

*This problem set is about the round-model synchrony assumptions, early-stopping TRB, uniform TRB, and arbitrary (aka “Byzantine”) failures.*

*The first question concerns the round-model assumption that clocks are perfectly synchronized, and how one can weaken this assumption.*

1. The round-model of computation (assumed by the  $t + 1$ -rounds TRB algorithm that we gave in class) is based on the assumption of perfect synchrony:
  - The processes’ clocks are perfectly synchronized with one another and with real-time.
  - there is a known bound  $\delta$  such that a message sent at real-time  $t$  is received by real-time  $t + \delta$ .

Thus, all processes start a round at the same real-time and end it simultaneously  $\delta$  real-time units later. However, perfect clock synchronization is not achievable.

Suppose clocks are only  $\epsilon$ -synchronized, i.e. at all times the difference between two clocks is bounded by  $\epsilon$ . For example, a process may “think” that the current time is  $c$ , while another may “think” that the current time is  $c - \epsilon$ . Thus, processes do not necessarily start a round at exactly the same real-time.

- (a) Show (by giving a counter-example) that the  $t + 1$ -rounds TRB algorithm given in class, where the length of a round is  $\delta$ , does not work anymore.
- (b) Modify the algorithm so that it works when the clocks are  $\epsilon$ -synchronized. (Note: This is a very simple modification.)

*The next questions concern Uniform TRB — a stronger version of TRB, where the Agreement requirement is replaced with:*

*Uniform Agreement: If any process (whether correct or faulty) delivers a message  $m$ , then all correct processes eventually deliver  $m$ .*

2. Consider the  $t + 1$ -rounds TRB algorithm given in the class notes. Show that this algorithm does not solve the Uniform TRB problem in a system with process crashes (i.e., give an explicit execution of this algorithm that violates Uniform Agreement).

Modify the algorithm so that it *does* solve Uniform TRB in systems with process crashes, and prove it correct (you only need to give the parts of the proof that changed from the original proof).

3. Consider the problem of solving Uniform TRB in a round-based system with *general omission* failures.
  - (a) Assume that a *majority* of the processes are correct, i.e.,  $n > 2t$ . Describe a round-based algorithm that solves the above problem and prove it correct.

- (b) Prove that this problem cannot be solved if  $n \leq 2t$ . (Hint: partition the  $n$  processes into two sets of size at most  $t$ , and consider several scenarios.)

*The next question concerns early-stopping TRB.*

4. Consider the early-stopping algorithm that we saw in class for TRB.

Replace the statement:

“**else if**  $|quiet^i| < i$  **then deliver SF**” with

“**else if**  $|quiet^i| = |quiet^{i-1}|$  **then deliver SF**”

In this protocol, a process halts if it does not see any *new* failures in a round. Thus, if all  $f$  failures occur in the first  $r$  rounds, every correct process delivers by round  $r + 2$  and halts by round  $r + 3$ .

- (a) Prove this is a  $t$ -tolerant early-stopping TRB protocol for crash failures.  
(b) Prove it is *not* a correct  $t$ -tolerant TRB protocol for send-omission failures.

*The last question is about arbitrary failures and a variation of TRB.*

5. Consider a variation of TRB known as the **crusader agreement problem**. In this problem, there are  $n$  processes, one of which is a sender with a message it wants to broadcast (different from the special value **SF**). Of the  $n$  processes, at most  $t$  may fail and the others are correct. The problem is to come up with an algorithm that achieves the following properties:

- **Termination:** Each correct process delivers a message.
- **Validity:** If the sender is correct, then all correct processes deliver the sender’s message.
- **Weak agreement:** If two correct processes deliver messages  $m$  and  $m'$ , and neither  $m$  nor  $m'$  is **SF**, then  $m = m'$ .

Informally, when the sender is faulty, this problem allows some correct processes to deliver **SF** and other correct processes to deliver some non-**SF** message. Correct processes, however, may not deliver different non-**SF** messages.

- (a) For the synchronous round model *with arbitrary failures* prove the following:
- i. If  $n > 3t$ , there is a two-round algorithm for the crusader agreement problem.
  - ii. If  $n > 3t$ , there is no one-round algorithm for the crusader agreement problem.
  - iii. If  $n \leq 3t$ , there is no algorithm for the crusader agreement problem (no matter how many rounds we allow).
- (b) For the synchronous round model *with arbitrary failures and message authentication*, give a 2-round algorithm for the crusader agreement problem that works for all  $n$  and  $t$  such that  $n \geq t \geq 1$ .