0368.3311 The Theory Behind Blockchains March 13

Classical Consensus

Lecturer: Nir Bitansky

In the first lecture, we've seen that the main question in having a decentralized blockchain is

how to agree on which blocks to add.

Here the goal is to design a *consensus protocol*, which allows a bunch of parties maintaining together a blockchain to exchange messages in order to agree on a block to be added. The challenge is that some of the parties may be dishonest and not follow the protocol. For example, in the context of digital currency, malicious parties may try to keep off the chain payments for their competitors. Of course that if all or even most parties are malicious there's no meaning to agreement. We'd like to design protocols for a reasonable case where almost all or at least most parties are honest.

The consensus problem has in fact been extensively studied in the context of fault tolerant distributed systems. It is pretty challenging even when parties (aka nodes) are not malicious, but just crash every now and then. In what follows, we will give some basic background about this problem and figure out if and how it fits in the context of decentralized blockchains.

1 The Byzantine Generals Problem

The consensus problem is commonly known as the Byzantine generals problems and was introduced in a seminal work by Pease, Shostak, and Lamport [PSL80].

The story. A group of (Byzantine) generals would like to jointly decide whether to attack or retreat, where each one of them has its own opinion. Regardless of their opinion, non of them is willing to attack unless the majority of generals are going to attack (a feeble attack is worst than retreat.) The generals are separated and can only communicate using smart phones. To make things worst, some of the generals are treacherous and may communicate contradicting opinions to different generals, trying to lead to a feeble attack in which some (minority) of the generals will be destroyed.

The formal setting. We have n parties (or nodes) $P_1, \dots, P_n$ with respective inputs bits $x_1, \dots, x_n \in \{0,1\}$ corresponding to retreat/attack. The parties are pairwise connected (but there is no common bulletin board or something of that sort).

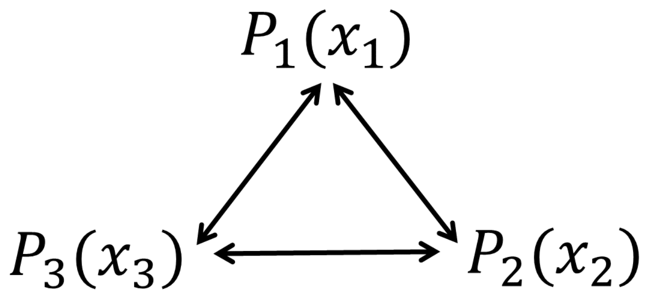

Figure 1: In the Byzantine generals parties $P_1, \dots, P_n$ each have their own input bit $x_1, \dots, x_n$ (above $n = 3$) and are pairwise connected to all other parties.

We want to guarantee:

- **Agreement:** all honest parties output the same bit y.
- **Validity:** if there is a bit y such that the input of all honest parties are y, they output y.

Is it possible to construct such protocols? under what conditions? how efficient can they be? There are certain well studied aspects that affect the answers to these questions:

- The number of dishonest parties t (or rather its fraction t/n).
- Dishonest parties vs crashing parties.
- What is the communication model (synchronous/asynchronous).
- Can parties use randomness?

2 Impossibility Without a Large Majority

One thing that we already understand is that there's no meaning to an agreement protocol in a setting where there's no honest majority, i.e. $n/2 > t$. It turns out that in general an even larger majority is required.

Theorem 2.1 ([PSL80])**.** *There is no Byzantine agreement protocol unless $n/3 > t$.*

Proof. We will now prove this theorem. We shall focus on the case that $n = 3$. In the homework you will generalize this for any n. Here we would like to show that there's no BA even of only a single party is dishonest.

Assume that a BA for $n = 3, t = 1$ exists. We will prove that if the protocol satisfies the validity property than it does not satisfy the agreement property.

We consider a setting where we have three parties A^*, B, C where A^* is malicious and B and C are honest parties with respective inputs 0 and 1.

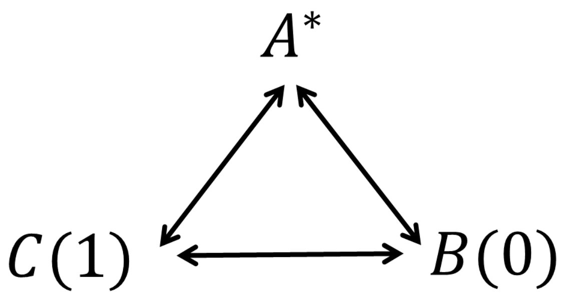

Figure 2: A^* is malicious and B and C are honest, but with opposite inputs.

Now imagine an adversary A^* that runs two honest copies $A(0)$ and $A(1)$ of the protocol. With B, A^* acts as if it has input 0, and with C, A^* acts as if it has input 1. (Incoming messages from B and C are fed to both $A(0)$ and $A(1)$. The outgoing messages of $A(0)$ are forwarded to B and the outgoing messages of $A(1)$ are forwarded to C.

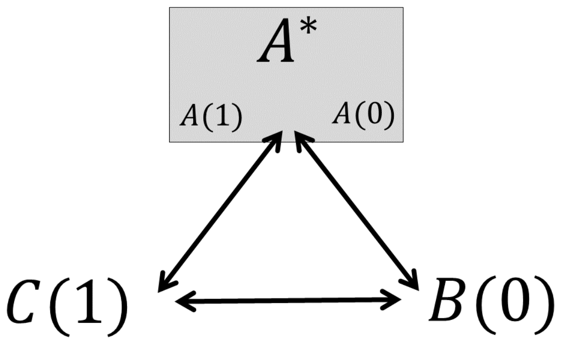

Figure 3: A^* emulates two honest executions of the protocol, one with input 0 and otehr with input 1.

Claim 2.2. *The honest parties output opposite values: B outputs 0 and C outputs 1.*

To prove the claim, let us look at the protocol from the perspective of B. We claim that what B sees in the protocol is identical to what it sees in a protocol where A is honest and C is replaced by a malicious party C^*.

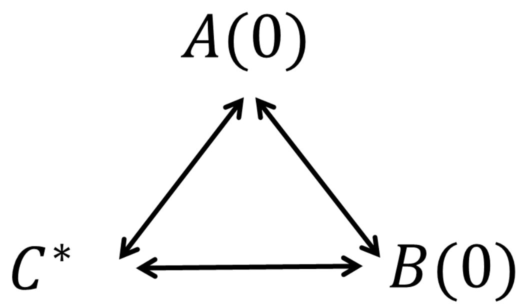

Figure 4: From B's perspective A is honest with input 0 and C^* is malicious.

The adversary C^* runs the honest C with input 1, but it ignores messages from $A(0)$. Instead, C^* runs its own copy of $A(1)$ and replaces the messages from $A(0)$ to $C(1)$ with the messages from its own copy $A(1)$. Due to the validity property, in this case B will output 0 (both honest parties agree on 0). By the way we defined C^* and A^*, B indeed has the same view in the two cases; therefore, B outputs 0 also in the original protocol.

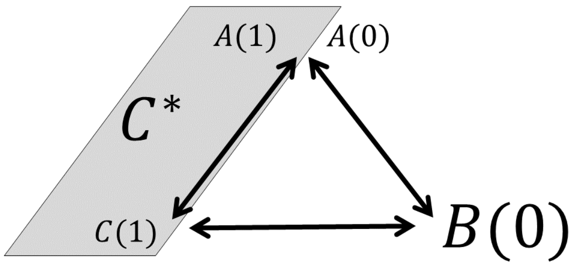

Figure 5: C^* acts like $C(1)$, but replaces messages from $A(0)$ with messages from its own copy of $A(1)$.

Symmetrically, we can show that C outputs 1 in the original protocol. All in all, in the original protocol the two honest parties B and C output opposite bits in contradiction to the agreement requirement.

□

In the homework, you will show how to extend this lower bound to any n and $t \geq n/3$.

3 Beating the Lower Bound Using Digital Signatures

It turns out that if we allow a digital signature setup and assume computationally bounded adversaries, we can do better than the above lower bound. We will see a protocol by Dolev and Strong [DS83] that works for any honest majority namely $n/2 > t$.

The broadcast problem. It will actually be easier to consider a related problem called *broadcast*. In this problem, one party is a designated *sender* (aka commander) P_1 with a message m that they would like to broadcast to all receiver parties (aka lieutenants) $P_2, \ldots, P_n$.

We want to guarantee:

- **Agreement:** all honest receivers output the same message m'.
- **Validity:** if the sender is honest and broadcasts m then the honest receivers output $m' = m$.

Claim 3.1. *Assume that we can solve the broadcast problem for $t < n/2$ malicious parties. Then we can solve the Byzantine agreement problem (for the same t).*

Can you see why the claim holds?

3.1 A Broadcast Protocol in $t + 1$ Rounds

We will now describe the protocol, starting from the basic setting.

A signature setup. We assume that each party P_i has corresponding signature keys sk_i, vk_i, and that all parties know all public verification keys. We will in fact, only achieve agreement and validity except with negligible probability over the choice of signature keys (and any coins used by the adversary). This is good enough for any real purpose (you can easily check that the lower bound we proved before even holds if we require that the protocol works with overwhelming probability).

The communication model. We assume that the protocol proceeds in rounds, where at each round all parties send/recieve messages to/from other parties. This is known as *synchronous communication.* Later on in the course we will also consider a more difficult setting of *asynchronous communication* where the adversary has extra control over the timing of message delivery.

The Dolev-Strong protocol. We now move to the protocol, which consists of $t+1$ rounds.

At a very high level, in the protocol, the sender will send everyone its message m. Then in the next rounds all honest parties that receive a message will attempt to propagate it to all other parties. After sufficiently many rounds, we would like to be convinced that if any given honest party got a certain message so did all other honest parties and that if the sender is honest the only message that can be propagated is her message. We will implement the propagation mechanism using signatures.

To formalize this, we will use the notion of signature chains:

Definition 3.2 (valid and new signature chains). *For every $i \leq t+1$, let $c = (m, p_1, \sigma_1, p_2, \sigma_2, \ldots, p_i, \sigma_i)$ be such that $m \in \{0,1\}$ is a message, each $p_j \in [n]$ is a party, and each σ_j is a signature.*

- *We say that c is a* **valid *i*-chain** *if for each $j \in [i]$, σ_j is a valid signature under vk_{p_j} on the prefix chain $(m, p_1, \sigma_1, p_2, \sigma_2, \ldots, p_{j-1}, \sigma_{j-1})$ and the parties $p_1, \ldots, p_i$ are distinct.*
- *For every party $k \in [n]$, we say that c is* **new for *k*** *if $k \notin \{p_1, \ldots, p_i\}$.*

The protocol proceeds as follows:

Round 1: $P_1(m)$ sends the chain $(m, 1, \sigma_1)$ to all parties, where σ_1 is her signature on m.

Round $i+1$ (for $1 \leq i \leq t$): Each party k, having just received (in round i) a **new and valid** i-chain c, extends it to a $i+1$-chain $c' = (c, k, \sigma_k)$, where σ_k is a signature on c. Considering all extended chains c', it now picks at most two, one for each possible value $m \in \{0,1\}$, and sends them to all other parties.

Decision: Any party k that received at any round i a valid i-chain c corresponding to message m and has not received at any round i' a valid i'-chain corresponding to message $1-m$, outputs m. Otherwise, it outputs a default value 0.

Why doesn't the lower bound apply? Before we go on to analyze this protocol it is worth while thinking why this protocol circumvents the lower bound we've seen. We can view the lower bound we've seen as saying that one of the following happens:

- There is an attack A^* on agreement.
- There is an attack C^* on validity.
- There is an attack B^* on validity (this was the symmetric case to C^*, which we didn't spell out).

Is this still true? Can C^* (respectively, B^*) be realized efficiently? (make sure you understand why not).

3.2 Analysis

We now show that the protocol satisfies validity and agreement. Throughout, we shall assume that signatures never get forged; indeed, this occurs only with negligible probability.

Validity. First note that if the sender is honest, then all parties $k \neq 1$ will receive at round 1 a 1-valid chain of the message m. Also, by the security of the signature scheme, they will not get a valid chain for $1-m$, or a signature under vk_1 is forged (make sure you understand why).

Agreement. To show agreement, we will show that if an honest party k obtains a valid i-chain c at round i corresponding to a message m, then all other honest parties receive a valid i'-chain at some round i' corresponding to the same message m. This ensures that all honest parties together will either output m, or output the default output 0.

We will assume w.l.o.g that c is new for k, indeed if it is not the case there exists some $i' < i$ and an i'-chain c', which is a prefix of c and is new for k, or k wouldn't have added her signature.

We consider two cases:

1. $i \leq t$**:** In the next round $i+1$, all parties will receive a valid $i+1$-chain c' from k, which extends c and thus corresponds to the same m.

2. $i = t+1$**:** Let $c = (m, p_1, \sigma_1, p_2, \sigma_2, \ldots, p_{t+1}, \sigma_{t+1})$. Since the parties $p_1, \ldots, p_{t+1}$ are distinct and at most t of them are malicious, the chain c includes at least one honest party p_j. The fact that p_j added her signature implies it received a new valid j-chain c' at round j where chain c extends c', and thus c' also corresponds to m. Accordingly, at round $j+1$, p_j sent a valid $j+1$-chain corresponding to m to all parties.

This completes the analysis.

Better protocols. The above protocol is quite simple, but has round complexity that could be as bad as $\Omega(n)$. Randomized BA protocols that require only a constant number of rounds in expectation do exist. You will see such a protocol in the recitation.

Next: we will go back to the question of consensus in the context of blockchains. In particular, we shall discuss if and how classical consensus protocols as the ones we've seen fit in that context.

References

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.