

# A Crash-tolerant Consensus Algorithm in Presence of Probabilistic Message Omission

Taisuke Izumi, Koichi Wada

## I. INTRODUCTION

*Consensus* problem is one of fundamental and important problems for designing fault-tolerant distributed systems. In the consensus problem, each process proposes a value, and has to agree on a common value that is proposed by a process unless it crashes. The consensus problem has many practical applications, e.g., atomic broadcast[1], shared objects[3], [4], weak atomic commitment[5] and so on. However, despite of such applications, it has no deterministic solution in asynchronous systems subject to only a single crash fault. Thus, several consensus algorithms have been considered on systems with some additional assumptions[1], [6], [7], [8]. Especially, the *round-based synchrony* is one of the most commonly used assumptions for designing consensus algorithms. Executions of the round-based synchronous systems proceed based on consecutive communication-closed rounds, where “communication-closed” means that every message sent in a round is received in the same round.

While the round-based synchrony provides some kind of simplicity to algorithm designs and analysis, it is not easy to implement round-based synchronous consensus algorithms on real systems. The most critical difficulty derives from the difference of the assumption for message delays: A realistic approach to implement round-based synchronous algorithms is to simulate the round-based synchrony on real systems with timeout mechanism by using the local clock of each process. That is, each process divides its execution into consecutive periods using local clocks, where one period corresponds to one round. Then, by making the length of each period sufficiently long, the simulated round-based synchrony becomes “almost” communication-closed. However, real distributed systems occasionally have messages with very large transmission delay exceeding the length of the round period (we call such ones *overdelayed messages*). Overdelayed messages break the communication-closedness of the simulated round-based synchrony. Unfortunately, we cannot completely eliminate overdelayed messages because of the essential unpredictability of distributed systems such as overload of processes or message congestion. Thus, it is impossible to simulate the perfect round-based synchrony on real distributed systems.

The overdelayed messages sent at a round  $r$  on simulated round-based systems is eventually received at round  $r + 1$  or later. However, to preserve the communication-closedness, such messages must be handled as *message omissions* at the end of round  $r$ . In this sense, the round-based synchronous

systems simulated on real systems can be regarded as the systems suffering omission faults. From the perspective of algorithm implementability, this observation implies that the omission-fault tolerance is required so that algorithms for round-based synchronous systems can work correctly on real systems. While the omission-fault tolerance on round-based synchronous systems are studied in several literatures[8], [9], all of them model message omissions as process faults, i.e., only messages sent by faulty processes can be omitted. This modeling does not match our objective because any message has a possibility of omission (that is, overdelayed) in the round-based synchronous systems simulated on real systems.

Motivated by the above discussion, in this paper, we introduce a novel model of round-based synchronous systems, where each process suffers *probabilistic omission faults* in addition to *crash faults*, and consider consensus algorithms on the model. In our model, any message can be omitted with a probability  $p$  (the value of  $p$  is a system parameter). As we mentioned in the above discussion, an omission fault is equivalent to a message overdelay, and is caused by nondeterminism and unpredictability in the behavior of distributed systems. Thus, it is natural to model the message omission as probabilistic faults. We propose a consensus algorithm working correctly in our model. Interestingly, compared with existing algorithms that does not tolerate probabilistic omissions, the worst-case expected round complexity of our algorithm does not become so large, even if the lost probability  $p$  is high. More precisely, the worst-case expected round complexity of our algorithm is  $O(f)$  rounds if  $p < 1 - 4/\sqrt{n}$ , where  $f$  is the actual number of crash faults, and  $n$  is the number of all processes in the system. Since it is proved that any algorithm tolerating  $f$  crash faults requires at least  $f+2$  rounds in the worst case, our algorithm has the round complexity only a constant time as much as existing non-omission-tolerant algorithms.

## II. OVERVIEW OF THE ALGORITHM

The design of our algorithm is based on the *rotating coordinator* paradigm. An execution of our algorithm is divided into several periods called *units*. A coordinator, which behaves like as a leader, is assigned to one unit. This assignment is static: The process having id  $i$  is the coordinator of units  $i, n+i, 2n+i, \dots$ . Each process maintains a candidate of the decision value. In each unit, the coordinator tries to impose its candidate value to all other processes by broadcasting a message including the candidate value. The processes that receive the message from the coordinator updates its candidate by the coordinator’s one, and tests whether the majority of

The authors are with the Graduate School of Engineering, Nagoya Institute of Technology (NIT), Aichi, Japan. E-mail: {t-izumi, wada}@nitech.ac.jp.

```

1: variable:
2:    $M_i$  : init  $\perp$ 

3: At round 1:
4:   if  $p_i$  is the source process then
5:      $\text{Send}_i(d)$  to all processes
6:      $M_i \leftarrow d$ 
7:   endif
8:   if a message  $d$  is received from  $p_j$  then  $M_i \leftarrow d$  endif
9: At round  $r(2 \leq r \leq l)$ :
10:  if  $M_i \neq \perp$  then  $\text{Send}_i(d)$  to all processes endif
11:  if a message  $d$  is received from  $p_j$  then  $M_i \leftarrow d$  endif

```

Fig. 1. A  $l$ -Round Full-Information Broadcast Algorithm for Transmission Data  $d$  (Code for a process  $p_i$ )

the processes update its value in the current unit. Then, it is guaranteed that at least one process passes the test with high probability in the unit such that its coordinator does not crash. The passed process decides its candidate, and broadcasts a success message with the decision value to all processes. If a process receives the success message, it also decides. The expected number of units that the algorithm takes by the decision is  $f + 2$  units because the first  $f + 1$  units includes at least one unit whose coordinator is not crashed. The key idea of our algorithm is the fact that the above impose/commit task can be processed within a constant number of rounds (that is, the length of each unit is a constant) even if messages are lost with high probability ( $< 1 - 4/\sqrt{n}$ ). Each task is realized by the full-information broadcast Primitive (FIB). The principle of FIB is quite simple (Figure 1): When an information  $d$  is broadcast by FIB, the information is repeatedly broadcast at each round by the source process of  $d$  and ones that receives  $d$  in previous rounds. More precisely, the FIB algorithm consists of  $l$  rounds (where  $l$  is a predefined constant). At the first round, the source process broadcasts the information  $d$  to all processes. At round  $r(2 \leq r \leq l)$ , all processes that received  $d$  in previous rounds broadcast the information  $d$ . In FIB algorithm,  $l = 3$  is sufficient to guarantee the success of broadcast with high probability. This implies that our algorithm can achieve  $O(f)$  round complexity. The pseudocode description of our algorithm is given in Figure 2.

We obtain the following theorem by this algorithm.

*Theorem 1:* For message omission probability  $p < 1 - 4/\sqrt{n}$ , there exists a consensus algorithm achieving  $O(f)$  expected round complexity.

## REFERENCES

- [1] Tushar Deepak Chandra and Sam Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM*, vol. 43, no. 2, pp. 225–267, 1996.
- [2] Vassoso Hadzilacos and Sam Toueg, "Fault-tolerant broadcasts and related problems," in *Distributed Systems*, Sape Mullender, Ed., chapter 5, pp. 97–145. Addison-Wesley, 1993.
- [3] Hagit Attiya and Jennifer L. Welch, "Sequential consistency versus linearizability," *ACM Transactions on Computer Systems*, vol. 12, no. 2, pp. 91–122, 1994.
- [4] M. Herlihy, "Wait-free synchronization," *ACM Transactions on Programming Languages and Systems*, vol. 13, pp. 124–149, 1991.
- [5] R. Guerraoui, "Revisiting the relationship between non-blocking atomic commitment and consensus," in *Proc. of 9th International Workshop on Distributed Algorithms(WDAG)*, Sep 1995, vol. 972 of LNCS.

```

1: variable:
2:    $e_i$  : init  $v_i$  /* The  $v_i$  is the  $p_i$ 's proposal */
3:    $t_i$  : init  $-1$ 
4:    $V_i$  : init  $\emptyset$ 
5:    $dec_i$  : init  $\perp$ 
6:    $nop_i$  : init FALSE

7: for each round  $r = 0, 1, \dots$  do :
8:   /* Aggregation Phase */
9:   if  $(r \bmod 9) < 3$  then
10:    if  $(r \bmod 9) = 0$  then  $\text{FIB}_3((t_i, e_i, dec_i))$  endif
11:    if  $(t_j, v_j, dec_j)$  is received from  $p_j$  then
12:       $V_i \leftarrow (t_j, v_j)$ ;  $dec_i \leftarrow dec_j$ 
13:      if  $(r/9) = i$  then /* Coordinator's task */
14:         $nop_i \leftarrow \text{FALSE}$ 
15:        if  $|V_i| > n/2$  then
16:          let  $(t, v)$  be the pair such that
17:             $t$  is the maximum in  $V_i$ .
18:           $(t_i, e_i) \leftarrow (r/9, v)$ ;  $V_i \leftarrow \emptyset$ 
19:          else  $nop_i \leftarrow \text{TRUE}$  endif
20:        endif
21:      /* Impose Phase */
22:      elseif  $3 \leq (r \bmod 9) < 6$  then
23:        /* Coordinator's task */
24:        if  $(r/9) = i$  and  $nop_i = \text{FALSE}$  then
25:           $\text{FIB}_3((t_i, v_i, dec_i))$  endif
26:          if  $(t_j, v_j, dec_j)$  is received from a process  $p_j$  then
27:             $(t_i, e_i) \leftarrow (r/9, v_j)$ ;  $dec_i \leftarrow dec_j$  endif
28:        /* Commit Phase */
29:        elseif  $6 \leq (r \bmod 9) < 9$  then
30:          if  $(r \bmod 9) = 6$  then
31:             $\text{FIB}_3((t_i, e_i, dec_i))$ 
32:          endif
33:          if  $(t_j, v_j, dec_j)$  is received from  $p_j$  then
34:             $V_i \leftarrow (t_j, v_j)$ ;  $dec_i \leftarrow dec_j$  endif
35:          if  $\exists w$  such that
36:            more than  $n/2$  entries in  $V_i$  are  $(r/9, *)$  then
37:               $dec_i \leftarrow w$ 
38:            endif
39:          endif
40:          if  $dec_i \neq \perp$  then  $\text{decide}(dec_i)$  endif
41:           $V_i \leftarrow \perp^n$ 
42: endifor

```

Fig. 2. A consensus algorithm tolerating crashes and probabilistic omissions (Code for a process  $p_i$ )

- [6] D. Dolev, C. Dwork, and L. Stockmeyer, "On the minimal synchronism needed for distributed consensus," *Journal of the ACM*, vol. 34, no. 1, pp. 77–97, 1987.
- [7] D. Dolev, R. Reischuk, and R. Strong, "Early stopping in byzantine agreement," *Journal of ACM*, vol. 37, no. 4, pp. 720–741, 1990.
- [8] K. J. Perry and S. Toueg, "Distributed agreement in the presence of processor and communication faults," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 3, pp. 477–482, 1986.
- [9] M. Raynal P. R. Parvédy, "Optimal early stopping uniform consensus in synchronous systems with process omission failures," in *Proc. of the 16th annual ACM symposium on Parallelism in algorithms and architectures(SPAA)*, Jun 2004, pp. 302–310.