value is of interest. A protocol for reaching a Consensus can also solve the BA problem. Hence, this paper focuses on the Consensus problem.

The Consensus problem in wired networks has been extensively studied over the past two decades. Many graceful Consensus protocols have been proposed with various network structure assumptions and different symptoms of faulty components assumptions. Examples are Consensus protocols for fully connected network with malicious faulty processors [17], broadcast network with malicious faulty processors and malicious faulty communication links [1] and un-fully connected network with hybrid (dormant and malicious) faulty processors [8, 14]. The detailed description of the symptoms of faulty components is presented in section 2.1.

Since wireless networks and mobile computing are becoming ubiquitous, the future network environments will consist of both wired and wireless networks and provide support for mobile computing. We know that the physical topology of a wired network is static, but the physical topology of a wireless network is dynamic. Hence, previous Consensus protocols are not applicable in a combined wired/wireless network.

Communication overhead in the Consensus protocol is another issue that we must not ignore. Due to the large communication overhead, it is difficult, or even impossible, to realize the Consensus protocol [7]. In this paper, we introduce a hierarchical concept in our system model to reduce communication overhead in the Consensus protocol.

In addition, mobile processors exhibit some specific aspects that must be addressed so as to design an efficient system. A crucial issue in the design and development of mobile processors is power consumption, as batteries have great influence on the weight, size and reliability of mobile processors [13]. That is why the bandwidth and computation ability of mobile processors are often weaker than that of stationary processors. We designed a transmission protocol "Secure Relay Fault-tolerance Channel" (SRFC), which is more efficient in increasing battery lifetime and provides secure communications between processors. SRFC is also applicable for stationary processors.

This paper is intended to investigate the Consensus problem in combined wired/ wireless network. Two protocols are proposed. The first is the transmission protocol SRFC. The second is the Consensus protocol "Client-initiated Consensus Protocol" (CCP). This paper is organized as follows. Section 2 provides the conditions for Consensus. Section 3 introduces concept and approaches. Section 4 dissects the correctness and complexity of proposed protocols. Finally, conclusions are induced in section 5.

2. THE CONDITIONS FOR CONSENSUS

To design a Consensus protocol, certain conditions must be taken into account. They are symptoms of faulty components, the system model, properties of the Consensus problem and the constraint of allowable faulty consensus-servers.

2.1 The Symptoms of Faulty Components

The symptoms of a faulty component (processor [11] or communication link [12, 17]) include crash, omission and Byzantine. (1) Crash failure: a faulty component stops executing prematurely and does nothing afterward. (2) Omission failure: a faulty component omits to send or receive messages. (3) Byzantine failure: a faulty component can exhibit any behavior whatsoever [4, 9, 11]. (*e.g.*, suffer benign failures, send bogus values in messages, send messages at the wrong time, send different messages to different processors and work in coordination with other faulty processors to prevent fault-free processors from reaching a common value). We may divide the above faulty component symptoms into two categories. They are dormant fault (crash failure and omission failure) and malicious fault (Byzantine failure) [8].

2.2 The System Model

In recent years, the bandwidth and quality of wireless networks has been drastically improved. Therefore, wireless network has become more and more popular [3], resulting in the development of nowadays network from wired or wireless network to combined wired/wireless network. We know that the communication overhead of the Consensus protocol is inherently large [7]. Previous Consensus protocols were designed for flat networks [8, 14, 17]. In a flat network, all processors undertake equal responsibility and all messages must propagate globally throughout the network. This makes the previous Consensus protocols inefficient. In this study, we use a hierarchical concept to reduce the communication overhead. Fig. 1 shows an example of the two-level combined wired/ wireless network. There are six consensus-servers, six stationary processors and nine mobile processors. The network is divided into six zones by six consensus-servers. Each consensus-server manages a zone's processors. For example, consensus-server CS_A manages processor A_1, A_2 and A_3 in the zone A. The parameters and assumptions of our system model are listed as follows:

- The underlying network is un-fully connected.
- The underlying network is a two-level combined wired/wireless network.

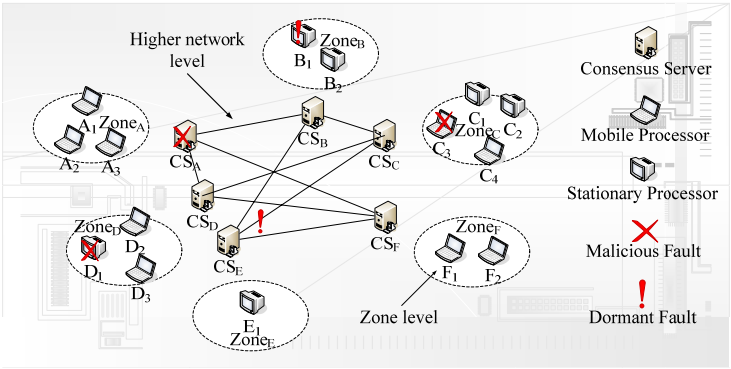

Fig. 1. Two-level combined wired/wireless network.

- The two levels include a higher network level and a zone level.
- The combined wired/wireless network consists of wired backbones and wireless cells that provide access to mobile processors.
- Processors include consensus-server, mobile processor and stationary processor.
- Clients include stationary processor and mobile processor.
- Consensus-server is a powerful and reliable computer with high bandwidth.
- Mobile processor is a processor with mobility.
- Stationary processor is a processor without mobility.
- Let N be the set of all processors in the network and $|N| = n$, where n is the number of processors in the underlying network.
- Let Z_N be the set of all consensus-servers in the network and $|Z_N| = z_n$, where z_n is the number of consensus-servers in the underlying network and $z_n \geq 4$.
- The underlying network is unreliable: messages may be dropped, reordered, inserted or duplicated by faulty components.
- Each processor in the network can be uniquely identified.
- Let p_m be the maximum number of malicious faulty processors allowed.
- Let p_d be the maximum number of dormant faulty processors allowed.
- Let z_m be the maximum number of malicious faulty consensus-servers allowed, $z_m \leq \lfloor (z_n - 1)/3 \rfloor$.
- Let z_d be the maximum number of dormant faulty consensus-servers allowed.
- Let c_m be the maximum number of malicious faulty communication links allowed in the higher network level.
- Let c_d be the maximum number of dormant faulty communication links allowed in the higher network level.
- Let c be the connectivity of each consensus-server in the higher network level, where $c > c_m + c_d + z_m + z_d$.
- The connectivity c_p of Zone$_p$ must be larger than the number of malicious faulty components (processors and communication links) plus the number of dormant faulty components in Zone$_p$, where $1 \leq p \leq z_n$.
- A processor does not know the faulty status of other components in the underlying network.

2.3 Properties of the Consensus Problem

Using the hierarchical concept, each processor in the zone is managed by its consensus-server. Hence, two properties of the Consensus problem in two-level combined wired/wireless network are modified as follows: (1) Consensus: All fault-free processors managed by the fault-free consensus-server agree on a common value; (2) Validity: If the initial value of all consensus-server is v_i, then all fault-free processors managed by the fault-free consensus-server shall agree on v_i.

2.4 The Constraint of Allowable Faulty Consensus-Servers

In the Consensus problem, the number of faulty processors allowed in the network depends on the total number of processors. Meyer and Pradhan [8] indicates the constraint of Consensus problem with malicious faulty processors and dormant faulty processors is $n > 3p_m + p_d$. Afterward, Siu *et al.* [14] finds that the correct constraint should be $n > \lfloor (n - 1)/3 \rfloor + 2p_m + p_d$. The network architectures of Meyer and Pradhan [8] and Siu *et al.* [14] are flat. All processors need to exchange the messages in the Message Exchanging Phase. In our Consensus protocol, the network architecture is hierarchical, and only the consensus-server needs to exchange the messages in the Message Exchanging Phase. Hence, the constraint of our model is $z_n > \lfloor (z_n - 1)/3 \rfloor + 2z_m + z_d$.

3. CONCEPT AND APPROACHES

In this section, Consensus protocol CCP is introduced. In CCP, transmission protocol SRFC is used to transmit messages, so we introduce the transmission protocol at first.

3.1 Secure Communication Channel

A close study of cryptographic technologies is not necessary for our purpose. Hence, we give a brief introduction of some cryptographic technologies that are used in the proposed protocols. Then, we introduce the proposed protocol SRFC.

3.1.1 Related cryptographic technologies

The brief introduction of Diffie-Hellman key exchange, advanced encryption standard and threshold signature are shown here. Diffie-Hellman key exchange [6] is a cryptographic protocol that allows two processors to agree on a secret key over an insecure communication channel.

In a symmetric cryptographic system, the communication parties share a key in advance. They encrypt and decrypt delivered messages by the shared key. The security is based on the shared key. If adversaries reveal the shared key, the symmetric cryptographic system will crash. Advanced Encryption Standard (AES) also known as Rijndael [5] is a block cipher adopted as an encryption standard by the US government and expected to be used worldwide.

In threshold signature scheme, the secret s is divided into k shares and a threshold

value h is set ($h \leq k$). If more than h shares have been collected, we can reconstruct the original secret s. The threshold signature is based on the following equation: $f(x) = a_{h-1}x^{h-1} + a_{h-2}x^{h-2} + \ldots + a_1x + a_0 \bmod p$ where p is a prime number.

3.1.2 Transmission protocol: secure relay fault-tolerance channel (SRFC)

Energy consumption is a major performance metric for mobile processors. If the power consumption is low, the battery lifetime will be longer [13]. In this section, a transmission protocol "Secure Relay Fault-tolerance Channel" (SRFC) is proposed. SRFC can remove the influence from the faulty intermediate component and reduces power consumption to provide a secure communication channel between sender and receiver. The transmission protocol SRFC is shown in Fig. 2.

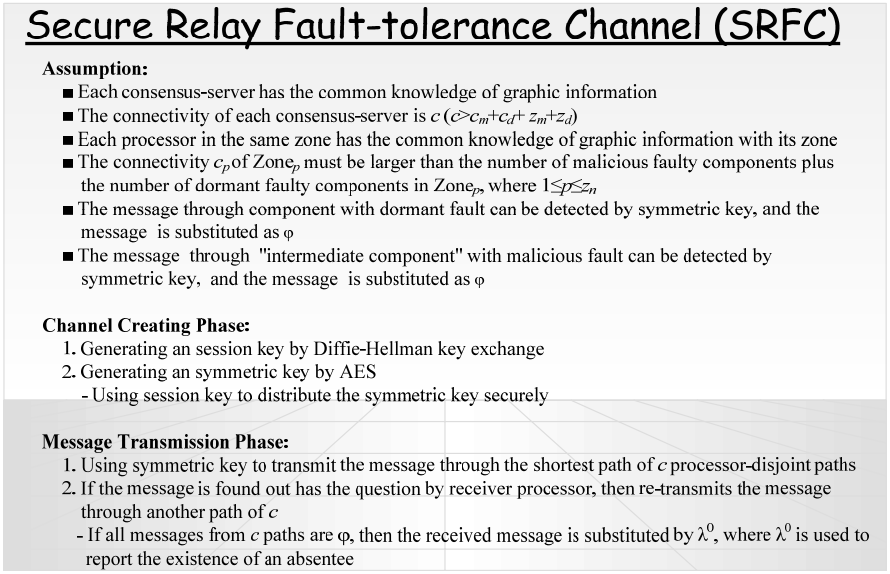

Fig. 2. Secure relay fault-tolerance channel (SRFC).

3.1.3 The connectivity constraint

In Meyer and Pradhan [8], the fallible components are dormant/malicious faulty processors. Meyer and Pradhan [8] used a time-out mechanism to detect a dormant faulty processor. However, the time-out mechanism cannot detect a malicious faulty intermediate component. To avoid the majority value from being dominated by malicious faulty intermediate components, the connectivity constraint in the network by Meyer and Pradhan [8] is c' ($c' > 2p_m + p_d$, where p_m is the maximum number of malicious faulty processors allowed and p_d is the maximum number of dormant faulty processors allowed). In Siu, Chin and Yang [15], the fallible components are dormant/malicious faulty processors and dormant/malicious faulty communication links. Siu, Chin and Yang [15] also used a time-out mechanism to detect a dormant faulty component. Thus, the connectivity constraint in the network by Siu, Chin and Yang [15] is c'' ($c'' > 2p_m + p_d + 2(l_m + l_d)$, where

p_m is the maximum number of malicious faulty processors allowed, p_d is the maximum number of dormant faulty processors allowed, l_m is the maximum number of malicious faulty communication links allowed and l_d is the maximum number of dormant faulty communication links allowed). In SRFC, the network connectivity constraint is improved. For the higher network level, the connectivity of each consensus-server is c ($c > c_m + c_d + z_m + z_d$ where c_m is the maximum number of malicious faulty communication link allowed, c_d is the maximum number of dormant faulty communication links allowed, z_m is the maximum number of malicious faulty consensus-servers allowed and z_d is the maximum number of dormant faulty consensus-servers allowed). For the zone level, the connectivity c_p of Zone_p must be larger than the number of malicious faulty components (processors and communication links) plus the number of dormant faulty components in Zone_p, where $1 \le p \le z_n$. Because a symmetric key is used, the receiver processor can detect a message that is influenced by dormant and malicious faulty intermediate components.

3.1.4 Four cases of fault handling

We classify the fault (or attack) that may take place in a transmission process into four cases: (Case 1) Sender with dormant fault or intermediate component with dormant fault, (Case 2) Intermediate component with malicious fault, (Case 3) Sender with malicious fault, and (Case 4) Receiver with dormant fault or receiver with malicious fault.

Our protocol SRFC can deal with Cases 1 and 2. For Case 1, a message sent through a dormant faulty component cannot be reconstructed by the symmetric key. In Case 2, we can detect that the message is false using the symmetric key. In Case 3, our SRFC cannot detect if the message is correct or not. Because the sender has the symmetric key, the sender has control over the message. Case 3 can be solved using our Consensus protocol CCP. The detailed description of CCP is presented in section 3.2. For Case 4, because the receiver is a faulty component, we do not care the message received from the faulty receiver.

If the network connectivity is c, we can determine c processor-disjoint paths between the sender and receiver. These c processor-disjoint paths can be predefined [16]. An example of c ($c = 3$) processor-disjoint path between CS_B and CS_C is shown in Fig. 3. In Path 2 there is a dormant faulty communication link between CS_E and CS_C. Hence, the message is influenced by the dormant faulty component (Case 1). CS_C can detect this problem using the symmetric key. In Path 3 there is a malicious faulty component CS_A (Case 2), CS_C also can detect this problem using the symmetric key. In Path 1, there is no faulty intermediate component between CS_B and CS_C, CS_C can receive the message from CS_B without faulty influence. That is, if $c > c_m + c_d + z_m + z_d$ ($3 > 0 + 1 + 1 + 0$, $3 > 2$), we can ensure that the receiver can receive the message without influence from faulty intermediate component.

To reduce power consumption, each sender only transmits one copy of the message through the shortest path of c paths. If the receiver detects that the message is false, it then re-transmits the message through another path in c processor-disjoint paths. If all messages from c paths are φ, then the received message is substituted by λ^0. λ^0 is used to report the existence of an absentee. That is, SRFC can remove the influence from dormant/malicious faulty intermediate components and the influence from dormant faulty sender. SRFC is an efficient transmission protocol which reduces the computation time and power consumption to provide a secure communication channel.

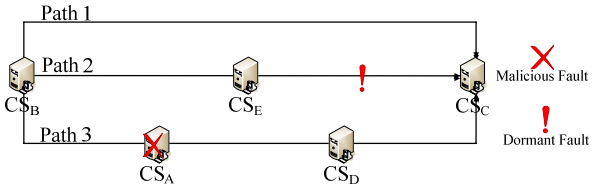

Fig. 3. The c disjoint paths between CS_B and CS_C, where $c = 3$.

3.2 Consensus Protocol: Client-Initiated Consensus Protocol (CCP)

In this section, we would like to focus our attention on the proposed Consensus protocol "Client-initiated Consensus Protocol" (CCP). To meet the characteristics of Consensus problem in combined wired/wireless networks, most of the communications and computation overhead must be fulfilled within by the consensus-servers. Therefore, only consensus-servers need to exchange messages and compute the common value in CCP. Furthermore, all messages in CCP are transmitted by SRFC. There are two stages in CCP, namely the Client-initiated Stage and the Consensus Stage. The Consensus protocol CCP is shown in Fig. 4.

In this section, we also give an example of executing SRFC and CCP. An example of two-level combined network used in this section is shown in Fig. 1. The dormant faulty components are B_1 and L_{CE}. The malicious faulty components are CS_A, D_1 and C_3.

3.2.1 Client-initiated stage

The purpose of the Client-initiated Stage is to collect the initial value from each client and compute the pre-consensus value for each consensus-server. Any client may initiate Consensus in the network. For example, B_2 wants to initiate a Consensus. Hence, B_2 creates a secure communication channel between CS_B by SRFC. B_2 then transmits the "initiate-consensus" message to CS_B through processor-disjoint path created by SRFC. After CS_B receives the "initiate-consensus" message, CS_B creates secure communication channels to all other consensus-servers by SRFC and then informs all consensus-servers to gather the initial value from each client in its zone. The initial value of each client is shown as follows.

Client ID:	A_1	A_2	A_3	B_1	B_2	C_1	C_2	C_3	C_4	D_1	D_2	D_3	E_1	F_1	F_2
Initial value:	0	1	0	λ^0	1	1	1	1	1	1	0	1	0	0	0

Each consensus-server then obtains the pre-consensus value using the threshold signature. For example, there are two processors in $Zone_B$, one is B_1 which is a dormant faulty processor, and another is B_2 which is a fault-free processor. Hence, CS_B can detect that the message from B_1 is influenced by a dormant faulty component. After B_1 and B_2 sign its initial value to CS_B, CS_B can obtain the pre-consensus value 1 (the number of value 1 is greater than or equal to half of the number of processor in $Zone_B$, λ^0 is ignore). The pre-consensus value of each consensus-server is shown as follows.

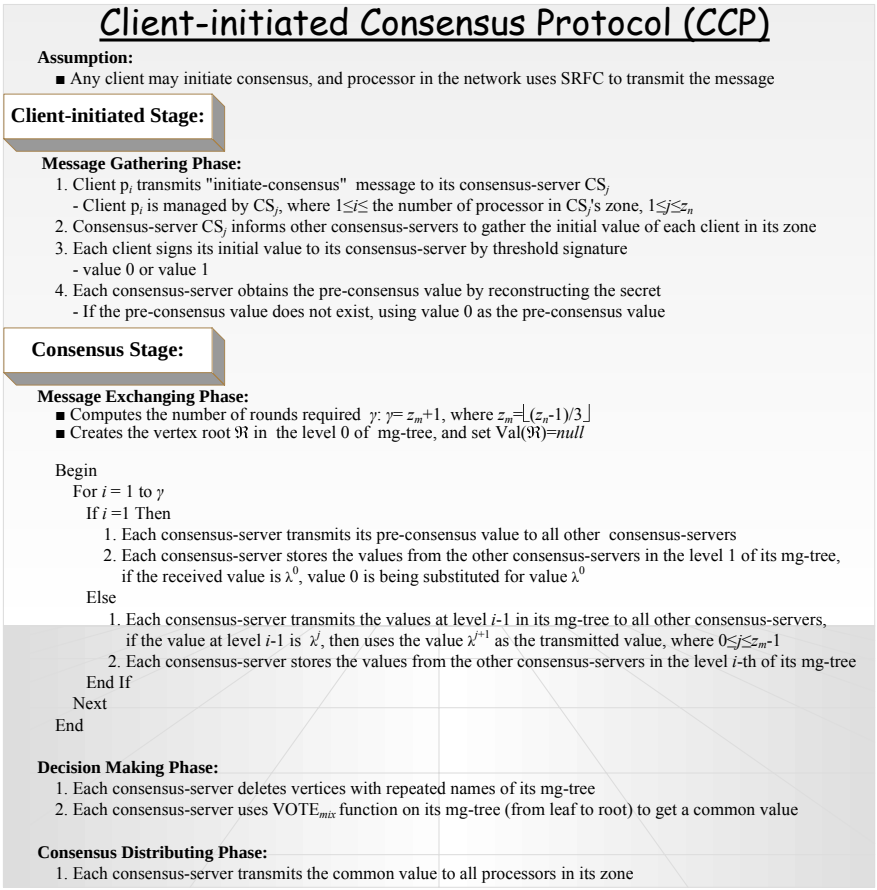

Fig. 4. Client-initiated consensus protocol (CCP).

Consensus-Server ID:	CS_A	CS_B	CS_C	CS_D	CS_E	CS_F
Pre-consensus value:	0	1	1	1	0	0

3.2.2 Consensus stage

The purpose of the Consensus Stage is to compute a common value. There are three phases in the Consensus stage; including the Message Exchanging Phase, the Decision Making Phase and the Consensus Distributing Phase.

In the Message Exchanging Phase, we first compute the number of rounds required γ. The number of rounds required for the network model in Fig. 1. is $\gamma = 2$ ($\gamma = \lfloor (6 - 1)/3 \rfloor + 1$). Then, we creates the vertex $\Re$ in the level 0 of its mg-tree, and set $Val(\Re) = null$. The mg-tree is a tree structure that is used to store received messages (the detailed description of the mg-tree is presented in Appendix). In the first round of the Message Exchanging Phase, each consensus-server transmits its pre-consensus value to all other consensus-servers by SRFC. Each consensus-server then stores the values from other

consensus-servers in the level 1 of its mg-tree. Since CS_A is a malicious faulty processor, CS_A may transmit different messages to different consensus-server to prevent the fault-free consensus-server from reaching a common value. The messages transmitted by CS_A in the first round of Message Exchanging Phase are shown in Fig. 5. An mg-tree of CS_B after the first round of Message Exchanging Phase is shown in Fig. 6.

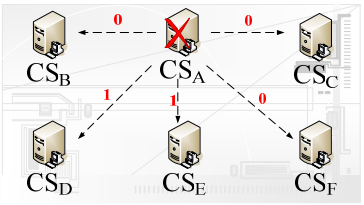

Fig. 5. CS_A transmits different message to different consensus-server.

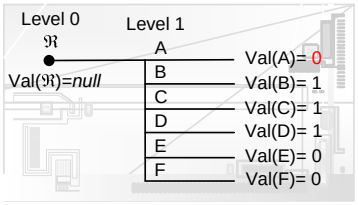

Fig. 6. A one-level mg-tree of CS_B.

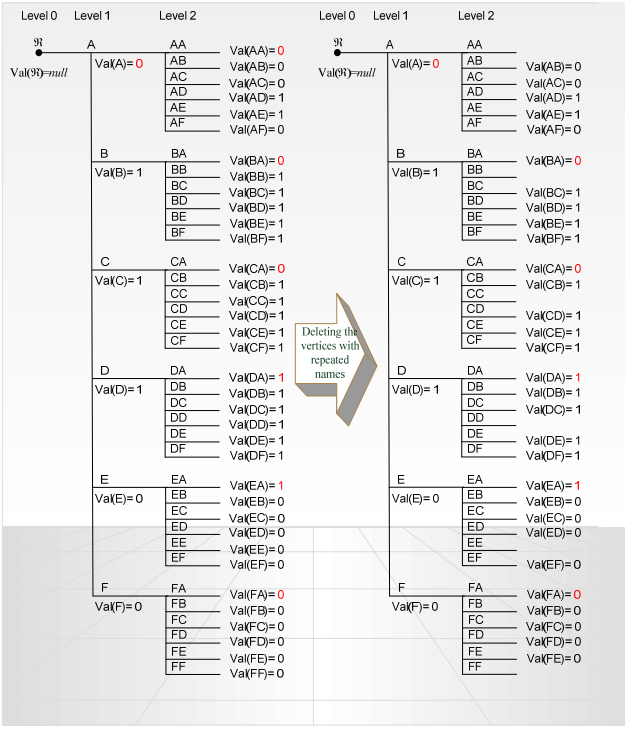

Fig. 7. An mg-tree of CS_B after the second round of the message exchanging phase.

Fig. 8. An mg-tree of CS_B without repeated name vertices.

In the second round of the Message Exchanging Phase, each consensus-server transmits the value at level 1 in its mg-tree to all other consensus-servers, and stores the value from other consensus-servers in the level 2 of its mg-tree. An mg-tree of CS_B after the second round of the Message Exchanging Phase is shown in Fig. 7.

3.2.3 Decision making phase

In the Decision Making Phase, each consensus-server deletes vertices with repeated names of mg-tree to avoid the repeated influence from faulty consensus-servers. An mg-tree of CS_B without repeated name vertices is shown in Fig. 8. Each consensus-server then uses the $VOTE_{mix}$ function on its mg-tree (from leaf to root) to compute the common value. The $VOTE_{mix}$ function is shown in Fig. 9. Conditions 1, 4 and 5 are similar to convention majority vote [11]. Condition 2 is used to deal with the dual failure mode (where both dormant fault and malicious fault exist). Condition 3 is used to describe the existence of an absentee. For example, CS_B can obtain the common value ϕ by $VOTE_{mix}$. $VOTE_{mix}(\Re) = ((0, 0, 1, 1, 0), (0, 1, 1, 1, 1), (0, 1, 1, 1, 1), (1, 1, 1, 1, 1), (1, 0, 0, 0, 0), (0, 0, 0, 0, 0)) = (0, 1, 1, 1, 0, 0) = \phi$.

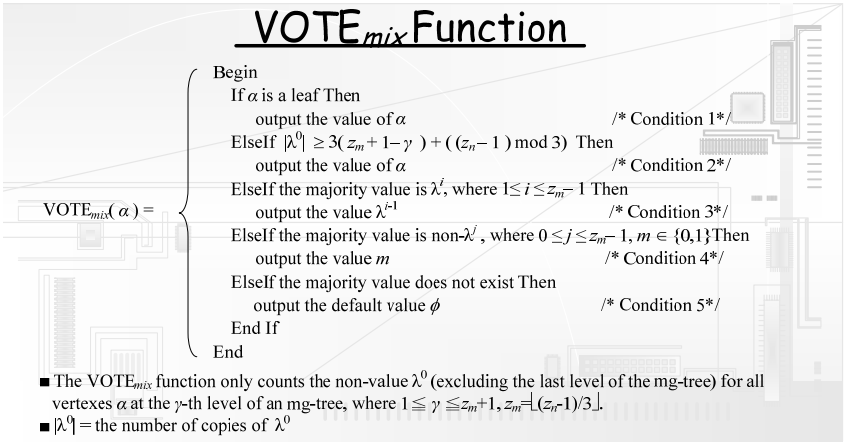

Fig. 9. The $vote_{mix}$ function.

3.2.4 Consensus distributing phase

Each consensus-server transmits the common value to all processors in its zone. All fault-free processors (both stationary processors and mobile processors), which are managed by the fault-free consensus-server, can obtain a common value. The value agreed upon by a processor, which is managed by faulty consensus-server, is ignored [11].

4. CORRECTNESS AND COMPLEXITY

The goal of CCP is to enable all fault-free consensus-server to reach a common value to solve the Consensus problem in a combined wired/wireless network. To prove the correctness of our protocol CCP, a vertex $\Re$ is called common [2] if each fault-free consensus-server has the same value for $\Re$. That is, if vertex $\Re$ is common, then the value stored in vertex $\Re$ of each fault-free consensus-server's mg-tree is identical.

Lemma 1 All correct vertices of an mg-tree are common after function $VOTE_{mix}$ is applied to mg-tree.

Proof: In the Decision Making Phase, all vertices with repeated names are deleted in an mg-tree. At level $z_m + 1$ or above, the correct vertex α has at least $2z_m + 1$ children, and out of which at least $z_m + 1$ children are correct. The true values of these $z_m + 1$ correct vertices are common, and the majority of the vertex value α is common. The correct vertex α is common in the mg-tree if the level of α is less then $z_m + 1$. Consequently, all correct vertices of the mg-tree are common.

Lemma 2 The common frontier exists in the mg-tree.

Proof: By definition, an mg-tree is a tree of level $z_m + 1$. There are $z_m + 1$ vertices along each root-to-leaf path of an mg-tree. Since at most z_m consensus-servers can fail, there is at least one correct vertex along each root-to-leaf path of the mg-tree. Using Lemma 1, the correct vertex is common and the common frontier exists in each fault-free consensus-server's mg-tree.

Lemma 3 Let α be a vertex, α is common if there is a common frontier in the sub-tree rooted at α.

Proof: If the height of α is 0 and the common frontier (α itself) exists, α is common. If the height of α is γ, the children of α are all in common under the induction hypothesis with the height of the children being $\gamma - 1$.

Corollary 1 The value of root $\Re$ is common if the common frontier exists in the mg-tree.

Theorem 1 The value of root $\Re$ of a fault-free consensus-server's mg-tree is common.

Proof: Using Lemmas 1, 2, 3 and Corollary 1, the theorem is proved.

Theorem 2 Protocol CCP solves the Consensus problem in a two-level combined wired/wireless network.

Proof: To prove this theorem, CCP must meet the constraints (Consensus') and (Validity').
(Consensus'): Root value is common. By Theorem 1, (Consensus') is satisfied.
(Validity'): VOTE (α) = v for all fault-free consensus-servers, if the initial value of all consensus-server is v_s say $v = v_s$.

Most consensus-servers are fault-free. The value of the correct vertices for all of the fault-free consensus-servers' mg-trees is v. Therefore, each correct vertex of the mg-tree is common (Lemma 1), and its true value is v. Using Theorem 1, this root is common. The computed value VOTE (α) = v is stored in the root for all the fault-free consensus-server. Therefore, (Validity') is satisfied.

Theorem 3 CCP requires $z_m + 1$ rounds in the Message Exchanging Phase to solve the Consensus problem in a two-level combined wired/wireless network, and $z_m + 1$ ($z_m = \lfloor (z_n - 1)/3 \rfloor$) is the minimum number of rounds in the "Message Exchanging Phase".

Proof: In the Consensus protocol, we use term "round" to compute the number of messages exchanged. A round is defined as follows: (1) sending messages to any set of nodes, (2) receiving messages, and (3) processing the messages locally [7, 11]. The "Message Exchanging Phase" is a time consuming phase. Fischer and Lynch [7] indicated that $t + 1$ ($t = \lfloor (n - 1)/3 \rfloor$) rounds are the minimum number of rounds required to get enough messages to achieve Consensus. The network architecture of Fischer and Lynch [7] is a flat architecture, but the network architecture of our system is two-level architecture. In our protocol, only consensus-servers need to exchange the messages in the Message Exchanging Phase, so the number of required rounds of message-exchange is $z_m + 1$ ($z_m = \lfloor (z_n - 1)/3 \rfloor$). Thus, CCP requires $z_m + 1$ rounds, and this number is the minimum.

5. CONCLUSION

Three motives are combined in this paper on the Consensus problem in combined wired/wireless network. First, most networks today are combined wired/wireless networks. Extant Consensus protocols are not applicable to combined wired/wireless networks. Hence, we proposed the protocol CCP to solve the Consensus problem in combined wired/wireless network. Second, the limited resources have made the computation ability of mobile processors often weaker than that of stationary processors. The proposed SRFC provides an efficient and secure communication channel. Third, the communication overhead of the Consensus protocol is inherently large. We used the hierarchical concept in CCP to reduce the large amount of communication overheads. Therefore, CCP is more efficient than the previous protocols when the network is logically divided into hierarchical architecture. Table 1 shows some instances of the number of rounds required for flat network and two-level network. Smaller number of zone is preferred since the number of rounds required in the Message Exchanging Phase is smaller.

In this paper, we solved the Consensus problem with dual failure modes (dormant/malicious faulty components) in the combined wired/wireless network. CCP requires only $z_m + 1$ rounds (minimum number of rounds) in the Message Exchanging Phase which is optimal for all fault-free processors managed by fault-free consensus-servers to reach a common value.

Table 1. Some instances of the number of rounds required for various consensus protocols.

	The number of rounds required in message exchanging phase			
		$n = 128, z_n = 32$	$n = 128, z_n = 16$	$n = 128, z_n = 8$
Flat Network	$t + 1$, $t = \lfloor (n - 1)/3 \rfloor$	43	43	43
Two-Level Network	$z_m + 1$, $z_m = \lfloor (z_n - 1)/3 \rfloor$	11	6	3

n: the number of processors in the underlying network and $n \geq 4$.
z_n: the number of zones in the underlying network and $z_n \geq 4$.

APPENDIX

The message gathering tree (mg-tree)

In the beginning Consensus Stage, each consensus-server creates the vertex $\Re$ in the level 0 of its mg-tree, and set Val ($\Re$) = *null*. At the first round of Message Exchanging Phase, each consensus-server transmits its initial value to the other consensus-servers. We assume that each receiver can always identify the sender of a message. For example, when CS_B receives the initial value sent from CS_A, CS_B stores the received value, denoted as *val* (*A*), in vertex *A* of its mg-tree as shown in Fig. 6. Similarly, when CS_B receives the initial value sent from CS_C, CS_B stores the received value, denoted as *val* (*C*), in vertex *C* of its mg-tree as shown in Fig. 6.

At the second round of Message Exchanging Phase, each consensus-server transmits the values in level 1 of its mg-tree to the other consensus-servers. If CS_A sends message *val* (*B*) to CS_B, then CS_B stores the received messages from CS_A, denoted as *val* (*BA*), in vertex *BA* of its mg-tree as shown in Fig. 7.

Generally speaking, message *val* (*ab…n*), stored in the vertex *ab…n* of an mg-tree, implies that the message just received was sent through the CS_a, CS_b, …, CS_n, where CS_n is the latest consensus-server to pass the message. When a message is transmitted through a consensus-server more than once, the name of the consensus-server will be repeated correspondingly. For instance, the appearance of message *val* (*AA*) in vertex *AA* in Fig. 8 indicates that the message is sent from CS_A to CS_A again; therefore, CS_A appears twice in vertex name *AA*.

REFERENCES

1. O. Babaoglu and R. Drummond, "Streets of Byzantium: Network architectures for fast reliable broadcasts," *IEEE Transactions on Software Engineering*, Vol. 11, 1985, pp. 546-554.
2. A. Bar-Noy, D. Dolev, C. Dwork, and R. Strong, "Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement," *Information and Computation*, Vol. 97, 1992, pp. 205-233.
3. T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing*, Vol. 2, 2002, pp. 483-502.
4. G. D. Crescenzo, R. Ge, and G. R. Arce "Securing reliable server pooling in MANET against Byzantine adversaries," *IEEE Journal on Selected Areas in Communications*, Vol. 24, 2006, pp. 357-369.
5. J. Daemen and V. Rijmen, "The rijndael block cipher," *AES Document Version* 2.
6. W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, Vol. 22, 1976, pp. 644-654.
7. M. Fischer and N. Lynch, "A lower bound for the assure interactive consistency," *Information Processing Letters*, Vol. 14, 1982, pp. 183-186.
8. F. J. Meyer and D. K. Pradhan, "Consensus with dual failure modes," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 2, 1991, pp. 214-222.
9. S. Marano, V. Matta, and L. Tong, "Distributed inference in the presence of Byzan-

tine sensors," in *Proceedings of IEEE Asilomar Conference on Signals*, *Systems*, *and Computers*, 2006, pp. 281-284.
10. M. Okum, "Agreement among unacquainted Byzantine generals," *Lecture Notes in Computer Science*, Vol. 3724, 2005, pp. 499-500.
11. M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of ACM*, Vol. 27, 1980, pp. 228-234.
12. G. Rabbat, D. Nowak, and A. Bucklew, "Generalized consensus computation in networked systems with erasure links," in *Proceedings of the 6th IEEE Workshop on Signal Processing Advances in Wireless Communications*, 2005, pp. 1088-1092.
13. T. Simunic, "Power saving techniques for wireless LANs," in *Proceedings of the Conference on Design*, *Automation and Test in Europe*, Vol. 3, 2005, pp. 96-97.
14. H. S. Siu, Y. H. Chin, and W. P. Yang, "A note on consensus on dual failure modes," *IEEE Transactions on Parallel and Distributed System*, Vol. 7, 1996, pp. 225-230.
15. H. S. Siu, Y. H. Chin, and W. P. Yang, "Byzantine agreement in the presence of mixed faults on processors and links," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, 1998, pp. 980-986.
16. D. B. West, *Introduction to Graph Theory*, 2nd ed., Prentice Hall, New Jersey, 2001.
17. K. Q. Yan, Y. H. Chin, and S. C. Wang, "Optimal agreement protocol in malicious faulty processors and faulty links," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, 1992, pp. 266-280.

Chien-Fu Cheng (鄭建富) received the Ph.D. in Computer Science from National Chiao Tung University, Taiwan. His current research interests include distributed data processing, fault tolerant computing, and mobile computing.

Shu-Ching Wang (王淑卿) received the B.S. in Computer Science from Feng Chia University, the M.S. in Electrical Engineering from National Cheng Kung University, and Ph.D. in Information Engineering from National Chiao Tung University, Taiwan. Currently, she is a Professor with the Graduate Institute of Informatics, Chaoyang University of Technology, Taichung County, Taiwan. Her current research interests include grid computing, distributed data processing, parallel processing, and algorithm analysis and design.

Tyne Liang (梁婷) received her Ph.D. in Computer Science and Information Engineering from National Chiao Tung University, Taiwan. Currently, she is an Associate Professor with Department of Computer Science, National Chiao Tung University. Her research interests include information retrieval, natural language processing and interconnection network.