

This problem set is about the *Coordinated Attack* problem. Every process starts with an initial input of 0 or 1. Each process can output a decision value of 0 or 1 (once a process decides, it cannot “change its mind”). Processes must satisfy the following properties:

- *Uniform Agreement*: No two processes decide differently.
- *Validity*:
  - (a) If all processes start with 0, then no process decides 1.
  - (b) If all processes start with 1, and there are no failures, then no process decides 0.
- *Termination*: All nonfaulty processes eventually decide.

In class we considered the above problem in a simple message-passing system  $\mathcal{S}$  consisting of two processes  $p$  and  $q$  connected by a (bidirectional) communication link such that (a) processes proceed in synchronous rounds, (b) processes do not fail, and (c) any number of messages can be lost. Note that under the assumption (b) that all processes are nonfaulty, Termination reduces to:

- *Termination*: All processes eventually decide.

In class we proved that the Coordinated Attack problem cannot be solved in system  $\mathcal{S}$ .

In questions 1, 2 and 3 of this problem set, we consider some variations of the CA problem, under the system  $\mathcal{S}$  defined above.

In question 4, we consider the CA problem under a slightly stronger model (where the communication link between  $p$  and  $q$  is lossy but “fair”).

1. Consider the following validity requirement:

- *Strong Validity*:
  - (a) If all processes start with 0, then no process decides 1.
  - (b) If all processes start with 1, then no process decides 0.

Give a *simple* proof that the problem consisting of *Agreement*, *Strong Validity*, and *Termination* cannot be solved in system  $\mathcal{S}$ . Do not use the result that we gave in class (your proof must be “from scratch”).

2. Consider the following termination requirement:

- *Weak Termination*: If there are no failures, then all processes eventually decide.

Is the problem consisting of *Agreement*, *Validity*, and *Weak Termination* unsolvable in system  $\mathcal{S}$ ? If yes, then give an impossibility proof; otherwise give an algorithm that solves it in system  $\mathcal{S}$ .

3. Consider the following requirement:

- *Unanimous Termination*: If any process decides, then all processes eventually decide.

Is the problem consisting of *Agreement*, *Validity*, *Weak Termination*, and *Unanimous Termination* unsolvable in system  $\mathcal{S}$ ? If yes, then give an impossibility proof; otherwise give an algorithm that solves it in system  $\mathcal{S}$ .

4. In system  $\mathcal{S}$  (which was assumed in the previous questions) there are no restrictions on message losses. We now consider a system  $\mathcal{S}'$  that differs from  $\mathcal{S}$  in that the communication link between  $p$  and  $q$  is *fair*: messages can get lost, but if  $p$  sends a message  $m$  to  $q$  infinitely often then  $q$  receives  $m$  infinitely often, and symmetrically for messages sent by  $q$  to  $p$ .

- (a) An algorithm is called *quiescent* if eventually no messages are sent. Give a quiescent algorithm that solves the original Coordinated Attack problem (i.e., *Agreement*, *Validity*, and *Termination*) in system  $\mathcal{S}'$ .
- (b) Explain exactly where the impossibility proof given in class for system  $\mathcal{S}$  breaks down when we assume that communication links are *fair*, i.e., if we try to apply it to system  $\mathcal{S}'$ .
- (c) An algorithm is called *halting* if eventually all processes reach a “halting” state from which they cannot execute any further steps. Obviously, a halting algorithm is quiescent. The converse is not necessarily true. A quiescent algorithm can reach a state where all processes are “idle” but not halted: each process reaches a state where it is ready to respond to a message if it ever gets one, but no messages are in transit, and so it remains in that idle state forever.

Give a halting algorithm that solves the Coordinated Attack problem in system  $\mathcal{S}'$  or prove that no such algorithm exists.