BLOCKCHAINS

ARCHITECTURE, DESIGN AND USE CASES

SANDIP CHAKRABORTY
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

PRAVEEN JAYACHANDRAN
IBM RESEARCH,
INDIA

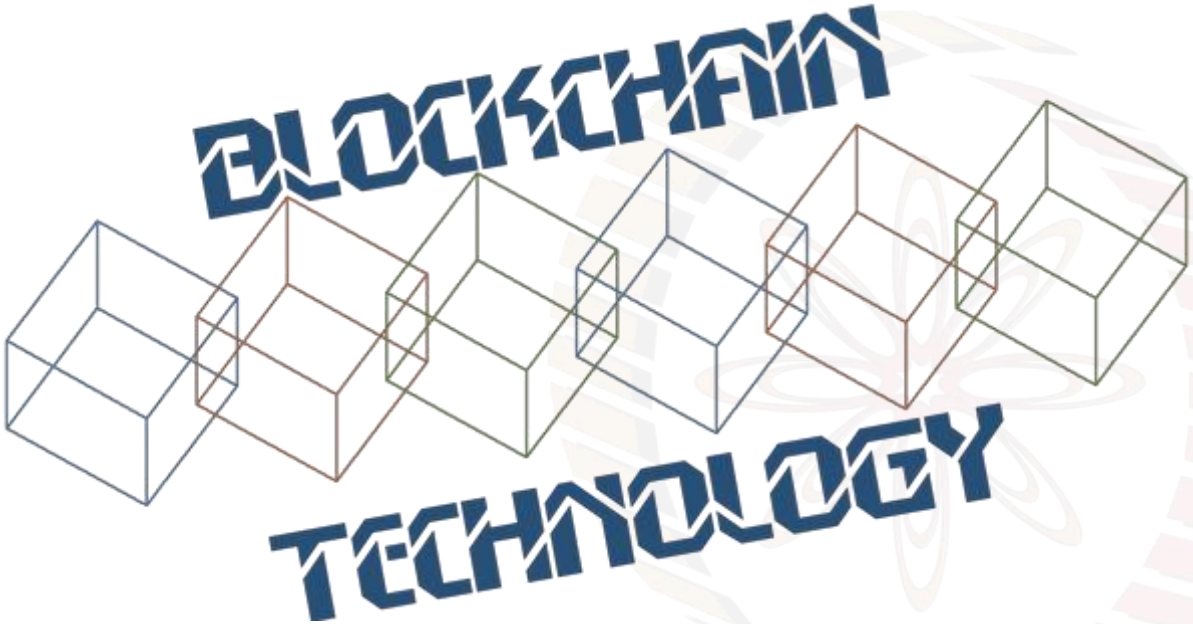
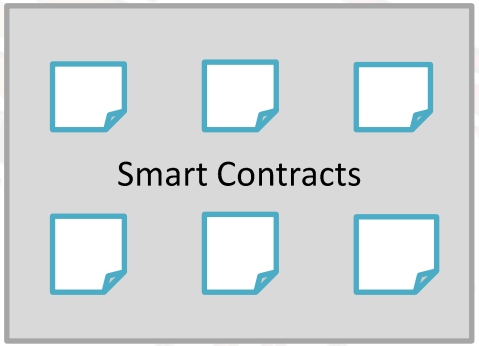


*Image courtesy: http://beetfusion.com/

ETHEREUM

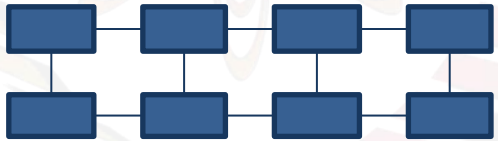
Introduction to Ethereum

Founded by Vitalik Buterin (bitcoin scripting language too restrictive for smart contracts)



The smart contracts are also know as Dapps (Decentralized Applications), and runs on the Ethereum Virtual Machine (EVM)

Ethereum is an overloaded term. May refer to public network, a private/permissioned network, or open source software

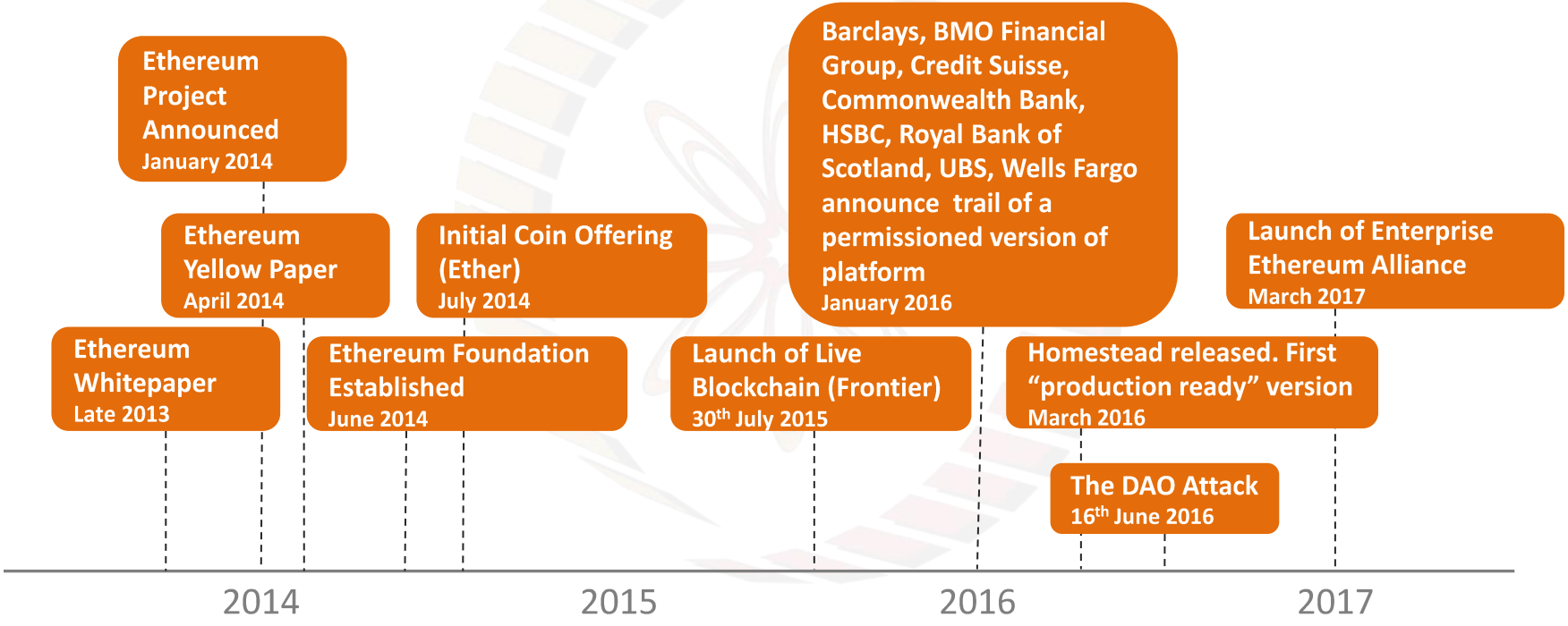


Public mainnet runs more than 16,000 nodes, average block time is 17s (as of May 2018); Miners currently get 5 ETH + transaction fees

Developed by the *Ethereum Foundation*, a Swiss nonprofit organization with many contributions coming from the broader community.

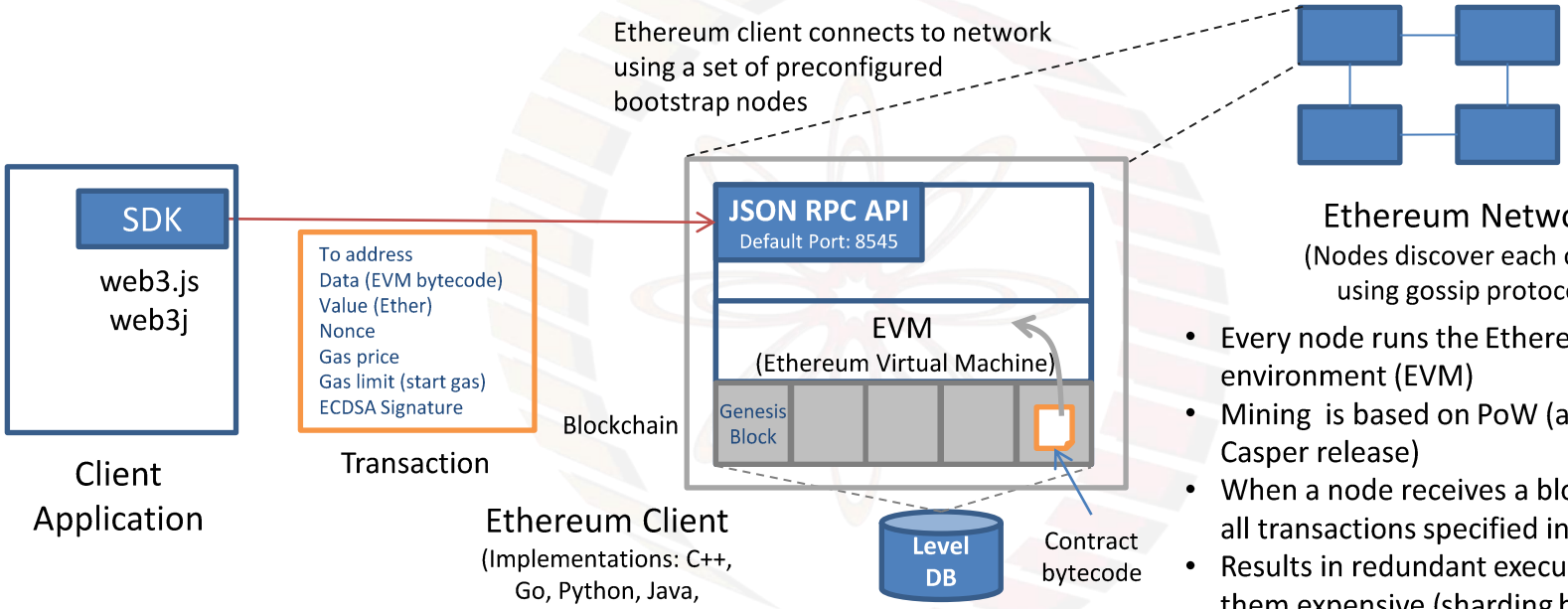
Refs: https://www.ethernodes.org, https://etherscan.io/

A Brief History



Ref: http://ethdocs.org/en/latest/introduction/history-of-ethereum.html

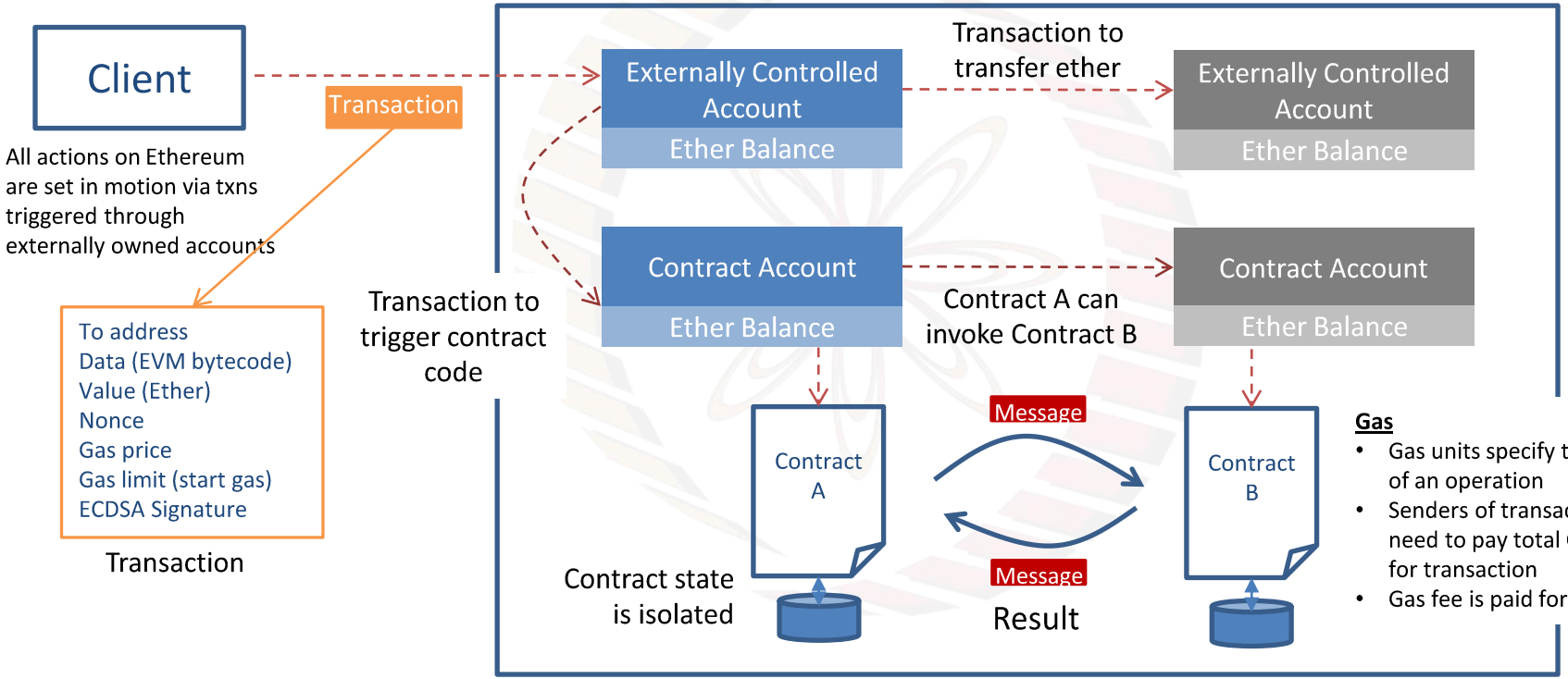
Ethereum Architecture



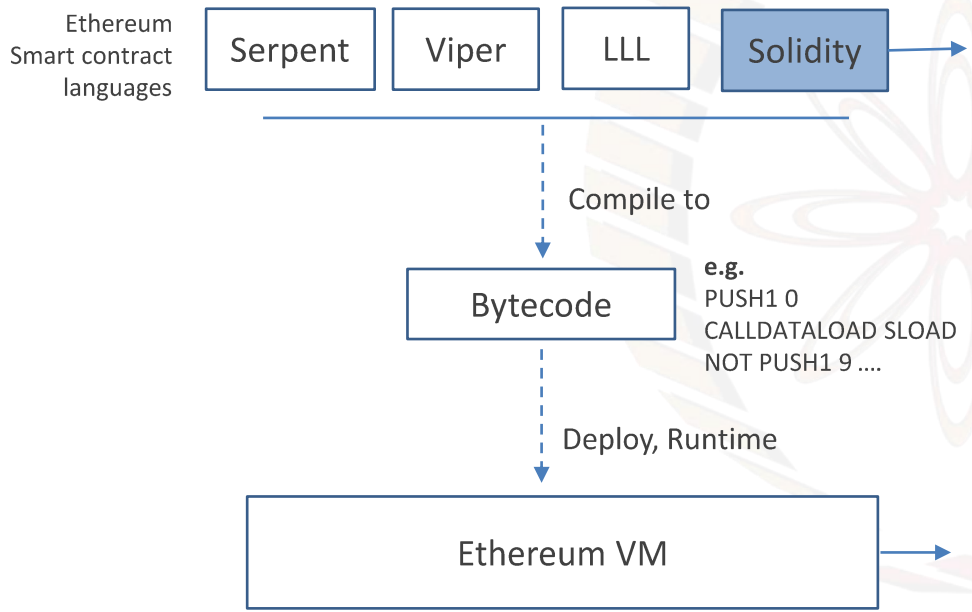
- Every node runs the Ethereum execution environment (EVM)
- Mining is based on PoW (and PoS with Casper release)
- When a node receives a block, the EVM runs all transactions specified in the block
- Results in redundant executions makes them expensive (sharding helps reduce this)

- Stores chain-data (list of blocks) and state data (results of state transitions)
- Default data directory ~/.ethereum

Account Types, Gas and Transactions



Ethereum Smart Contracts



Solidity

- Solidity is a object oriented high-level language for writing EVM based Smart Contracts
- Syntax is similar to that of JavaScript – designed based on ECMAScript syntax
- Statically typed with support for multiple inheritance
- Support for complex user-defined types through structs
- Core language capabilities reflect limitations of EVM. E.g. basic string manipulation is not supported in language, but rather through libraries
- Browser-based IDE with integrated compiler and Solidity runtime environment without server-side components.
- **Code is law!** (at least in belief of design of Ethereum, no authority, no governance)

Ethereum Virtual Machine

- (quasi) Turing complete 256bit Virtual Machine
  - Computation is intrinsically bounded through a parameter, gas
- Allows execution of EVM Bytecode
- JIT interpreter that compiles the byte code into manageable data types and structures.
- Define ~70 OPCODEs
- Gas defined for each OPCODE (e.g., IF, AND, MULTIPLY, SEND), paid for using ether
- Designed for simple operations: “Roughly, a good heuristic to use is that you will not be able to do anything on the EVM that you cannot do on a smartphone from 1999. ”

Example Solidity Code

```
pragma solidity ^0.4.4;
import "KVStore.sol";

contract CustomerManager {
  KVStore private kvStore;

  struct Customer {
    byte32 customerId;
    byte32 name;
  }
  mapping(byte32 => Customer) customers;

  modifier customerNotExists(bytes32 customerId) {
    if (hasCustomerId(customerId)) throw;
    _;
  }

  event NewCustomer(bytes32 indexed customerId, bytes32 name);

  function createNewCustomer(bytes32 customerId, bytes32 name)
           customerNotExists(customerId) {
    //…
    customers[customerId] = Customer(customerId, name)
    NewCustomer(customerId, name);
  }

  function getCustomer(bytes32 customerId) constant
                returns(bytes32 name, bytes32 customerId){}
}
```

Reference to another smart contract

Define Transaction Event

Fire Transaction Events

Modifier functions: ensure execution proceeds only if defined pre-conditions are met

Constant keyword signifies read operations

Indicate return types (and values)

Smart Contract Patterns

- Non trivial DApps (Distributed Apps) often have data and logic over multiple contracts
- Generally five contract model types
  - Database Contract
  - Controller contract: Operate on database contracts, batch reads, work with multiple database contracts
  - Contract Managing Contract (CMC)
  - Application Logic Contract (ALC)
  - Utility Contract
- Dapps typically also have a GUI, allowing users to interact with contracts similar to Web 2.0 apps (HTML/JS/CSS)

Additional Capabilities

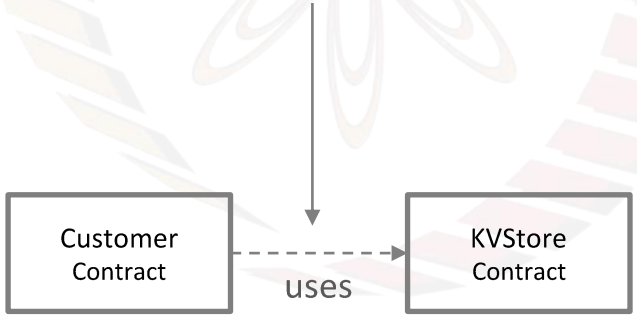
- Proof of stake to save energy consumption, Casper release
- Swarm file storage
  - Decentralized file storage built into Ethereum
  - Uses contracts and ether to encourage cooperation among nodes
- Whisper messaging for secure private communications
  - Messages are encrypted and announced to the network off-chain; person-to-person and contract-to-contract
  - Inherently supports unicast, multicast and broadcast
  - Ability to use masks/filters to narrow down topics of interest without giving away what topic is being sought
  - High latency, low bandwidth
- Raiden state channels – similar to bitcoin lightning network / payment channels; off-chain transactions with proof / dispute resolution on-chain
- Sharding transaction groups – Nodes are partitioned so that nodes don't have to store and process entire blockchain (still in development)

Limitations

- Smart contract state is bound to specific contract. Storing state along with business logic reduces maintainability, since every contract ‘update’ requires state migration from old contract.
- Separation of state and application logic is used as a pattern to mitigate this
- Does not have native string manipulation (e.g. concatenation, find, split ...)
- Cannot return complex data types from functions, only primitives
- Functions cannot contain more than 16 parameters and returns values. Due to the lack of string handling and lack of complex type returns. we have to return multiple individual fields as results to queries. The limit of 16 returns means this model cannot scale.
- Debugging DApps is harder, since the EVM (Ethereum Virtual Machine) does not currently return meaningful error messages.

Inter-Contract Calls

- Cannot pass dynamically sized data (e.g. variable length strings, arrays) to functions called inside smart contracts.
- Calling one contract from another creates a new EVM instance for the called contract. The performance cost of this can quickly add up especially when storing complex types in a generic database contract



Fun Reading

- Ethereum documentation: http://www.ethdocs.org/en/latest/
- White paper: https://github.com/ethereum/wiki/wiki/White-Paper
- Yellow paper: https://ethereum.github.io/yellowpaper/paper.pdf
- Raiden state channels: https://raiden.network/


thank you!