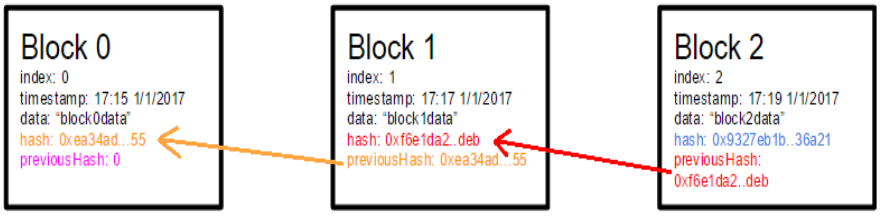
ایده پایه ای بلاک چین بسیار ساده می باشد: پایگاه داده توزیع شده ای که بصورت مداوم لیستی از رکوردهای مرتب و درحال رشد را مدیریت میکند. متاسفانه اکثر افراد هنگام معرفی بلاک چین توضیحات خود را با بیت کوین ترکیب میکنند و مفهوم را کمی پیچیده تر می کنند. واژه بلاک چین ارتباط نزدیکی با تراکنش ها، قراردادهای هوشمند و ارزهای رمزپایه دارد.

این خود باعث شده تا درک مفهوم بلاک چین پیچیده تر بنظر برسد درصورتیکه اینگونه نیست. در این نوشته به معرفی یک بلاک چین ساده که در کمتر از ۲۰۰ خط نوشته شده است می پردازم.

ساختار بلاک

اولین مرحله منطقی تصمیم گیری درباره ساختار و بدنه بلاک می باشد. برای اینکه همه چیز را در ساده ترین حالت خود نگه داریم، تنها موارد ضروری را درون بلاک خود درنظر می گیریم از جمله: index, timestamp, data, hash, previews_hash



(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/p۱.png)

برای حفظ یکپارچگی بلاک ها باید مقدار hash بلاک قبلی درون بلاک فعلی وجود داشته باشد.

```
class Block {
    constructor(index, previousHash, timestamp, data, hash) {
        this.index = index;
        this.previousHash = previousHash.toString();
        this.timestamp = timestamp;
        this.data = data;
        this.hash = hash.toString();
    }
}
```

(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/c۱.png)

هش بلاک

برای اینکه بتوانیم یکپارچگی داده ها را حفظ کنیم بایستی بلاک را هش کنیم. در این نمونه از SHA-۲۵۶ استفاده می کنیم. دقت داشته باشید که این هش هیچ ارتباطی با mining ندارد چراکه در این نمونه کد ProofOfWork وجود ندارد.

```
var calculateHash = (index, previousHash, timestamp, data) => {
    return CryptoJS.SHA256(index + previousHash + timestamp + data).toString();
};
```

(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/c۲.png)

تولید یک بلاک

برای تولید یک بلاک باید هش بلاک قبلی را بدانیم و سپس باقیمانده هش مورد نیاز را ایجاد کنیم(index, hash, data, timestamp=). داده درون بلاک چیزی است که توسط کاربر ارائه می شود.

```
var generateNextBlock = (blockData) => {
    var previousBlock = getLatestBlock();
    var nextIndex = previousBlock.index + 1;
    var nextTimestamp = new Date().getTime() / 1000;
    var nextHash = calculateHash(nextIndex, previousBlock.hash, nextTimestamp, blockData);
    return new Block(nextIndex, previousBlock.hash, nextTimestamp, blockData, nextHash);
};
```

(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/c۳.png)

ذخیره سازی بلاک ها

در اینجا از یک آرایه درون حافظه برای ذخیره سازی بلاک چین استفاده کرده ایم. اولین بلاک از بلاک چین را genesis-block می نامند که در این مثال hard code شده است.

```
var getGenesisBlock = () => {
    return new Block(0, "0", 1465154705, "my genesis block!!", "816534932c2b7154836da6afc367695e6337
};

var blockchain = [getGenesisBlock()];
```

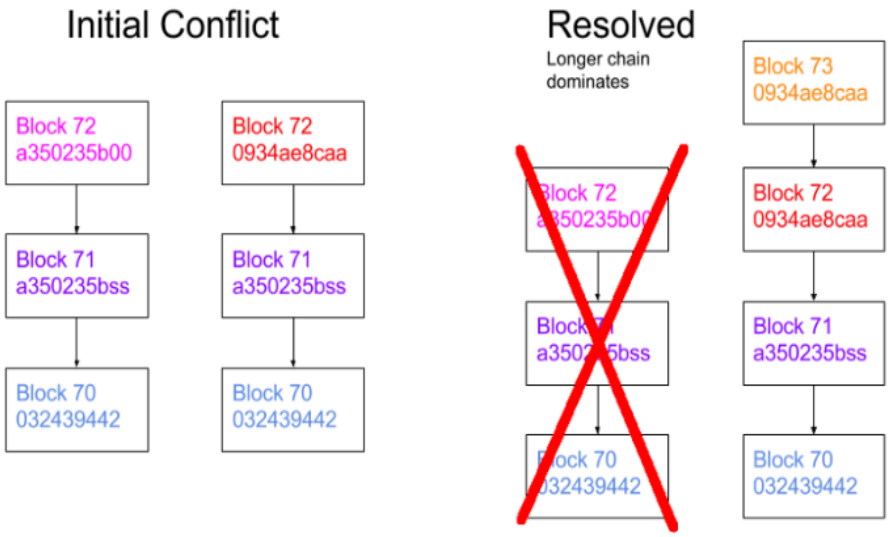
(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/c۴.png)

اعتبارسنجی یکپارچگی بلاک ها

باید در هر زمان بتوانیم یکپارچگی بلاک های درون بلاک چین را اعتبارسنجی کنیم. بخصوص در مواردی که بلاک های جدیدی دریافت میکنیم و باید درباره پذیرفتن یا رد کردن آن تصمیم گیری کنیم.

انتخاب طولانی ترین زنجیره

همیشه فقط و فقط بایستی یک مجموعه از بلاک ها درون زنجیره وجود داشته باشد. این خاصیت اصلی بلاک چین می باشد. در حالتی که تداخلی بوجود امده باشد(بعنوان مثال دو گره بلاک شماره ۷۲ را تولید می کنند)، آن زنجیره ای را انتخاب می کنیم که طولانی ترین تعداد بلاک ها را داشته باشد.



(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/p۲.png)

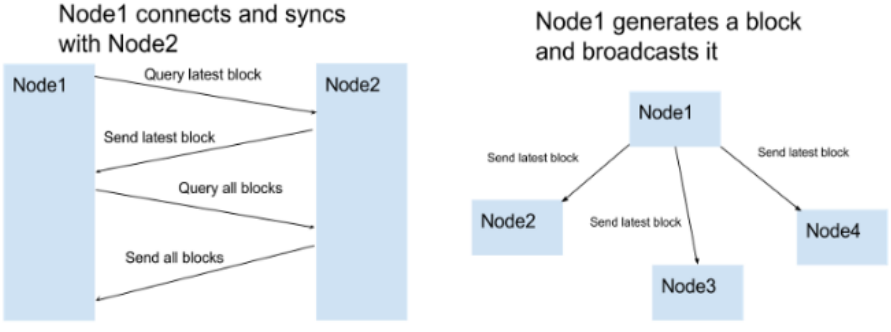
```
var replaceChain = (newBlocks) => {
    if (isValidChain(newBlocks) && newBlocks.length > blockchain.length) {
        console.log('Received blockchain is valid. Replacing current blockchain with received blockchain')
        blockchain = newBlocks;
        broadcast(responseLatestMsg());
    } else {
        console.log('Received blockchain invalid');
    }
};
```

(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/c۶.png)

برقراری ارتباط با دیگر گره ها

یکی از بخش های ضروری هر گره به اشتراک گذاری و همگام سازی بلاک چین خود با دیگر گره ها می باشد. قوانین زیر برای همگام نگه داشتن گره های درون شبکه مورد استفاده قرار می گیرند:

-
-
-



(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/p۳.png)

کنترل گره ها

کاربر بایستی بتواند به نحوی بر روی گره کنترل داشته باشد. برای این کار یک سرور HTTP راه اندازی میکنیم.

```
var initHttpServer = () => {
    var app = express();
    app.use(bodyParser.json());

    app.get('/blocks', (req, res) => res.send(JSON.stringify(blockchain)));
    app.post('/mineBlock', (req, res) => {
        var newBlock = generateNextBlock(req.body.data);
        addBlock(newBlock);
        broadcast(responseLatestMsg());
        console.log('block added: ' + JSON.stringify(newBlock));
        res.send();
    });
    app.get('/peers', (req, res) => {
        res.send(sockets.map(s => s._socket.remoteAddress + ':' + s._socket.remotePort));
    });
    app.post('/addPeer', (req, res) => {
        connectToPeers([req.body.peer]);
        res.send();
    });
    app.listen(http_port, () => console.log('Listening http on port: ' + http_port));
};
```

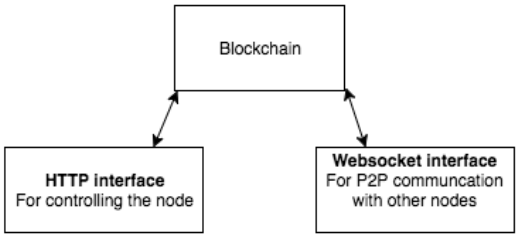
(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/c۷.png)

همانطور که در کد بالا دیدید، کاربر می تواند عملیات های زیر را با گره انجام دهد:

-
-
-

معماری

دقت داشته باشید که این گره دقیقا دو وب سرور را مشخص می کند. یکی برای کاربر که بتواند گره را کنترل کند(سرورHTTP) و دیگری برای مدیریت ارتباطات و تعاملات بین گره ها(Websocket HTTP Server).



(http://aminsajedi.ir/wp-content/uploads/۲۰۱۷/۱۱/p۴.png)

برای کسب اطلاعات بیشتر در زمینه فنی این نمونه کد می تواند از صفحه این کد در گیت هاب (https://github.com/lhartikk/naivechain) بازدید کنید.

« »