Blockchains By Example

Tommi Jalkanen - Kodan

Overview

1. Blockchain
2. Node API
3. Demo time

Blockchain

```rust
pub struct Blockchain {
    pub chain: Vec<Block>,
    current_transactions: Vec<Transaction>,
    pub nodes: HashSet<String>,
}
```

Block

```rust
pub struct Block {
    pub index: u64,
    timestamp: DateTime<Utc>,
    pub transactions: Vec<Transaction>,
    pub proof: u64,
    pub previous_hash: String,
}
```

Transaction

```rust
pub struct Transaction {
    sender: String,
    recipient: String,
    amount: i64,
}
```

Genesis block

```
pub fn new() -> Blockchain {
    let mut blockchain = Blockchain {
        chain: vec![],
        current_transactions: vec![],
        nodes: HashSet::new(),
    };
    blockchain.new_block(100, Some("1"));
    blockchain
}
```

Creating a block

```rust
pub fn new_block(&mut self, proof: u64, previous_hash: Option<&str>) -> Block {
    let block = Block {
        index: (self.chain.len() + 1) as u64,
        timestamp: Utc::now(),
        transactions: self.current_transactions.drain(0..).collect(),
        proof,
        previous_hash: previous_hash.unwrap_or("0").to_string(),
    };

    self.chain.push(block.clone());
    block
}
```

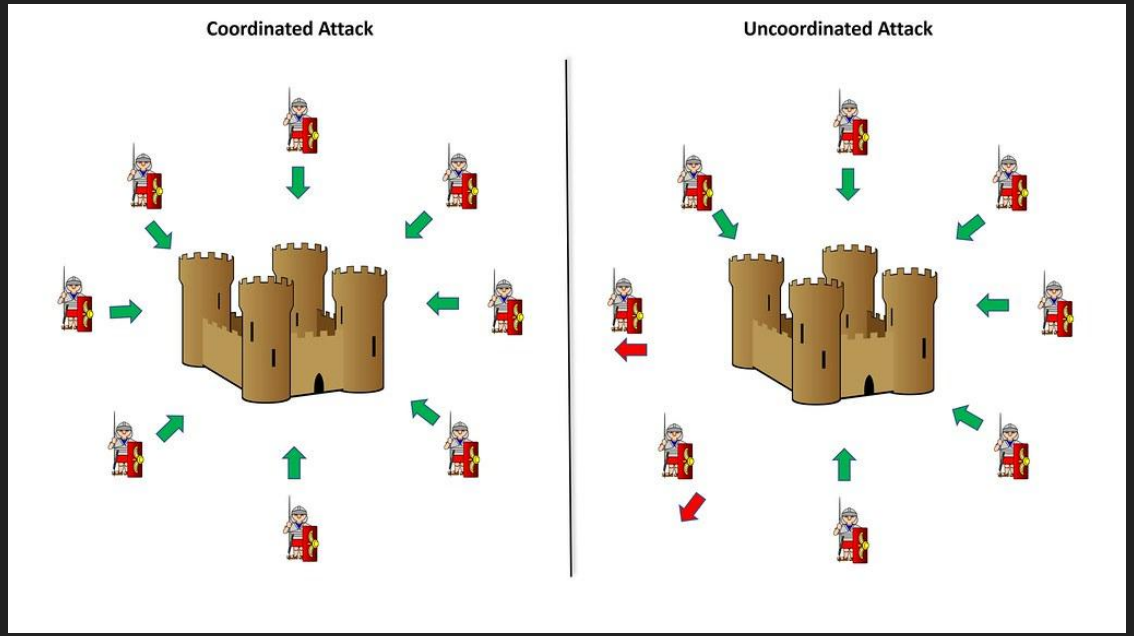
Proof of Work

```
pub fn proof_of_work(last_block: &Block) -> u64 {
        let mut proof = 0;
        let last_proof = last_block.proof;
        let last_hash = &last_block.previous_hash;
        while !Self::valid_proof(last_proof, proof, last_hash) {
            proof += 1;
        }
        proof
    }

fn valid_proof(last_proof: u64, proof: u64, last_hash: &String) -> bool {
        let guess = format!("{}{}{}", last_proof, proof, last_hash);
        let guess_hash = hex_digest(Algorithm::SHA256, guess.as_bytes());
        guess_hash.ends_with("0000")
    }
```

Decentralization?

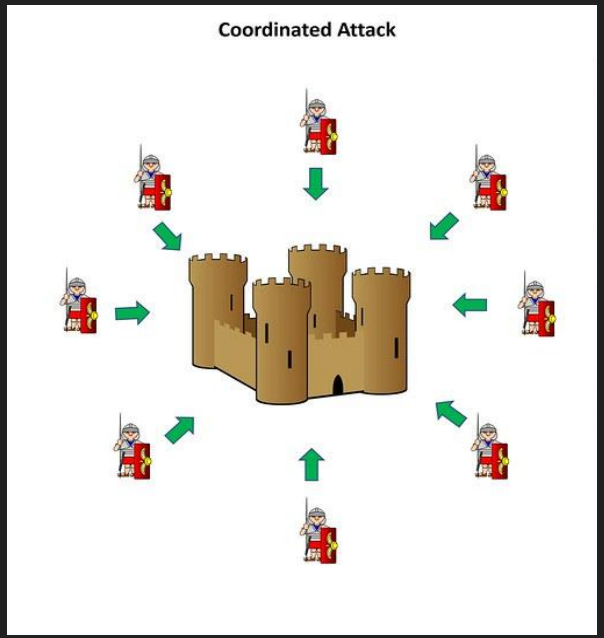
Byzantine Generals Problem



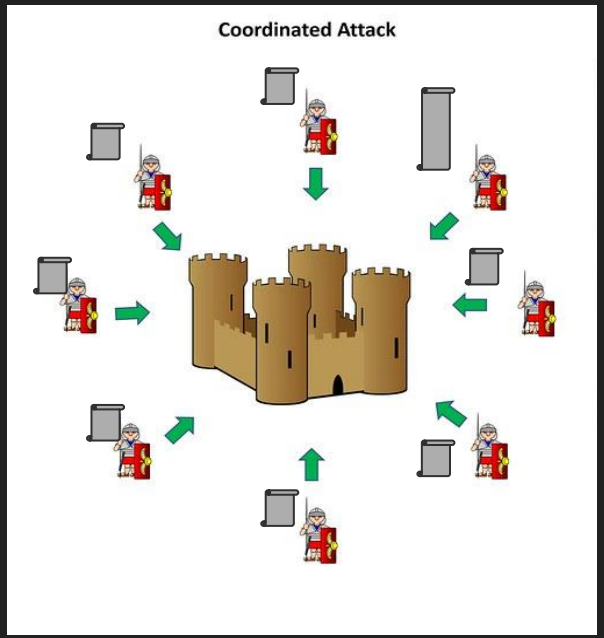
Reaching Consensus

```rust
pub fn resolve_conflicts(&mut self) -> bool {
    let mut max_length = self.chain.len();
    let mut new_chain: Option<Vec<Block>> = None;
    for node in &self.nodes {
        let mut response = reqwest::get(&format!("http://{}/chain", node)).unwrap();
        if response.status().is_success() {
            let node_chain: Chain = response.json().unwrap();
            if node_chain.length > max_length && self.valid_chain(&node_chain.chain) {
                max_length = node_chain.length;
                new_chain = Some(node_chain.chain);
            }
        }
    }
    match new_chain {
        Some(x) => {
            self.chain = x;
            true
        }
        None => false,
    }
}
```

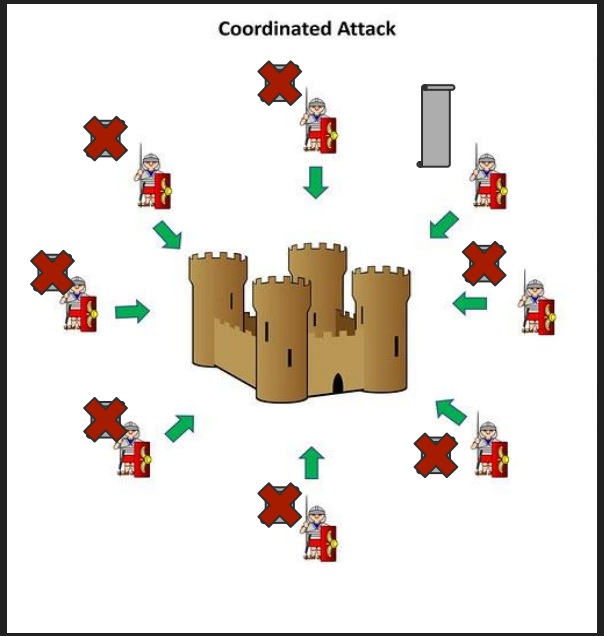
Byzantine Generals Problem



Byzantine Generals Problem



Byzantine Generals Problem



API Overview

```
let shared_chain = web::Data::new(Mutex::new(blockchain::Blockchain::new()));
let node_identifier = web::Data::new(Uuid::new_v4().to_simple().to_string());

HttpServer::new(move || {
    App::new()
        .register_data(shared_chain.clone())
        .register_data(node_identifier.clone())
        .data(web::JsonConfig::default().limit(4096))
        .service(web::resource("/mine").route(web::get().to(api::mine)))
        .service(web::resource("/transactions/new").route(web::post().to(api::new_transaction)))
        .service(web::resource("/chain").route(web::get().to(api::chain)))
        .service(web::resource("/nodes/register").route(web::post().to(api::register_node)))
        .service(web::resource("/nodes/resolve").route(web::get().to(api::resolve_nodes)))
})
.bind(format!("127.0.0.1:{}", port))
.unwrap()
.run();
```

Response/Request Structures

```rust
#[derive(Deserialize)]
pub struct RegisterRequest {
    nodes: Vec<String>,
}

#[derive(Serialize)]
pub struct RegisterResponse {
    message: String,
    total_nodes: Vec<String>,
}
```

Mining Endpoint Example

```
pub fn mine(
    node_identifier: web::Data<String>,
    state: web::Data<Mutex<Blockchain>>,
    _req: HttpRequest,
) -> HttpResponse {
    let (proof, previous_hash) = {
        let blockchain = state.lock().unwrap();
        let last_block = blockchain.last_block().unwrap();
        let proof = Blockchain::proof_of_work(&last_block);
        let previous_hash = Blockchain::hash(last_block);
        (proof, previous_hash)
    };
    let mut blockchain = state.lock().unwrap();
    blockchain.new_transaction("0", &*node_identifier, 1);
    let block = blockchain.new_block(proof, Some(&previous_hash));
    HttpResponse::Ok().json(MiningRespose {
        message: "New Block Forged".to_string(),
        index: block.index,
        transactions: block.transactions,
        proof,
        previous_hash,
    })
}
```

Demo time

https://github.com/Koura/blockchain-example

Questions?

Tommi Jalkanen
github: koura
twitter: @om_nommy