Journal of Network and Computer Applications 48 (2015) 87–98

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A recursive *Byzantine*-resilient protocol

Chien-Fu Cheng [a,b,*], Kuo-Tang Tsai [b]

[a] Department of Computer Science and Information Engineering, Tamkang University, No. 151, Yingzhuan Road, Tamsui District, New Taipei City 25137, Taiwan

[b] Graduate Institute of Networking and Communication, Tamkang University, No. 151, Yingzhuan Road, Tamsui District, New Taipei City 25137, Taiwan

ARTICLE INFO

Article history:
Received 6 December 2013
Received in revised form
9 September 2014
Accepted 31 October 2014
Available online 12 November 2014

Keywords:
Distributed system
Fault-tolerant
Consensus problem
Byzantine fault

ABSTRACT

To solve the consensus problem, the classical consensus protocols require $t+1$ rounds of message exchange to tolerate t faulty processors, where $t=\lfloor(n-1)/3\rfloor$ and n is the total number of processors in the network. With advancement of software and hardware technologies in recent years, the "actual number of faulty processors" (f_{act}) in a network is usually smaller than t, and $f_{act} \ll t$. However, the classical consensus protocols still need to execute $t+1$ rounds of message exchange even if there are no faulty processors in the network. To address this issue, we propose a new consensus protocol called Recursive Byzantine-Resilient protocol (RBR protocol). We integrate the concepts of parallel computing, grouping, hierarchy and recursion into this protocol to reduce its time and space complexity. Specifically, the RBR protocol can solve the consensus problem in the presence of $2^h(\lfloor((n/4^h)-1)/3\rfloor+1)-1$ Byzantine faulty processors, where $h=\lfloor(\lg(n)-2)/2\rfloor$. The time complexity and space complexity of RBR protocol are $O(\lg(n))$ and $O(nk^{\lg(n)})$ respectively. The results reveal that RBR protocol outperforms previous protocols in terms of time complexity and in terms of space complexity. In this paper, we also discuss how to enhance the fault-tolerance capability of RBR protocol in achieving consensus through repetitive execution of the protocol when the number of Byzantine faulty processors is greater than $2^h(\lfloor((n/4^h)-1)/3\rfloor+1)-1$.

1. Introduction

The consensus problem plays a key role in the design of a fault-tolerant distributed system (Silberschatz et al., 2013). The goal of consensus protocols is to make each fault-free processor reach a common consensus value (Fischer et al., 1985; Neiger, 1994; Pease et al., 1980; Siu et al., 1996). With a common consensus value, many important distributed services/applications can be carried out. Some examples are the implementation of practical intrusion-tolerant systems (Colon Osorio, 2007; Correia et al., 2006), the implementation of the web services atomic transactions over the Internet (Zhang et al., 2012), group membership service (Guerraoui and Schiper, 2001), file consistency problem of file-sharing in P2P network (Amir et al., 2010; Shen, 2010) and the cruise control (Correia et al., 2008).

The original consensus protocol was proposed by Pease et al. (1980). The consensus protocols proposed by Fischer et al. (1985), Neiger (1994) and Siu et al. (1996) were all based on the original consensus protocol, and they have the same execution time (time complexity) and the number of messages required (space complexity) as the original consensus protocol. Fischer et al. (1985) show that consensus cannot be solved deterministically in completely asynchronous distributed systems, even in the presence of only one crash processor. Neiger (1994) included a strong validity constraint (Strong Validity: the output value of each fault-free processor must be the initial value of some fault-free processors) in the original consensus protocol to propose a new consensus protocol. Siu et al. (1996) proposed a consensus protocol to solve the consensus problem with dual failure modes. Their protocol was to improve the system's fault tolerance by classifying faults into Byzantine fault and dormant fault.

We call a consensus protocol a t-Resilient algorithm, where t is a threshold on the number of potential failures. The algorithm is correct as long as no more than t processors fail. Most of the previous consensus protocols assume that the number of fallible processors in the network is $t=\lfloor(n-1)/3\rfloor$ to design a t-Resilient algorithm (Fischer et al., 1985; Neiger, 1994; Pease et al., 1980; Siu et al., 1996). The number of rounds of message exchange of these consensus protocols is $t+1$. The time complexity and space complexity of these algorithms are denoted as $O(n)$ and $O(n^n)$ respectively, where n is the total number of processors in the network ($n \geq 4$). The term round[1] is

* Corresponding author at: Department of Computer Science and Information Engineering, Tamkang University No. 151, Yingzhuan Road, Tamsui District, New Taipei City 25137, Taiwan.
E-mail address: cfcheng@mail.tku.edu.tw (C.-F. Cheng).

[1] A round is defined as follows: (1) Sends messages to any set of processors; (2) Receives messages from this round; and (3) Does local processing (Neiger, 1994).

http://dx.doi.org/10.1016/j.jnca.2014.10.010
1084-8045/

used to compute the number of message exchanges. More precisely, the consensus problem is defined by the following three properties: (1) validity: if the initial value of all processors is v_i, then *all* fault-free processors shall agree on v_i; (2) consensus: all fault-free processors agree on a common value; (3) termination: every fault-free processor eventually decides.

With advancement of software and hardware technologies in recent years, the actual number of faulty processors (f_{act}) in a network is usually smaller than $t=\lfloor(n-1)/3\rfloor$. In practice, f_{act} is usually small, and $f_{act} \ll t$. Given the probability of faulty processors in modern networks is generally smaller than 1/3, using previous t-Resilient algorithms (1/3 processors are fallible) to calculate the consensus value will cause a waste of time on unnecessary rounds of message exchange. To enhance the performance of consensus protocols, recent researchers have attempted to propose the early stopping algorithm for the consensus problem (Krings and Feyer, 1999). The feature of the early stopping algorithm is that it determines the sufficiency of messages collected for earlier termination of message exchange. The number of rounds of message exchange of early stopping algorithm is $min\{f_{act}+2, t+1\}$, where f_{act} is the actual number of Byzantine faulty processors in the network. Despite the effectiveness of the early stopping algorithm in addressing time and space complexity, there may be other ways to further reduce the time and space complexity of consensus protocols. Therefore, in this study, we propose a new consensus protocol based on the concepts of parallel computing, grouping, hierarchy and recursion for networks where the probability of faulty processors is smaller than 1/3. The proposed protocol is called Recursive Byzantine-Resilient protocol (RBR protocol). The time complexity and space complexity of RBR protocol are $O(\lg(n))$ and $O(nk^{\lg(n)})$ respectively. Results indicate that RBR protocol is better than previous protocols (Fischer et al., 1985; Krings and Feyer, 1999; Neiger, 1994; Pease et al., 1980; Siu et al., 1996) both in terms of time complexity and in terms of space complexity.

This paper consists of seven sections, and the remainder is organized as follows. Section 2 describes the system model and problem formulation. Section 3 describes the concept and approach of the proposed protocol. Section 4 presents the proposed protocol. Section 5 shows the correctness and complexity of the proposed protocol. Section 6 discusses how to enhance fault-tolerance capability of the proposed protocol. Finally, the conclusion is drawn in Section 7.

2. System model and problem formulation

In previous literature, Fischer et al. (1985) have shown that consensus cannot be solved deterministically in completely asynchronous distributed systems[2], even in the presence of only one crash processor. Therefore, in the most of the consensus problem (Fischer et al., 1985; Neiger, 1994; Pease et al., 1980; Siu et al., 1996), processors are fully connected by reliable channels in synchronous networks. In this study, the consensus problem is also considered in a synchronous fully connected network with reliable channels. In such a network, the bounds on the processing and communication delays of fault-free processors are finite, and processors are connected to each other directly (Baldoni et al., 2010; kamil et al., 2010). It is assumed that fallible processors have the most serious Byzantine fault in this study. When a Byzantine fault occurs, the processor may respond in all kinds of unpredictable ways. In the worst case, the Byzantine faulty processors may conspire with other faulty processors (Bessani et al., 2009; Biely and Hutle, 2011; Zdarsky et al., 2012). The parameters of our system model are listed as follows:

- $\mathcal{P}$ is the set of processors in the network, where $\mathcal{P}=\{p_1, p_2,\ldots, p_n,\}$ and $n=|\mathcal{P}|$.
- $\mathcal{H}$ is the set of fault-free processors in the network, where $\mathcal{H}\subseteq\mathcal{P}$.
- f is the number of tolerable Byzantine faulty processors in the network, where $\mathcal{F}$ is the set of Byzantine faulty processors, $\mathcal{F}\subseteq\mathcal{P}$ and $f=|\mathcal{F}|$.
- $\mathcal{V}(p)$ is the initial value of processor p.
- $\mathcal{D}(p)$ is the consensus value of processor p.
- $\mathcal{T}$ is the execution time of the proposed protocol.
- $\mathcal{M}$ is the amount of messages generated.

As aforementioned, the goal of consensus protocols is to make each fault-free processor obtain a common consensus value. It is better that the number of rounds of message exchange and the amount of message generated are fewer. Hence, we set our objective function as expressed in Eqs. (G1)–(G3). Besides, Eq. (C1) is used to limit the initial value of each processor p; Eq. (C2) is used to limit the number of Byzantine faulty processors in the network.

$$\mathcal{D}(p)=\mathcal{D}(q), \forall p,q\in\mathcal{H}, p\neq q. \tag{G1}$$

$$minimize\ \mathcal{T} \tag{G2}$$

$$minimize\ \mathcal{M} \tag{G3}$$

Subject to:

$$\mathcal{V}(p)\in\{0,1\},\quad \forall p\in\mathcal{H} \tag{C1}$$

$$f\le 2^h(\lfloor((n/4^h)-1)/3\rfloor+1)-1,\ \text{where } h=\lfloor(lg(n)-2)/2\rfloor \text{ (Theorem 1)} \tag{C2}$$

3. The concept and approach of the proposed protocol

In this study, we propose the Recursive Byzantine-Resilient protocol, RBR protocol in short, to solve the consensus problem with Byzantine faulty processors in the synchronous fully connected network. RBR protocol is designed on the basis of the classical consensus protocols (Fischer et al., 1985; Neiger, 1994; Pease et al., 1980; Siu et al., 1996). It integrates the concepts of grouping, recursion and hierarchy, and uses parallel computing to reduce the execution time and the amount of messages required to achieve consensus.

Theorem 1. *There are $h+1$ levels in the network which has n processors, where $h=\lfloor(lg(n)-2)/2\rfloor$.*

Proof: The RBR protocol puts 4–5 processors/subgroups in a group (in each grouping operation, at most 3 groups will have 5 processors/subgroups). Given a network with n processors, the recursive grouping operation must satisfy the constraint of $n/4^h\ge 4$. After transposition, we can get $h=\lfloor(\lg(n)-2)/2\rfloor$.

The grouping method of RBR protocol is as follows: RBR protocol puts 4–5 processors/subgroups in a group (at most 3 groups have 5 processors/subgroups). In a network with n processors, there will be $h+1$ hierarchies (numbered from 0, $h=\lfloor(\lg(n)-2)/2\rfloor$ as shown in Theorem 1). Thus, if n is between 4 and 15, h will be 0 ($h=\lfloor(\lg(n)-2)/2\rfloor=0$, $n=4$–15). In this case, grouping is not applicable, because the grouping result will not meet the requirement of the consensus protocol that the number of groups must be greater than or equal to 4 (Fischer et al., 1985; Neiger, 1994; Pease et al., 1980; Siu et al., 1996). If $n\ge 16$, h will be greater than or equal to 1 ($h=\lfloor(\lg(n)-2)/2\rfloor\ge 1$, $n\ge 16$), and grouping is applicable. RBR protocol groups processors based on a recursive method, which is to group processors, from one

[2] The features of completely asynchronous distributed systems are: (1) No access to real-time clocks; and (2) No assumption on communication delays and relative speed of processors (kamil et al., 2010).