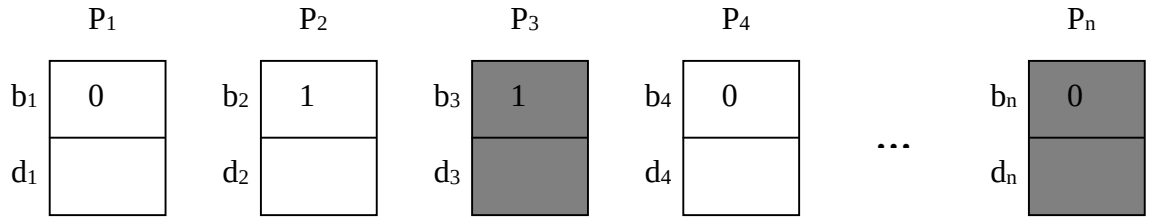
CS174 Lecture 18

Byzantine Agreement

This lecture, we describe the *Byzantine agreement* problem. Given n processors, the goal is to have all processors agree on a binary decision. This is a basic and deceptively difficult task in distributed computing. The difficulty comes because in practice, we must assume that some processors may be faulty or even *malicious*. That is, the bad processors may deliberately try to produce an incorrect outcome. The situation is shown below:



Each processor has a bit b_i that stores a value which is 0 or 1. There is also a variable d_i which will eventually hold the processor's decision. In this application, the decision will be a binary value. Some number of processors may be bad, and these are shown in gray. The objective of the protocol is that

1. All good processors terminate with the same decision.
2. If all the good processors begin with the same value v in b_i, they should all terminate with v as their decision.

Item 2 is needed because a trivial protocol for objective 1. would be "output 0". Objective 2 ensures that the good processors can actually make and agree on computational decisions. You will see later that the algorithm we describe does better than 2. suggests. With a strong enough majority vote (but not a unanimous vote) for v among good processors, all processors will output v after two more rounds.

Rounds

The agreement protocol proceeds in *rounds*. In one round every processor sends a message to every other processor. Therefore, each processor receives a message from every other processor. The messages sent by one processor in a round need not be the same, and can depend on the destination.

All good processors follow the protocol given on the next page. Agreement is reached when conditions 1. and 2. above are met. Agreement should always be reached in a finite number of steps no matter what the faulty processors do.

Global Coin Toss

In addition to the processors, there is a trusted processor that performs a *global coin toss*. That is, it fairly tosses a coin and broadcasts the results to all the processors. In practice this function does not require a trusted processor and can be distributed among the processors P_i, but this makes the protocol somewhat more complicated.

The agreement protocol is basically an iterated voting protocol. Processors all vote, and the good processors adjust their votes in the direction of the majority. If enough of the processors are good, those processors eventually converge on the majority vote.

The protocol requires three quantities, L, H, and G

$L = (5n/8) + 1$

$H = (3n/4) + 1$

$G = (7n/8)$

The protocol is

Input: A value b_i.

Output: A decision d_i.

1. vote = b_i.
2. For each round, **do**
    3. Broadcast vote.
    4. Receive votes from other processors.
    5. *maj* ← majority value among votes received including own vote.
    6. *tally* ← number of occurrences of *maj* among votes received.
    7. **If** *coin* = HEADS **then** *threshold* ← L;

       else *threshold* ← H;
    8. **If** tally ≥ *threshold* **then** *vote* ← *maj*;

       else *vote* ← 0;
    9. **If** *tally* ≥ G **then** set d_i to *maj* permanently.

We will assume that $t < n/8$. First notice that if all the processors start with the same initial value, their final decision is the same. That's because n-t processors will broadcast the same vote, and this vote will be the majority value of all those good processors. That is, tally will be $> 7n/8$. It

doesn't matter whether *threshold* is set to H or L at step 7. In either case, tally > threshold, so vote is set to the majority (which is the same as it was). Finally, at step 9, the tally is greater than G, so each good processor halts with its decision set to the majority.

So if all the good processors agree initially, the protocol halts after one round.

If the processors disagree initially, then we have to consider two possibilities. First of all, some of the good processors may disagree about what the majority is. Note that this can only happen if some faulty processor broadcasts different values to the good processors (which is legal).

Suppose any two good processors come up with a different value for maj at step 5. That means that the number of good processors that voted 0 or 1 is $\leq n/2$ in both cases. Otherwise, they would have forced the majority to be 0 or 1 for all the good processors. No matter what the bad processors do, the value of tally on good processors can be at most $n/2 + n/8$, which is $5n/8$ at step 6. That value is less than both H and L, so step 8. always causes vote to be set to 0 for all good processors. Then on the next round, all the good processors will agree and by the previous analysis, the protocol will terminate after one round. So it takes a total of two rounds to reach a decision if there is disagreement about the majority value.

It remains to consider the case where all good processors agree on the majority value maj. What happens next depends on the comparison of each processor's *tally* with the *threshold* value at step 8. Note that the threshold value is the same for all good processors, since it comes from the trusted coin toss. Suppose the result of this comparison is the same for all good processors. Then the vote variable on every processor is set to maj or to 0. In either case, they all agree after this round, and the protocol will terminate on the next round.

But it is possible that the comparison at step 8. yields different results for different processors. For example, if the number of good processor votes for the majority is between $n/2+1$ and $(5n/8)$, and if the coin toss causes L to be the threshold, then the faulty processors can cause some of the tests at step 8 to succeed and others to fail. In this case, we say that the faulty processors *foil* the threshold L. If the number of good processor votes is between $(5n/8)+1$ and $(3n/4)$, and the coin toss causes H to be the threshold, then the faulty processors can foil the threshold H and cause some step 8. tests to succeed and others to fail.

But notice that the ranges of good processor votes needed in each case are disjoint. That is, there can be at most $(5n/8)$ good processor votes for the majority in the case where threshold L is foiled, and at least $(5n/8)+1$ votes for the majority in the case where H is foiled. So there is only one toss of the coin that can allow a threshold to be foiled on a given round. The probability of this happening is ½, because H or L is decided by the coin toss.

The expected number of steps before no threshold is foiled (exponential r.v.) is therefore $1/p$ which is 2. Once this happens, i.e. once there is a round where no threshold is foiled, all the step 8. tests agree. It follows that all good processors end up with the same value after that round.

If you take the maximum of all these cases, it follows that the expected number of rounds before all good processors agree is 2.