Note: These notes are based on material from James Aspnes textbook (see reference below) and [1]

1 The Consensus Problem

In the *Consensus Problem*, every process starts with a single input bit. We require that all non-faulty processes terminate and also:

- **Agreement:** All non-faulty processes decide the same value
- **Validity:** The value decided must equal the input bit of some non-faulty process

1.1 Problem Model

We assume *synchronous communication*: there is some known bound on how long it takes any message to get from a sender to a receiver. In particular, an upper bound (Δ) on the message transit time is known to all processors. Hence, the processors can operate in rounds of duration Δ.

We will assume that up to f processors can fail. When a process fails, one or more of its outgoing messages are lost in the round it fails in and no processes ever hear from it in subsequent rounds.

2 The Dolev-Strong Flooding Algorithm

This flooding algorithm is due to Dolev and Strong (see refs below). It runs in $f + 1$ rounds.

2.1 The Algorithm

Each process keeps a single set of (process, input) pairs. For process p with input b, this is initially just the set $\{(p, b)\}$. In round r, I broadcast my set to everyone, and then take the union of my set and all the sets I receive. At round $f+1$, I decide on $F(S)$, where the function f is any deterministic function that outputs 0 if all input bits in the pairs in S are 0 and 1 if all input bits in the pairs in S are 1; for all other inputs, it can output either 0 or 1. The same function $F()$ must be used by all processes to ensure agreement. For example, $F()$ could output the majority bit in S, could output the bit held by the smallest ID process, could take the minimum of all bits, etc.

2.2 Analysis

Lemma 1. *After $f + 1$ rounds of Dolev-Strong, all non-faulty processes have the same set, and so will solve consensus.*

Proof: Let S_i^r be the set for process i after r rounds. We'll show that if there are no faults in round k, then $S_i^r = S_j^{k+1}$ for all processes i, j and rounds $r > k$.

To see this, fix some round k in which there are no faults and let L be the set of processes that are alive at the start and end of round k. Let $S = S^{k+1} = \cup_{p \in L} S_p^k$. Since all messages sent in round k are received, it means that for any process $p \in L$, $S_p^{k+1} = S$. So after round k, all processes have the same set S. Thus, in subsequent rounds, all processes will take the union of a bunch of sets equal to S, which union will be S. Hence, both $S_i^{k+1} = S$ for all processes j, and also $S_i^r = S$ for all processes i and rounds $r > k$.

Since all good processes have the same set S, their outputs, which is $F(S)$ will satisfy agreement and validity. □

How many total messages does this algorithm send if $f = \Theta(n)$? How many total bits are sent?

2.3 Authenticated Dolev-Strong Flooding

A *Byzantine fault* on a process means that the process can send out arbitrary messages that try to thwart the algorithm. This is in contrast to the crash faults discussed above where a faulty process simply stops sending messages. If a process does not suffer a Byzantine fault, we call it *good*

If we have access to cryptography, then we can augment the Dolev-Strong algorithm to tolerate $f < n/2$ Byzantine faults and run in $f+1$ rounds. In particular, we must assume the existence of a *public key infrastructure (PKI)*: every process knows the public key for every other process. In this algorithm, the function F taken over the set of accepted input bits must be the majority function.

The main idea of the algorithm is:

- All messages include: an input bit of some process; the path travelled by the message; and digital signatures along that path. For example, a value v_1 that is input to process 1 and reached you via steps through process 2 and process 3, will arrive as the message $(v_1, 123, S_3(S_2(S_1(v_1)))$
- A process only accepts messages in round r if they are digitally signed by r processes. It signs and resends all accepted messages.
- At the end of round $f+1$, each process creates a set S of all input bits accepted in that round. Then, the process outputs $F(S)$.

Theorem 1. *Assume $f < n/2$. Then, after $f+1$ rounds of Authenticated Dolev-Strong, all good processes solve Byzantine consensus.*

Proof: First, we show that if a message containing process p's signed input bit is ever sent out by a good process, process p's signed input will be in the final S set of all good processes. Let q be the good process that sends out p's bit in some round r. If $r = 1$, then $q = p$, and p sends it's input bit in round 1, and this is accepted by all other good processes since it contains $r = 1$ signature. If $r > 1$, then $q \neq p$, and q accepted p's input bit in round $r-1$ because it was signed by $r-1$ processes. In round r, q also signed the input bit, and sent it to all good processes. So these processes will accept it in round r. Since the number of good processes is greater than $f+1$, by induction on the round number, some message containing p's signed input will be accepted by all good processes in round $f+1$.

Second, any bit of some process p that a good process accepts in round $f+1$ is passed through $f+1$ processes, at least one of which is good. So, there was some round r in which a good process sent out p's bit.

Finally, observe that every good process's bit is sent out by a good process in round 1. Hence all input bits of good processes are in the S sets of all good processes.

Thus, the S sets of all good processes are: (1) the same; and (2) contain the input bits of all good processes - and maybe some bad. Since the F function takes the majority of the bits in S and $f < n/2$, all good processes will satisfy agreement and validity. □

What is the number of messages and number of bit sent in this algorithm? Do you see how to reduce the number of messages and message sizes?

3 References

- "Authenticated Algorithms for Byzantine Agreement" by Dolev and Strong. https://www2.imm.dtu.dk/courses/02220/2015/L12/DolevStrong83.pdf
- "Notes on Theory of Distributed System" by James Aspnes. https://www.cs.yale.edu/homes/aspnes/classes/465/notes.pdf

References

[1] Garay, J., Kiayias, A., and Leonardos, N. The bitcoin backbone protocol: Analysis and applications. *Journal of the ACM 71*, 4 (2024), 1–49.