Blockchain, Ethereum and Business Applications

@ZimMatthias       Matthias Zimmermann
BSI Business Systems Integration AG
# Crypto Currencies Market

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Symbol</th>
<th>Market Cap</th>
<th>Price</th>
<th>Circ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bitcoin</td>
<td>BTC</td>
<td>$27,712,501,250</td>
<td>$1698.20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Ethereum</td>
<td>ETH</td>
<td>$8,323,039,528</td>
<td>$211.03</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Ripple</td>
<td>XRP</td>
<td>$7,104,942,831</td>
<td>$0.87191</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Litecoin</td>
<td>LTC</td>
<td>$1,461,241,032</td>
<td>$29.06</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NEM</td>
<td>XEM</td>
<td>$999,720,000</td>
<td>$0.111090</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Dash</td>
<td>DASH</td>
<td>$718,371,555</td>
<td>$98.62</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Ethereum Classic</td>
<td>ETC</td>
<td>$603,415,937</td>
<td>$6.60</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Monero</td>
<td>XMR</td>
<td>$431,937,555</td>
<td>$29.94</td>
<td></td>
</tr>
</tbody>
</table>

List by the International Monetary Fund (Partial forecasted estimates for 2017)⁹

<table>
<thead>
<tr>
<th>Rank</th>
<th>Country</th>
<th>GDP (millions of int$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>China</td>
<td>126,387,917</td>
</tr>
<tr>
<td></td>
<td>European Union</td>
<td>20,852,702</td>
</tr>
<tr>
<td>2</td>
<td>United States</td>
<td>19,417,144</td>
</tr>
<tr>
<td>132</td>
<td>Tajikistan</td>
<td>27,802</td>
</tr>
<tr>
<td>133</td>
<td>Mauritius</td>
<td>27,507</td>
</tr>
<tr>
<td>156</td>
<td>Fiji</td>
<td>8,798</td>
</tr>
<tr>
<td>157</td>
<td>Burundi</td>
<td>8,024</td>
</tr>
<tr>
<td>158</td>
<td>Suriname</td>
<td>7,961</td>
</tr>
<tr>
<td>159</td>
<td>Lesotho</td>
<td>7,287</td>
</tr>
<tr>
<td>160</td>
<td>Bhutan</td>
<td>7,045</td>
</tr>
</tbody>
</table>
Bitcoin
First decentralized digital currency

- By «Satoshi Nakamoto»
- White Paper 2008
- Open Source Software 2009
Gratulation, du hast deine erste Zahlung erhalten! Hast du deine Wallet bereits gesichert, um dich gegen Verlust zu schützen?

14. November 12:25
1Mk6 LVaT nMf
9Ytu 316N qKQE
SM44 C3Ja1e

+ 26.45

CHF 18.55
Coffee at Bob’s

Address: 1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA
Amount: 0.015
Name Recipient: Bob’s Café

Source: «Mastering Bitcoin», Andreas M. Antonopoulos
What is a Bitcoin Address?

Addresses corresponds to «Accounts»

Encoded Numbers
- Example: `1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA`
- Derived from a **private Key**
- Private key is 256-Bit random number

Getting, using and loosing Accounts
- Create your account with «Coin, Paper and Pencil»
- No ID required, no showing up at local branch, ...
- To send Bitcoins you need your private Key
- **You loose your private key ➔ you loose your money**
The Coffee Transaction

- Bitcoin network confirms coffee TX after ~ 10 min.
- TX Elements

![Screenshot: blockchain.info]
TX Blocks and the Blockchain

- **Block #277316** includes coffee TX
- **Block Elements**

**Block Hash** – do you see the leading zeros?

419 TX (€495,000)

Link to previous Block

---

Screenshot: blockchain.info
From Transactions to Blocks (Mining)
From Transactions to Blocks (Mining)

1. New TX are propagatet through Bitcoin **peer-to-peer network**
2. Bitcoin client verifies new TX and adds it to local «**mempool»**
3. Client starts to «**mine**» transactions:
   - Assemble TX from mempool to **block candidate**
   - Starts to solve the block candidate’s **crypto challenge**
   - Computes **MANY** hashes to solve the crypto challenge
   - Client solving the challenge first, gets **block reward** and all **TX fees**
4. Winning client sends the new block to its peers
5. Arrival of new block triggers the next challenge
Bitcoin Mining Today

- **Mining-pools**: Include many ASIC computers (PC way too slow)
- **AntMiner**: 10,000x faster than PC, burns 10x more electricity
- **Energy Costs**: # of hashes per KWh is central criteria + cooling(!)

**Hashing Power over the Years**

![Graph showing the increase in hashing power over the years, with a peak value of $1.5 \times 10^{18}$]

*IGraph: blockchain.info*
Distributed Consensus Mechanism

Preventing Forks

The Challenge

- Mining clients build block candidates independently
- Several new blocks might be found at the «same» time
- Clients may receive new blocks that are inconsistent
- The local copy of the blockchain may have forks

The Solution

- The «true» blockchain is defined by the highest cumulative PoW (difficulty)
- By selecting the greatest-difficulty chain, eventual consensus is achieved
- Miner majority vote defines the true chain
- Miners «vote» for the true chain by deciding which block/fork to extend
«Unhappy with some TX?»
The 51% Attack

Attacking Bitcoin (any PoW Blockchain)
1. Install more mining capacity than the rest of the world (>= 51%)
2. Censor/suppress unwanted TX
3. Mine a secret branch containing acceptable TX
4. Continue to mine until PoW of secret branch exceeds official branch
5. Broadcast secret branch to Bitcoin network
6. All Bitcoin clients will switch to this new branch

Ok, this is kind of hard – but:

- Miners earn ~1.2bn/year
- HW cost to match Bitcoin mining capacity: ~ $400m (Antminer 9s)
Bitcoin Challenges

Current Issues

- Increasing TX backlogs
- TX confirmation can take hours (instead of 10’)
- Increasing TX fees
- «War» between Bitcoin Core and Bitcoin Unlimited (Scaling Debate)
- Decreasing market share
Bitcoin Market Share
Bitcoin Price

![Bitcoin Price Chart](chart.png)

The latest Bitcoin Price Index is $2,168.73 USD.}

- **Open**: $2,055.62
- **High**: $2,185.89
- **Low**: $2,045.79
- **Market Cap**: $16,344,725
- **Volume**: $2100

Source: CoinDesk
Bitcoin Recap

Bitcoin Success

- Completely decentral currency (no need for central banks)
- Open Source (GitHub) and Open Data (complete TX history)
- First successful implementation of any crypto currency
- «Gold Standard» since 2009
- Record price levels

Bitcoin Challenges

- Declining market share
- Scaling debate/war
Bitcoin Resources
Ethereum
Decentralized Smart Contracts

• 2014 by Vitalik Buterin
• Distributed Turing complete VM
• Open source software 2014
• Is crypto currency too
Ethereum vs Bitcoin

Common Traits
- Local clients/nodes with complete blockchain (open data)
- Concepts of addresses, transactions, mining
- Virtual currency Ether
- Open source

Main Differences
- Specification with different implementations (Bitcoin: single client)
- Turing complete scripting (Bitcoin: very limited scripting)
Ethereum as Virtual Currency Platform

- **Ether** is currency unit (1 Ether ~ 87$ 17.05.17)
- **Wei** is smallest denomination (10^{-18} Ether)
- TX mining: Proof of Work (PoW)
- Distributed consensus like Bitcoin (true chain == highest cumulative PoW)

**Storing Information**
- **Ethereum clients**: Maintain blockchain data + state data
- **State data**: Account balances + nonces
- **Transaction data**: Ether transfers
Ethereum and App Integration

Ethereum Client
Geth/TestRPC/...

Ethereum Peer-to-Peer Network

JavaScript
web3

Java
web3j

Interface
JSON-RPC

http://localhost:8545
Ethereum Hands-on

Ideal Development Setup

- Offline, repeatable, fast
- **Java** (this is a JUG talk after all)

We can have all this 😊

- **Docker** (repeatable, shareable)
- **TestRPC** (offline + local blockchain /w immediate TX confirmations)
- **Web3j** (Ethereum Java Library)
TestRPC Docker Image

TestRPC Client

The TestRPC projects provides the fastest Ethereum client for testing and development of Ethereum applications. This makes the TestRPC client ideal for development and automated tests.

Start with building the TestRPC Docker image from this repository’s main directory.

```
~/Web3j Demo
docker build -t ethereum_testrpc docker_testrpc
```

The above command will create a Docker image named `ethereum_testrpc`. Expect this process to take a couple of minutes.

The resulting Docker image can then be used to start a TestRPC client.

```
~/Web3j Demo
docker run -p 8545:8545 -d ethereum_testrpc
```

The TestRPC container comes with 10 accounts that have an initial balance of 100 Ethers. All accounts are unlocked and can be used for Ether transfers.
Running TestRPC container

Open shell in container

Start node.js

```bash
docker ps
```

```bash
docker exec -it 8edec9b18489 bash
```

```javascript
root@8edec9b18489:/H cd /usr/lib/node_modules/ethereumjs-testrpc
root@8edec9b18489:/usr/lib/node_modules/ethereumjs-testrpc# node
```

```javascript
var Web3 = require('web3')
undefined
var web3 = new Web3()
undefined
undefined
web3.eth.mining
true
```

```javascript
web3.eth.coinbase
'0x1a0504f6e47b92b8f375fd552bea2e4a1de47f'
```

```javascript
web3.fromWei(web3.eth.getBalance(web3.eth.coinbase), "ether")
{ [String: '94.053061993622566916'] s: 1, e: 1, c: [ 94, 5306199362256, 6916000000000 ] }
```
TestRPC initial accounts

Get initial balance

Send 123456789 Weis

New balance
web3j: Ethereum and Java

- Java implementation of JSON-RPC client API
- A couple of other features
- Android support

```xml
<dependencies>
  <dependency>
    <groupId>org.web3j</groupId>
    <artifactId>core</artifactId>
    <version>2.1.0</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```
public void run() throws Exception {
    // show client details
    Web3ClientVersion client = web3j.web3ClientVersion().sendAsync().get();
    System.out.println("Connected to " + client.getWeb3ClientVersion() + "\n");

    // get basic info
    EthMining mining = web3j.ethMining().sendAsync().get();
    EthCoinbase coinbase = web3j.ethCoinbase().sendAsync().get();
    EthAccounts accounts = web3j.ethAccounts().sendAsync().get();
    System.out.println("Client is mining: " + mining.getResult());
    System.out.println("Coinbase address: " + coinbase.getAddress());
    System.out.println("Coinbase balance: " + Web3jUtils.getBalanceEther(web3j, coinbase.getAddress()) + "\n");

    // get addresses and amount to transfer
    String fromAddress = coinbase.getAddress();
    String toAddress = accounts.getResult().get();
    BigInteger amountWei = Convert.toWei("0.123", Convert.Unit.ETHER).toBigInteger();

    // do the transfer
    demoTransfer(fromAddress, toAddress, amountWei);  
}
web3j: Creating and Sending TX

```java
// step 1: get the nonce (tx count for sending address)
EthGetTransactionCount transactionCount = web3j
    .ethGetTransactionCount(fromAddress, DefaultBlockParameterName.LATEST)
    .sendAsync()
    .get();

BigInteger nonce = transactionCount.getTransactionCount();
System.out.println("Nonce for sending address (coinbase): " + nonce);

// step 2: create the transaction object
Transaction transaction = Transaction
    .createEtherTransaction(
        fromAddress,
        nonce,
        Web3jConstants.GAS_PRICE,
        Web3jConstants.GAS_LIMIT,
        amountWei);

// step 3: send the tx to the network
EthSendTransaction response = web3j
    .ethSendTransaction(transaction)
    .sendAsync()
    .get();

String txHash = response.getTransactionHash();
System.out.println("Tx hash: " + txHash);
```
Ethereum Accounts and Nonces

How nonces are used

- Each account has a **nonce** value (account state data)
- Accounts start with nonce value 0
- TX: includes **sender address** and its **nonce** value
- TX can only be mined if:
  - Account has sufficient funds
  - TX nonce == current account nonce
- If TX is mined successfully: Nonce increased by 1
**Ethereum, Gas and TX Fees**

**What is gas?**
- Special unit to pay fees to mining nodes
- Gas has price in Ethers (decouples computing costs and Ether price)

**What fees?**
- Computations performed by **Ethereum Virtual Machine** (EVM)
- EVM is working as long as there is gas
- **Example 1:** SHA3 computation costs 30 gas
- **Example 2:** EVM always terminates (stays in **infinite loop** until gas runs out)
Smart Contracts
What is a Smart Contract?

- Piece of (byte) code
- Is executed by the Ethereum Virtual Machine (EVM)
- Has an owner
- Has a life cycle
- Might have some purpose

Examples

1. The DAO
2. Flight delay insurance
3. «Truly» autonomous cars
The DAO (2016)

- Distributed venture capital fund
- Amount raised $150,000,000
- Largest crowdfunding project
- Successfully «attacked»

Attack Result: Ethereum hard fork
- ETH «true» blockchain
- ETC «forked» blockchain
Flight Delay App

- Involves Oraclize service to access flightstats.com
«Truly» Autonomous Cars

- Smart contract to ordering vehicle to transport goods/people
- Smart contract to pay for energy/services
Smart Contracts Life Cycle

Deploying and using Smart Contracts
1. Write contract in high level language (eg. Solidity)
2. Compile contract to EVM byte-code
3. Pack byte code into a contract creation TX and sent to the network
4. The TX gets ist own contract account
5. Contract account has address, balance, nonce and holds byte code
6. Invoke methods using calls (free) or transactions (cost gas)
contract greeter {

    /* Owner of this contract */
    address owner;

    /* Configurable greeting */
    string greeting;

    /* Constructor runs when contract is deployed */
    function greeter(string _greeting) public {
        owner = msg.sender;
        greeting = _greeting;
    }

    /* Main function */
    function greet() constant returns (string) {
        return greeting;
    }

    /* Function to recover the funds on the contract */
    function kill() {
        if (msg.sender == owner)
            selfdestruct(owner);
    }
}
contract greeter {

  /* Counter for deposits calls */
  uint public deposits;

  /* Default function.
   * 'payable': Allows to move funds to contract.
   * Changes state: Costs gas and needs contract transaction.
   */
  function() payable {
    deposits += 1;
  }

  /* Returns number of deposits.
   * 'constant': This function does not change contract state.
   * Does not change state and does not cost gas/fees.
   * No contract transaction needed.
   */
  function deposits() constant returns (uint) {
    return deposits
  }

  address owner;
  string greeting;

  function greeter(string _greeting) public {
    deposits = 0; ...
  }
  function greet() constant returns (string) {
    ... }
  function kill() {
    ... }
}
Solidity Compiler (online)

byte code (EVM) to deploy contract

Deploy script (JS)
Deploy (Console)

```
{ [Function: bound ]
  request: [Function: bound ],
  sendData: [Function: bound ],
  getData: [Function: bound ],
  allEvents: [Function: bound ]
} Contract mined! address: 0x3647d1140d2596a46eb164b65bd44c4372218b77 transactionHash: 0xb84108d0
> undefined_greeter.greet();
'hello world'
> undefined_greeter.deposits();
{ [String: '0'] s: 1, e: 0, c: [ 0 ] }
```
From Solidity to Java Contract Class

1. Compile `greeter.sol` (e.g. using online compiler)
   \(\Rightarrow\) `greeter.bin, greeter.abi`

2. Create contract wrapper class (use Web3j command line tool)
   \(\Rightarrow\) `Greeter.java`

1. Use `Greeter.java` in your Java code
Generated Contract Wrapper

```java
/**
 * (p)Auto generated code.<br>
 * <strong>Do not modify!</strong><br>
 * Please use @link org.web3j.codegen.SolidityFunctionWrapperGenerator to update.<br>
 * @p>Generated with web3j version 2.1.0.<br>
 */

public final class Greeter extends Contract {
    private static final String BINARY = "0x666666040523461000576040516102a33832806102a3833901016040528051015b6000805600160a060020a0319166c

    private Greeter(String contractAddress, Web3j web3j, Credentials credentials, BigInteger gasPrice, BigInteger gasLimit) {
        super(contractAddress, web3j, credentials, gasPrice, gasLimit);
    }

    private Greeter(String contractAddress, Web3j web3j, TransactionManager transactionManager, BigInteger gasPrice, BigInteger gasLimit) {
        super(contractAddress, web3j, transactionManager, gasPrice, gasLimit);
    }

    public Future<Uint256> deposits() {
        Function function = new Function("deposits",
            Arrays.<Type>asList(),
            Arrays.<TypeReference<?>>asList(new TypeReference<Uint256>() {}))
            return executeCallSingleValueReturnAsync(function);
    }
```
@Override
public void run() throws Exception {
    super.run();

    fundAlice();  // make alice rich
    Greeter greeter = deployContract(); // deploy the greeter contract
    sendFunds(greeter);  // send money to contract
    callGreet(greeter);  // call greet()
    killContract(greeter);  // kill contract
}
private Greeter deployContract() throws Exception {
    System.out.println("// Deploy contract Greeter");

    Greeter contract = Greeter
        .deploy(
            web3j,
            Alice.CREDENTIALS,
            Web3jConstants.GAS_PRICE,
            Web3jConstants.GAS_LIMIT_GREETER_TX,
            BigInteger.ZERO,
            new Utf8String("hello world"))
        .get();

    // get tx receipt
    TransactionReceipt txReceipt = contract
        .getTransactionReceipt()
        .get();
```java
private void callGreet(Greeter contract) throws Exception {
    Utf8String message = contract.greet();
    System.out.println("Message returned by Contract.greet(): " + message.toString());
    System.out.println();
}
```

```
// Deploy contract Greeter
Deploy hash: 8x4Aff7be2b7d4b666faab2269ff200e6556b0a5a52c309494bb3f4914a3f024
Deploy fees: 0.0047888
Contract address: 8xb455a5b8203d5b75b86010f78ca658b87f12534eb86
Alice's account balance (after deploy): 0.02021112
// Send 0.05 Ethers to contract
Contract address balance (after funding): 0.05
Contract.deposits(): 1
// Call greet()
Message returned by Contract.greet(): hello world
Alice's account balance (after greet): 0.02021112
// Kill contract
Contract.kill() fee: 0.00021572
Alice's account balance (after kill): 0.0699954
```
Trading-Network Demo
Ethereum, web3j, Eclipse Scout
Trading Network Demo

Use Case

➤ **Currency Hedging**: Buy orders and Sell orders (€ / US$)

➤ **Classical Business App**
  - **Identity Management** for mapping real persons ↔ BC addresses
  - **User Interface**

➤ **Blockchain Benefits**
  - **Efficiency**: No central organization/infrastructure
  - **Trust**: Tampering-proof ledger, trust by blockchain
Eclipse Scout
Business Application Framework

- Plain Java Application Model
- Multi-Device Support
- HTML5/CSS3 Rendering

```java
public class FirstNameField extends AbstractStringField {
    @Override
    protected String getConfigurerLabel() {
        return TEXTS.get("FirstName");
    }
}
```
Eclipse Scout
UI (web application)

Eclipse Scout
Backend

web3j
JDBC

web3

Ethereum Client
TestRPC

PostgreSQL
Welcome to your FX-Trading Tool

Country: Switzerland

Balance

<table>
<thead>
<tr>
<th>Currency</th>
<th>Balance</th>
<th>Refresh</th>
</tr>
</thead>
</table>

Overview

<table>
<thead>
<tr>
<th>Information</th>
<th>Count</th>
</tr>
</thead>
</table>

USD / EUR - Buy - Inactive 2

USD / EUR - Sell - Inactive 2
Resources
What next?
Blockchain Summary

Gist
- Cool new technology, including much hype 😊
- Internet of decentralized trust

Blockchain Technology is great for
- Efficient value exchange for untrusted environments
- Pushes distributed business models
- Option for the «unbanked»

Current Challenges
- Privacy
- Scalability
- Maturity (blockchain still in its infancy)
Next Steps

Socalizing

➔ Go to talks
➔ Join meetups (Bitcoin Meetup Switzerland, Blockchain Meetup Switzerland, Crypto Valley Forum, …)

Increase Context

➔ Youtube, Blogs, Twitter, ...

Doing

➔ GitHub (web3j/web3j, matthiaszimmermann/web3j_demo, …)
Everybody can do this 😊

«Recipe»
1. Curious about new technologies? Take yourself seriously!
2. Invest some of your time
3. Take advantage of your education options (time, money, ...)
4. Building small teams makes it even more fun
5. Create value for your employer (internal, external)
6. Do it!
Thanks!

@ZimMatthias