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 S, which will just 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 $f+1$, the process creates a set S of all input bits accepted in any of the rounds. Then, the process outputs $F(S)$.

Theorem 1. *After $f+1$ rounds of Authenticated Dolev-Strong, all good processes solve consensus.*

Proof: First, we show that if a message containing process p's input bit is ever sent out by a good process, process p's bit 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.

Second, 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.

Finally, any bit of some process p that a good process accepts in round $f+1$ 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.

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. □

3 Blockchain Consensus

The set of processes are divided into *good*: follow the protocol; and *bad*: may deviate from protocol.

The algorithm below is a simplification of blockchain consensus. A round consists of a time period for mining (say 10 minutes) in which one solution is accepted.

1. $C \leftarrow$ some initial chain
2. For $r \leftarrow 0 \ldots \infty$
   (a) Let x be the block I want to add to chain C
   (b) Repeat for this round
      i. Choose a random y
      ii. If $h(C, x, y) \leq D$ then: $C \leftarrow Cx$; Broadcast C along with y. End the round.
3. $C \leftarrow$ longest valid chain received this round

Let G be the set of good processes, and B be the set of bad. For process i, let p_i be the expected number of puzzles solved by i in one round.

Let $\alpha = \sum_{i \in G} p_i$ and $\beta = \sum_{i \in B} p_i$.

Theorem 2. *Assume $\beta \leq 1/2(\alpha - \alpha^2)$. Then, with probability at least $1 - O(1/m)$, all but at most the last $17(\alpha - \alpha^2) \log m$ blocks in the longest blockchain are valid.*

Proof: By Markov's inequality, the probability that a good process adds a block in a round at least equals the expected number of blocks added (i.e. p_i). In the following, when we say "add" a block, we mean add a block to some chain, not necessarily the one that eventually is the longest.

Then, inclusion-exclusion says that the probability that the good processes add at least one block in round r is

$$\sum_{i \in G} p_i - \sum_{i,j \in G} p_i p_j \geq \alpha - \alpha^2$$

(Do you see why $\sum_{i,j \in G} p_i p_j \leq \alpha^2$?)

Let X_r be an indicator r.v. that is 1 if the good processes add a block in round r. Then by linearity, the good processes add an expected $\sum_{r=1}^{m} E(X_r) = m(\alpha - \alpha^2)$ blocks in m rounds.

Let Y_i be an indicator variable for the success of the i-th puzzle attempt by a bad process. Then, by linearity, the expected number of blocks added by bad processes in m rounds is is $\sum_i E(Y_i) = m\beta$ (this last summation is over the number of rounds times the number of attempts per round).

Let X be the number of blocks added by the good and Y be the number of blocks added by the bad. By above, we'll assume that $E(X) = m(\alpha - \alpha^2)$ and $E(Y) = m\beta$. Both X and Y are the sum of $0-1$ independent random variables. So by Chernoff bounds, for any $\delta > 0$,

$$Pr(X \leq (1-\delta)m(\alpha - \alpha^2) \leq e^{-\delta^2 m(\alpha - \alpha^2)/2}$$
$$Pr(Y \geq (1+\delta)m\beta) \leq e^{-\delta^2 m\beta/2}$$

Let $k = m(\alpha - \alpha^2) = E(X)$. Then $E(Y) = m\beta \leq k/2$. Set $\delta = 1/3$ in the above and we get:

$$Pr(X \leq (2/3)k) \leq e^{-k/18}$$
$$Pr(Y \geq (2/3)k) \leq e^{-k/36}$$

The above pessimistically assumes that if two good blocks are mined in the same round, neither is accepted*

By a union bound, with probability of failure at most $e^{-k/18} + e^{-k/36} \leq e^{-k/17}$, neither of these events happen, and then the number of good blocks is larger than the number of bad blocks. In this case, any invalid blocks must be in the last $(2/3)k$ blocks of the blockchain. □

4 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.