

TD 13: Approximate Agreement, Renaming in AMP

In this TD, assume that there are n processes p_1, \dots, p_n and that the model of communication is the Asynchronous Message Passing model (AMP). Assumptions on the AMP model:

- All processes know n . If we assume that the model has at most f crash failures, the processes know f as well.
- Processes can send messages to everybody else.
- There is no upper bound on the time it takes for a message to be received, and that messages sent to non-faulty processes (processes that do not crash) are always received.

1 Approximate Agreement

Assume that at most f processes may crash. As shown during the class, consensus cannot be solved in the AMP model with even one crash failure. However, it is possible to solve a close variant of the consensus task, which is known as the ε -approximate agreement task. Any algorithm solving this task must satisfy the following conditions:

Processes start with real input values in $[0, 1]$, and must output values in $[0, 1]$, satisfying

- **ε -Agreement:** If non-faulty processes p_i and p_j decide v_i and v_j , then $|v_i - v_j| \leq \varepsilon$.
- **Validity:** If a non-faulty process (i.e., a process that does not crash) outputs v , then v must belong to the convex hull of all the processes inputs (even those of faulty ones), i.e., there exist processes with inputs c_1 and c_2 such that $c_1 \leq v \leq c_2$.
- **Termination:** All non-faulty processes decide.

The goal is to design an algorithm that solves ε -approximate agreement resilient to at most f crash failures.

Definition. (Averaging function) Define function avg_f so that, given a set $V = \{v_1, \dots, v_N\}$ where $v_1 \leq \dots \leq v_N$,

$$\text{avg}_f(V) = \frac{v_1 + v_{f+1} + \dots + v_{(\lambda-1)f+1}}{\lambda}$$

where $\lambda = \lceil N/f \rceil$.

† **Problem 1.1.** Suppose V, W , and U are nonempty multisets with $|V| = |W| = m$, $V \subseteq U$, $W \subseteq U$, and $|V - W| = |W - V| \leq f$. Show that

$$|\text{avg}_f(V) - \text{avg}_f(W)| \leq \frac{\max(U) - \min(U)}{\lceil m/f \rceil}.$$

Hint: Show that $\max(v_{if+1}, w_{if+1}) \leq \min(v_{(i+1)f+1}, w_{(i+1)f+1})$ where $V = \{v_1 \leq \dots \leq v_m\}$ and $W = \{w_1 \leq \dots \leq w_m\}$.

† **Problem 1.2.** Let U be the set of initial values of all the processes. Give an algorithm \mathcal{A} which satisfies the following: If U' is the set of output values of the decided processes, then

$$\max(U') - \min(U') \leq (\max(U) - \min(U)) \left\lceil \frac{n-f}{f} \right\rceil^{-1},$$

and moreover, $[\min(U'), \max(U')] \subseteq [\min(U), \max(U)]$.

Problem 1.3. Modify \mathcal{A} to obtain an ε -approximation algorithm. How large can f be with respect to n ?

2 Renaming Task

Surprisingly, even though consensus is not solvable in the AMP model, there is a task which is the complete opposite of the consensus task and is solvable:

Definition 2.1 (Renaming task). The renaming task with *initial name space* of size M (one can have $|M| = \infty$) and *new name space* of finite size N and $M > N$ is defined as follows.

Every process initially has as input a *distinct* identifier from the initial name space. Any algorithm solving the renaming task must guarantee that:

- Processes that decide must all output distinct identifiers.
- All non-faulty processes must decide.

Note: Even though we shall refer to the processes as p_1, \dots, p_n , the processes themselves are not aware of this indexing, otherwise one would have a trivial algorithm where p_i outputs i .

Consider the following algorithm where processes have distinct initial identifiers from an *unbounded* domain. We also assume that $f < n/2$ where f is the maximum possible number of crash failures.

Algorithm at process p_i with initial identifier id_i

0. Initialize set $known \leftarrow \{id_i\}$ and counter $c \leftarrow 0$
1. a. Send $known$ to every other process
b. $c \leftarrow 1$
2. Wait until you receive a message V' :
 - a. If $V' \subsetneq known$ then goto 2
 - b. If $V' \setminus known \neq \emptyset$ then: $known \leftarrow known \cup V'$, goto 1
 - c. If $known = V'$ then:
 - a. $c \leftarrow c + 1$
 - b. If $c < n - f$ then goto 2 else goto 3
3. a. $v \leftarrow |known|$
b. Let r be the rank of id_i in the (sorted) set $known$
c. Choose your new name to be the pair $\langle v, r \rangle$.
4. Continue forever; whenever you receive a message V' such that $V' \setminus known \neq \emptyset$:
 - a. $known \leftarrow known \cup V'$
 - b. Send $known$ to every other processor

Problem 2.2. Try to understand what the algorithm is doing, in particular at steps 2 and 3.

† **Problem 2.3.** The processes are clearly exchanging certain sets. A set V is *stable* if there is some process p_i that has received $n - f - 1$ messages containing identical copies of V while the value of its $known$ variable is also V .

Show for $f < n/2$ that the set of stable sets is totally ordered (i.e., every stable set is contained in some other stable set) in any run of the algorithm.

†† **Problem 2.4.** Show for $f < n/2$ that every non-faulty processor eventually obtains a stable set V . Conclude that all such processors decide.

† **Problem 2.5.** Show for $f < n/2$ that the new names chosen by the non-faulty processes are distinct. What is the size of the new name space?

††† **Problem 2.6.** Show that for any $N \geq n + 1$, if $f \geq n/2$, then there is no algorithm for solving the renaming task with new name space of size N . *Note:* Even having an intuition for this is enough.