3 Byzantine Agreement

The most thoroughly studied problem in distributed computing is Byzantine Agreement, also known as the *consensus* problem.

We assume a system of n processors, $p_1, \ldots, p_n$, some number t of which may fail in an arbitrary fashion (Byzantine failures). Each processor p_i has an initial vote $v_i \in \{0, 1\}$. At some point in the computation each processor must irreversibly *decide* on a value (formally, enter one of two possible decision sates d_0, d_1). We require:

- agreement: all non-faulty processors decide on the same value;
- validity: if all non-faulty processors begin with the same value, say, v, then all non-faulty processors must decide v.

We will assume the processors operate in lock-step synchrony, with all messages taking exactly one time unit to be delivered. Thus, a message sent at one step will be received at the next step.

Theorem 3.1 *Any t-resilient protocol for Byzantine agreement, for $t \geq 1$, requires at least $3t + 1$ processors.*

Proof: For the case $n = 2$ there is clearly no solution (either party could be faulty; then consider the case in which the two parties start with different values).

The proof for $n > 2$ is in two parts. In one part, we show there is no 3-processor agreement protocol that tolerates a single faulty processor. In the other part we show that if for some $t > 1$ there exists a t-resilient agreement protocol requiring at most

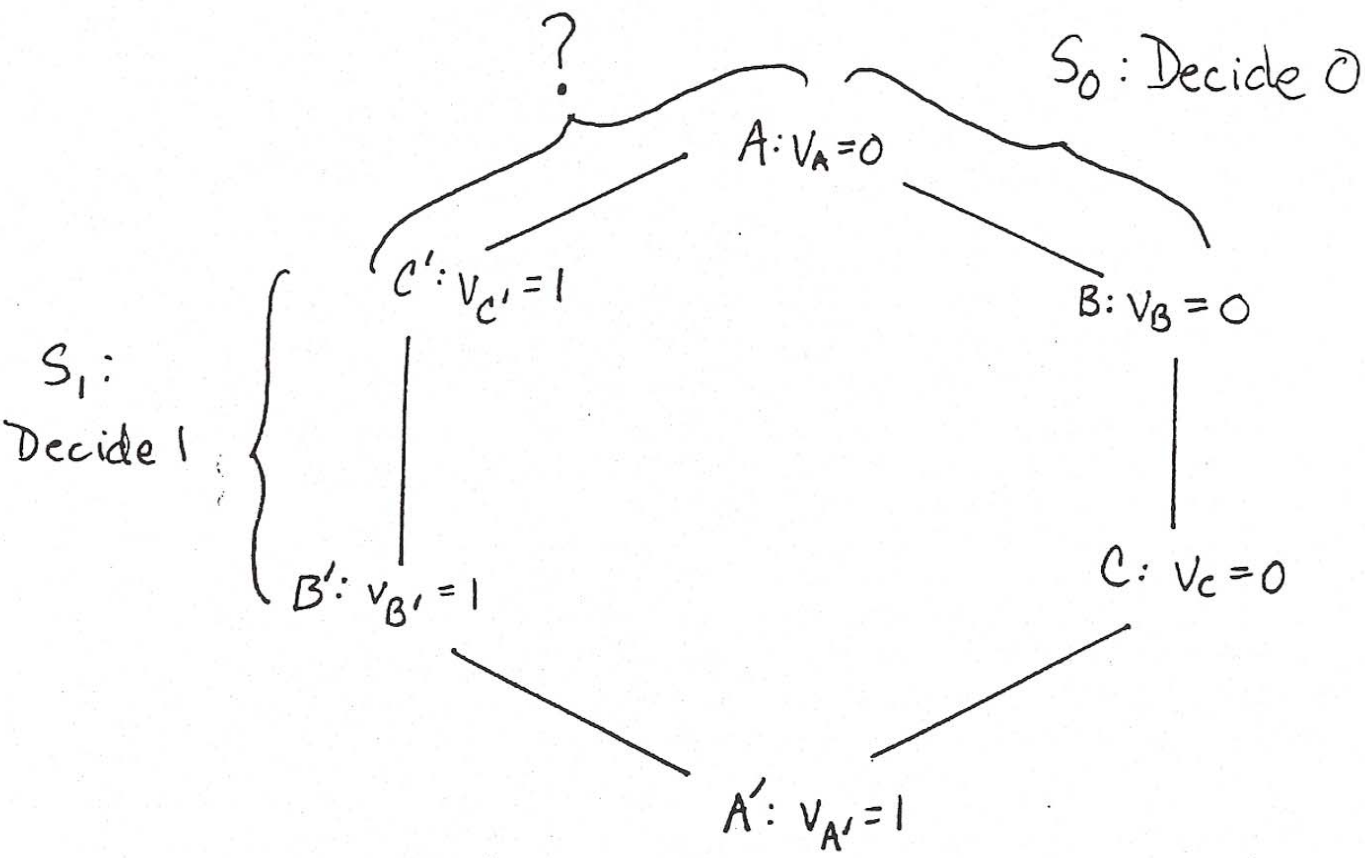

Figure 2: Impossibility of 1-resilient 3-processor Agreement

$3t$ processors, then there is 1-resilient, 3-processor protocol. Thus, there can be no t-resilient protocol requiring at most $3t$ processors.

For the first part, assume for the sake of contradiction that there exists a 1-resilient 3-processor protocol. Let the three processors be A, B, C. Let us make two copies of each processor, and call the second copies A', B', C', respectively. Let $v_A = v_B = v_C = 0$, and $v_{A'} = v_{B'} = v_{C'} = 1$, and arrange the copies as shown in Figure 2. Note that to each processor it looks as if it is in the original 3-processor system.

Consider the scenario in which A and B are non-faulty, with initial values 0, and in which C is faulty and behaves towards A as if its input were 1, while behaving toward B as if its input were 0. Formally, this is captured (see Figure 2) by connecting the processors $C' - A - B - C$. To A and B it appears as if they are in a three-processor system in which C is faulty. Thus, the Byzantine agreement protocol will eventually yield decisions of 0 for both B and C. Let us call this scenario S_0.

Next, consider the processors $A' - B' - C' - A$. To B' and C' it appears that they are running in a 3-processor system in which A is faulty, behaving toward B' as if its input were 1, while behaving toward C' as if its input were 0. Thus, in this system B' and C' must decide on the value 1. We call this scenario S_1.

Finally, consider processors $B' - C' - A - B$. Since A cannot distinguish this scenario from S_0, A will decide 0. Since C' cannot distinguish this scenario from S_1, C' will decide 1. This violates the agreement condition. Thus, there is no 1-resilient 3-processor protocol for Byzantine agreement.

Now, suppose there were a t-resilient agreement protocol requiring $m \leq 3t$ processors, for $t \geq 2$. Split the processors into three sets, A, B, and C, of size at least 1 and at most t each. Define a 3-processor agreement protocol as follows: p_A simulates all the transitions and transmissions of processors in A, p_B does the same for B, and p_C for C. In particular, they simulate the execution in which every processor in A has the same initial value as p_A, and similarly for B and C. Messages within a subset are simulated, and messages between subsets are sent explicitly.

The simulation is a 3-processor protocol. The failure of any one processor corresponds to a failure of at most t processors in the original system, because each set contains at most t processors. By the assumed t-resilience of the original protocol, the simulation works correctly in the presence of any single processor failure, contradicting the result in the first part of the proof. ∎