區塊鏈技術與應用

廖峻鋒

cfliao@nccu.edu.tw

國立政治大學 資訊科學系

國立政治大學 金融科技研究中心

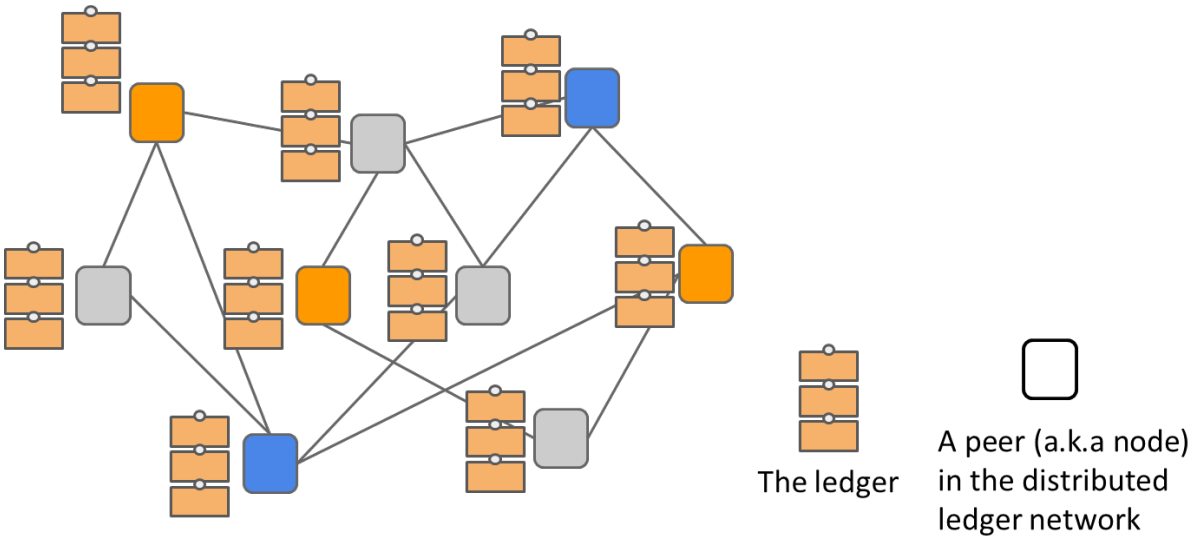
大綱

- 簡介
- Blockchain技術
- Blockchain應用
  - Blockchain與智能合約開發
  - Blockchain與物聯網整合
- 未來展望

Note: to make it concrete, in this talk, we use terms and designs of Ethereum when discussing the blockchain concepts

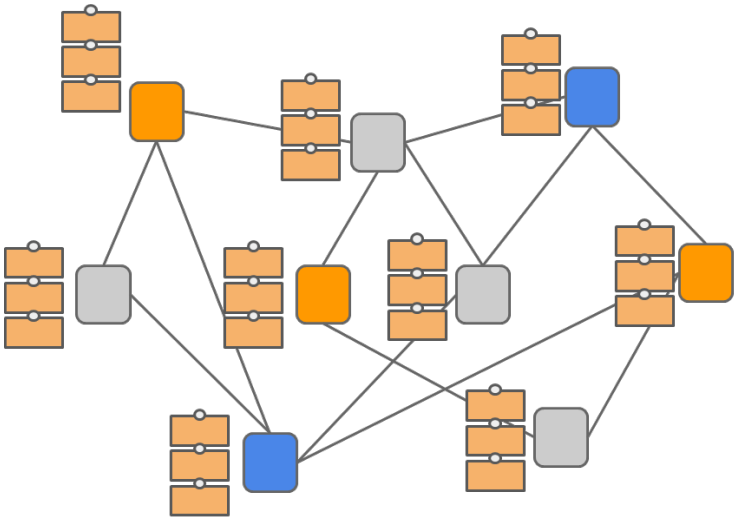
Introduction

- A blockchain system
  - Consists of a network of interconnected peer nodes
  - Each node maintains a replicated and identical copy of ledger
  - The “canonical” ledger is derived using “consensus” mechanism



Introduction

- A blockchain system = a p2p distributed ledger
  - Cryptographically secure 製造假交易非常困難
  - Append-only 可追朔
  - Writable only via consensus among peers 達成共識後修改才永久確認



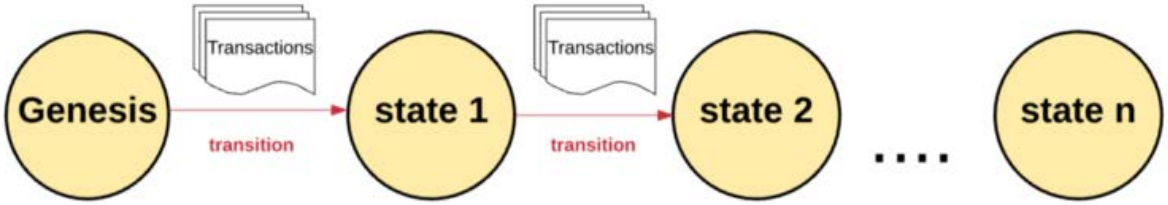
The ledger is replicated and stored in each node

The ledger

A peer (a.k.a node) in the distributed ledger network

Transaction and State

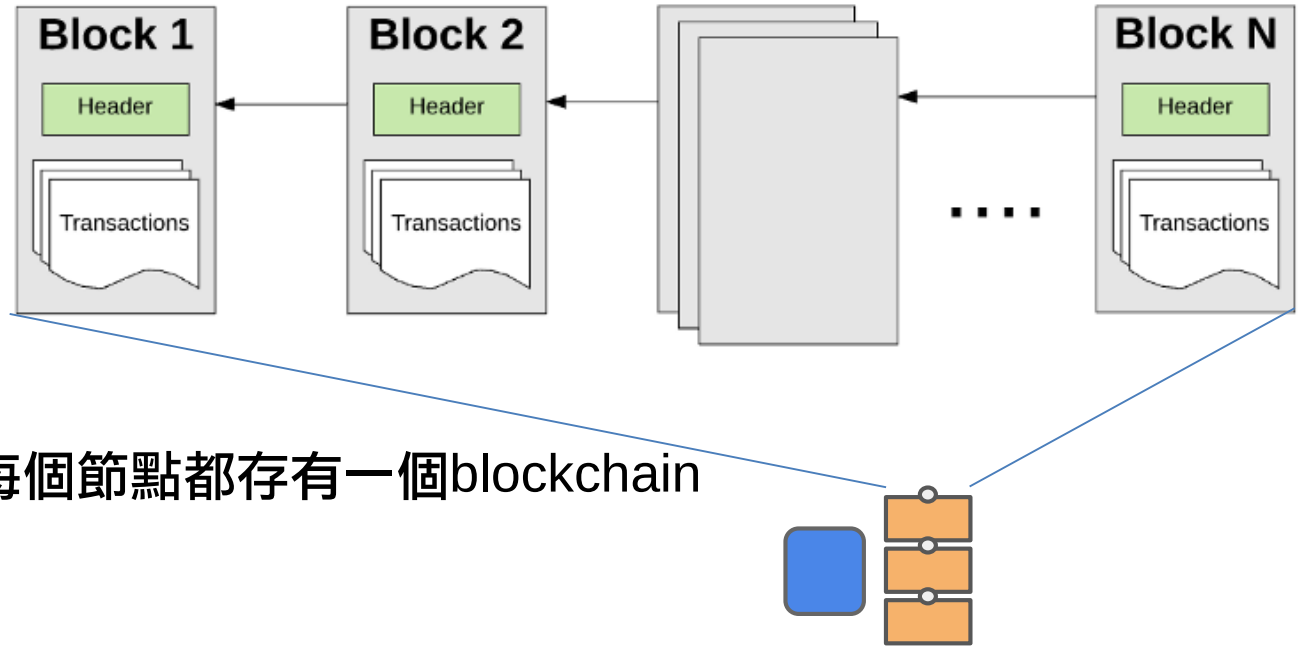
- The blockchain is a transactions-based state machine



- State
  - 任何儲存在distributed ledger中的資訊
- Transaction
  - 修改state的”企圖”稱為Transaction
  - Transaction被所有節點接受後，修改才會被接受

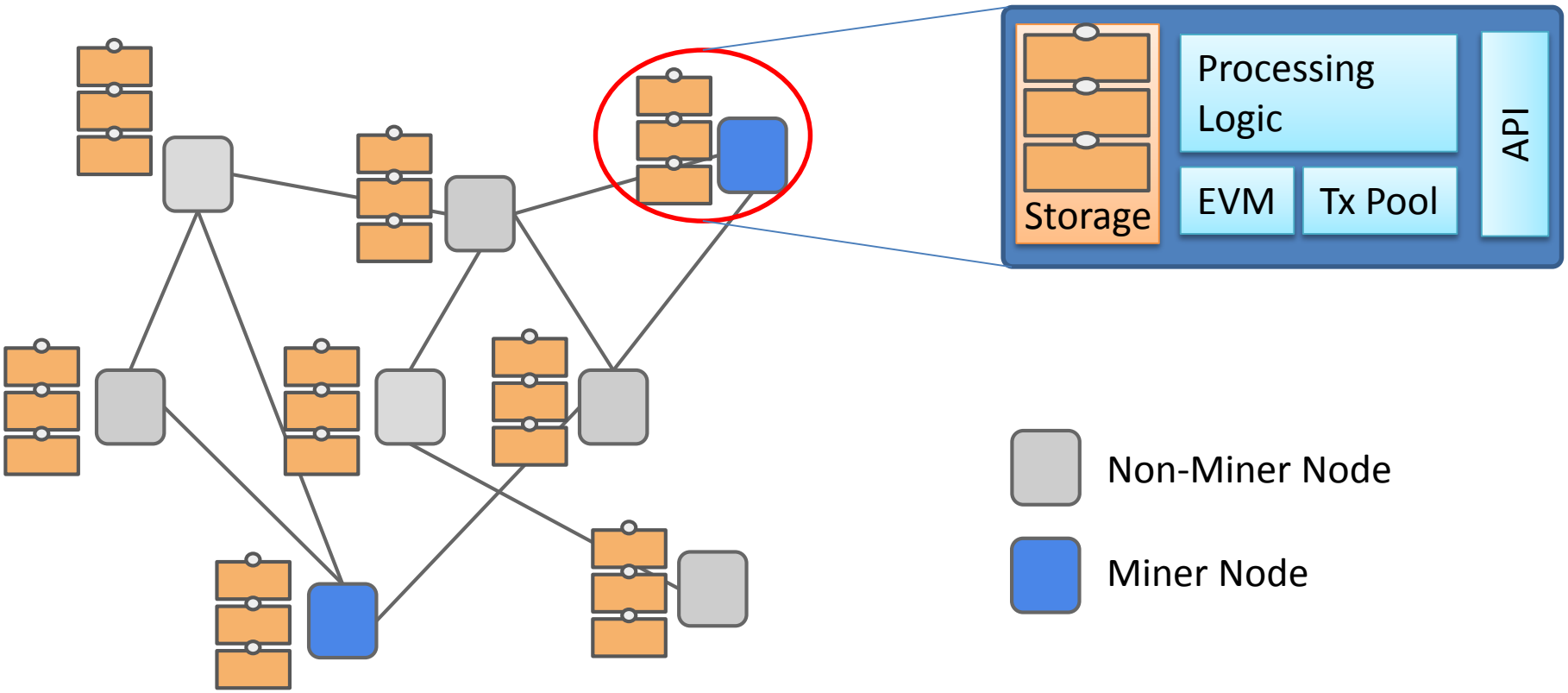
Transaction and State

- The transactions (and its results) are grouped into “blocks.”
  - Miner節點不斷建立空blocks，將所知驗證過的Transactions置入封裝
  - The blocks are chained together (by pointing to the previous block)



  - 每個節點都存有一個blockchain

Blockchain Endpoint (Node)



Smart Contract

- Concept pioneered by Nick Szabo in early 1990s

- From text contracts written in legal languages

將合約條款轉成程式邏輯

自動適時執行

- To code contracts in computer languages

```
contract Escrow { //第三方履約保證
  address buyer;
  address seller;
  address agent;
  function Escrow(address _agent, address _seller) {
   ...
  }
  function release() {
    if (msg.sender == agent)
       suicide(seller); // Send all funds to seller
    else throw;
  }
  function cancel() {
    if (msg.sender == agent)
       suicide(buyer);
       // Cancel escrow and return all funds to buyer
    else throw;
  }
}
```

Smart Contract Example

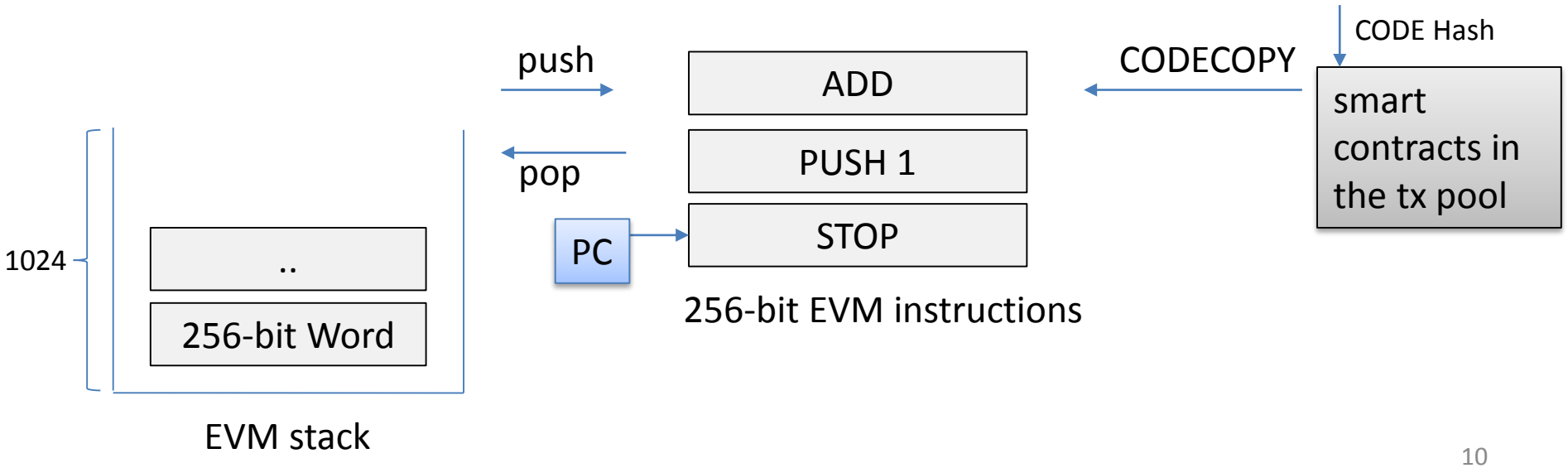
```
contract Test1
{
    uint x=2;
    function addition(uint x) returns (uint y) {
            y=x+2;
    }
}
```

Bytecode (in readable form)

Opcodes PUSH1 0x60 PUSH1 0x40 MSTORE PUSH1 0x2 PUSH1 0x0 SSTORE CALLVALUE PUSH1 0x0 JUMPI JUMPDEST PUSH1 0x45 DUP1 PUSH1 0x1A PUSH1 0x0 CODECOPY PUSH1 0x0 RETURN PUSH1 0x60 PUSH1 0x40 MSTORE PUSH1 0xE0 PUSH1 0x2 EXP PUSH1 0x0 CALLDATALOAD DIV PUSH4 0x989E1731 DUP2 EQ PUSH1 0x1C JUMPI JUMPDEST PUSH1 0x0 JUMP JUMPDEST CALLVALUE PUSH1 0x0 JUMPI PUSH1 0x29 PUSH1 0x4 CALLDATALOAD PUSH1 0x3B JUMP JUMPDEST PUSH1 0x40 DUP1 MLOAD SWAP2 DUP3 MSTORE MLOAD SWAP1 DUP2 SWAP1 SUB PUSH1 0x20 ADD SWAP1 RETURN JUMPDEST PUSH1 0x2 DUP2 ADD JUMPDEST SWAP2 SWAP1 POP JUMP

Ethereum Virtual Machine (EVM)

- A simple VM that runs bytecodes of a contract
  - All full nodes have EVMs
  - 256-bit word, 1024 entries stack
  - Turing complete but limited by gas



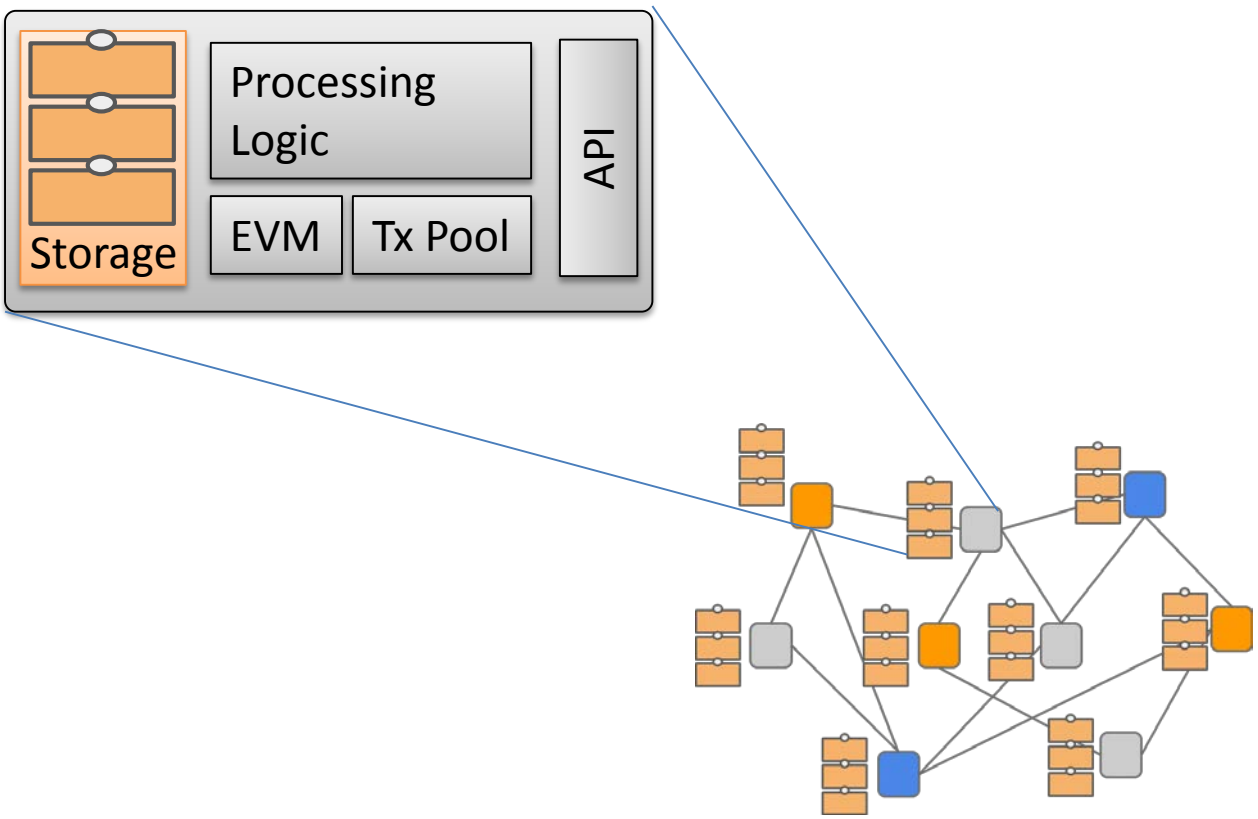
帳戶 Accounts

- 二種類型
  - 使用者 (Externally owned account, EOA)
    - 直接被使用者操作的account
    - 可存錢且可轉(收)錢到(從)另一個account
    - 可建立與呼叫合約
  - 合約 (Contract Account, CA)
    - 由合約所控制的account
    - 可存錢且可轉(收)錢到(從)另一個account

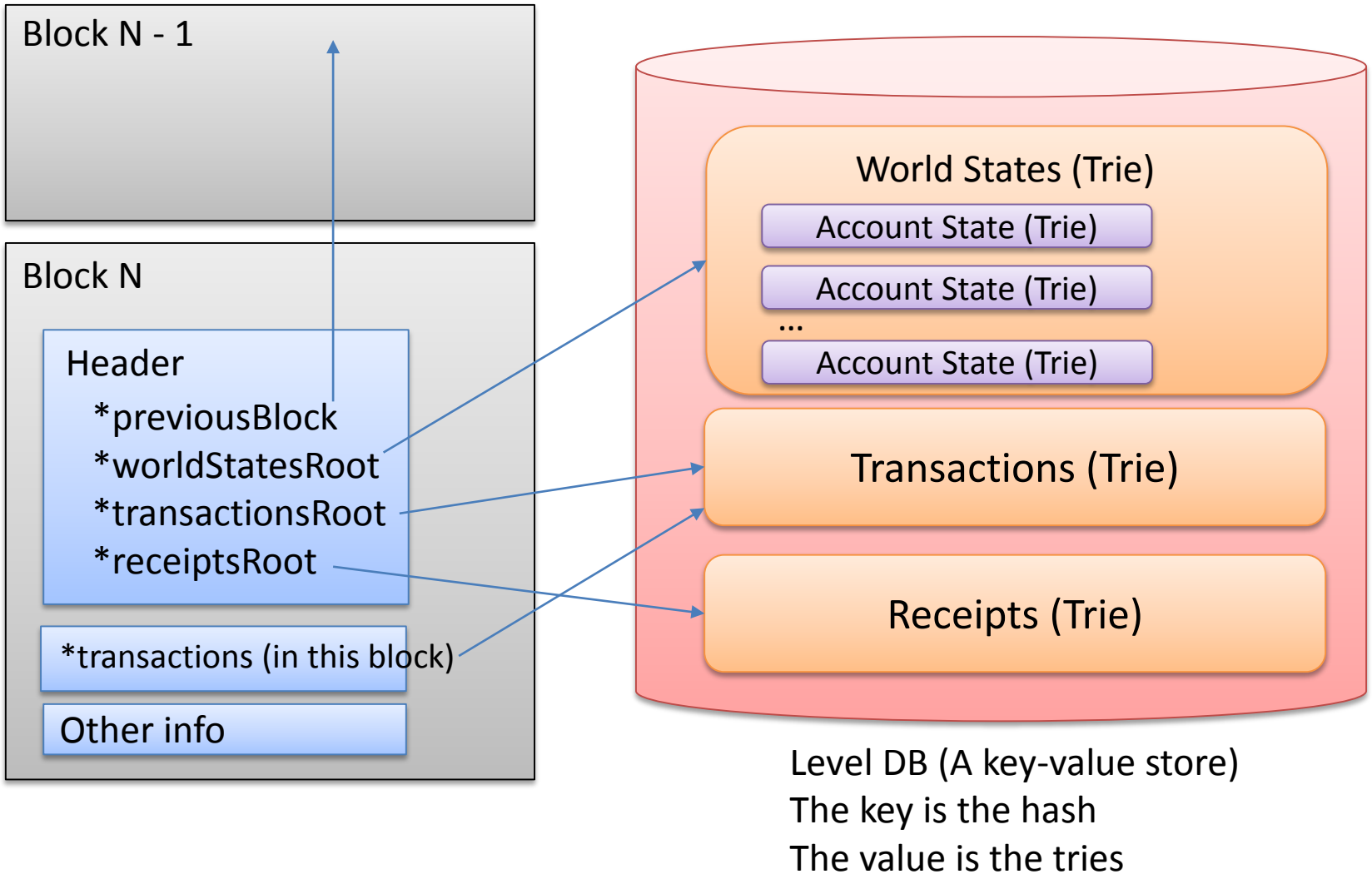
交易 Transactions

- 二種類型
  - 更動資料 Message call transactions
    - 轉錢
    - 透過呼叫合約更動資料/錢
  - 建立合約 Contract creation transactions
- Gas Note: Gas is the not a mining fee (block creation fee)
  - 進行任何交易都需要付的手續費
    - 送出交易就必須付上一筆Gas fee; 不同指令不同價格
  - 手續費不足時，交易會被rollback
  - 手續費太多時，剩下的會在交易執行後歸還

Inside the Shared Ledger



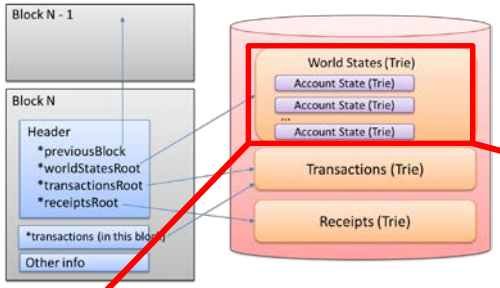
Inside the Shared Ledger



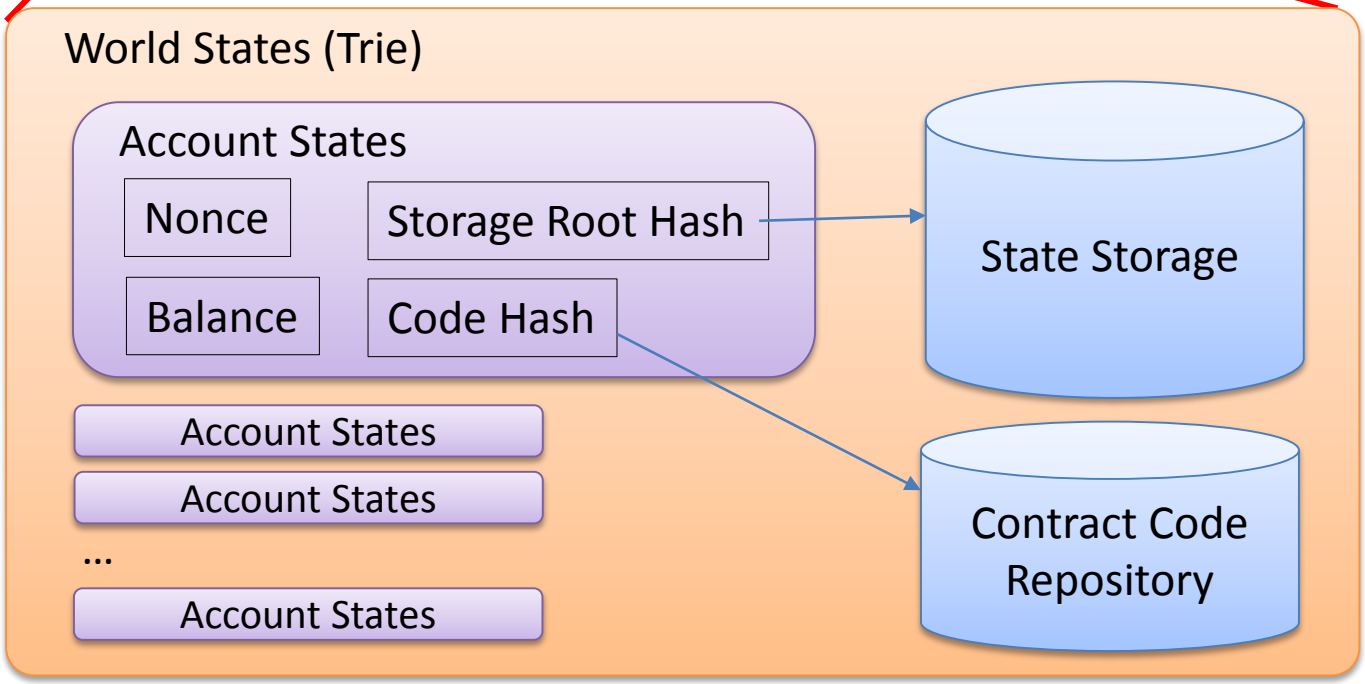
Level DB (A key-value store)
The key is the hash
The value is the tries

[Exploring Ethereum's state trie with Node.js]

Account State and World State

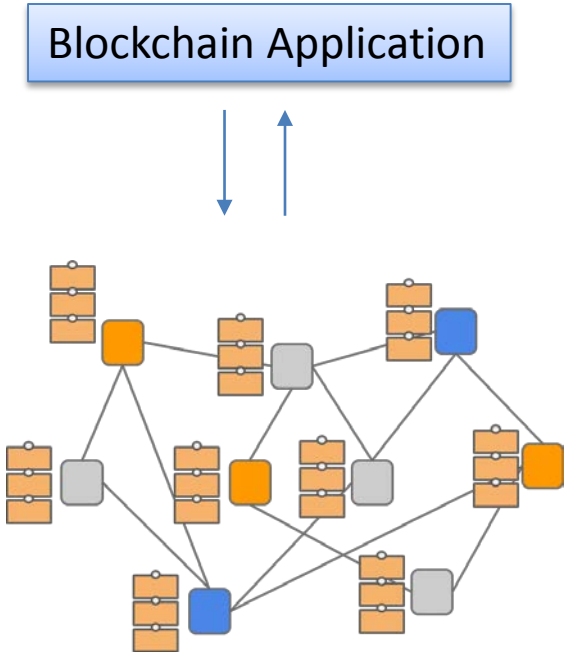


Code Hash: 如果是contract account，則為Smart Contract的hash，否則就是空字串的hash

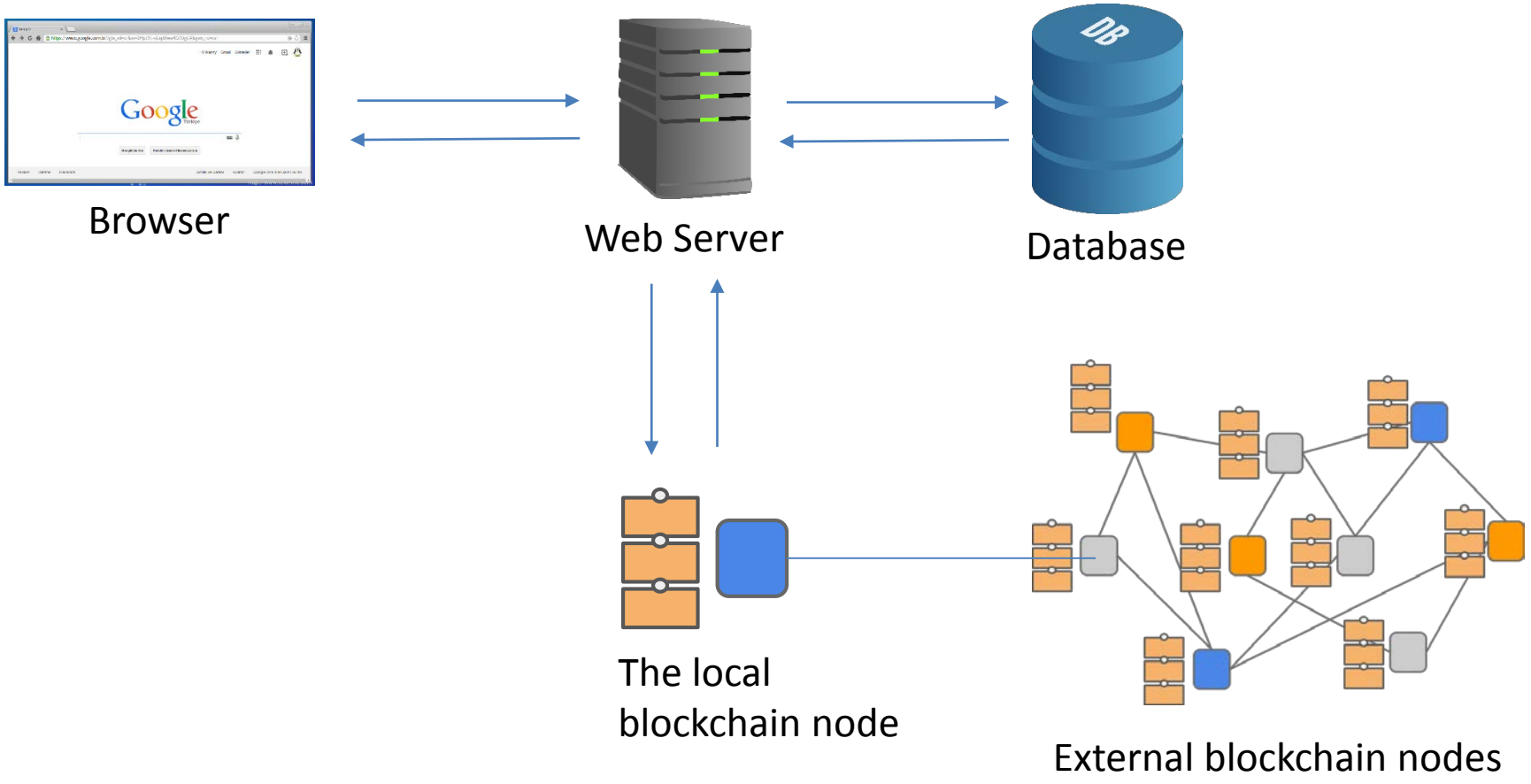


Account nonce: incremented when this account sends a transaction

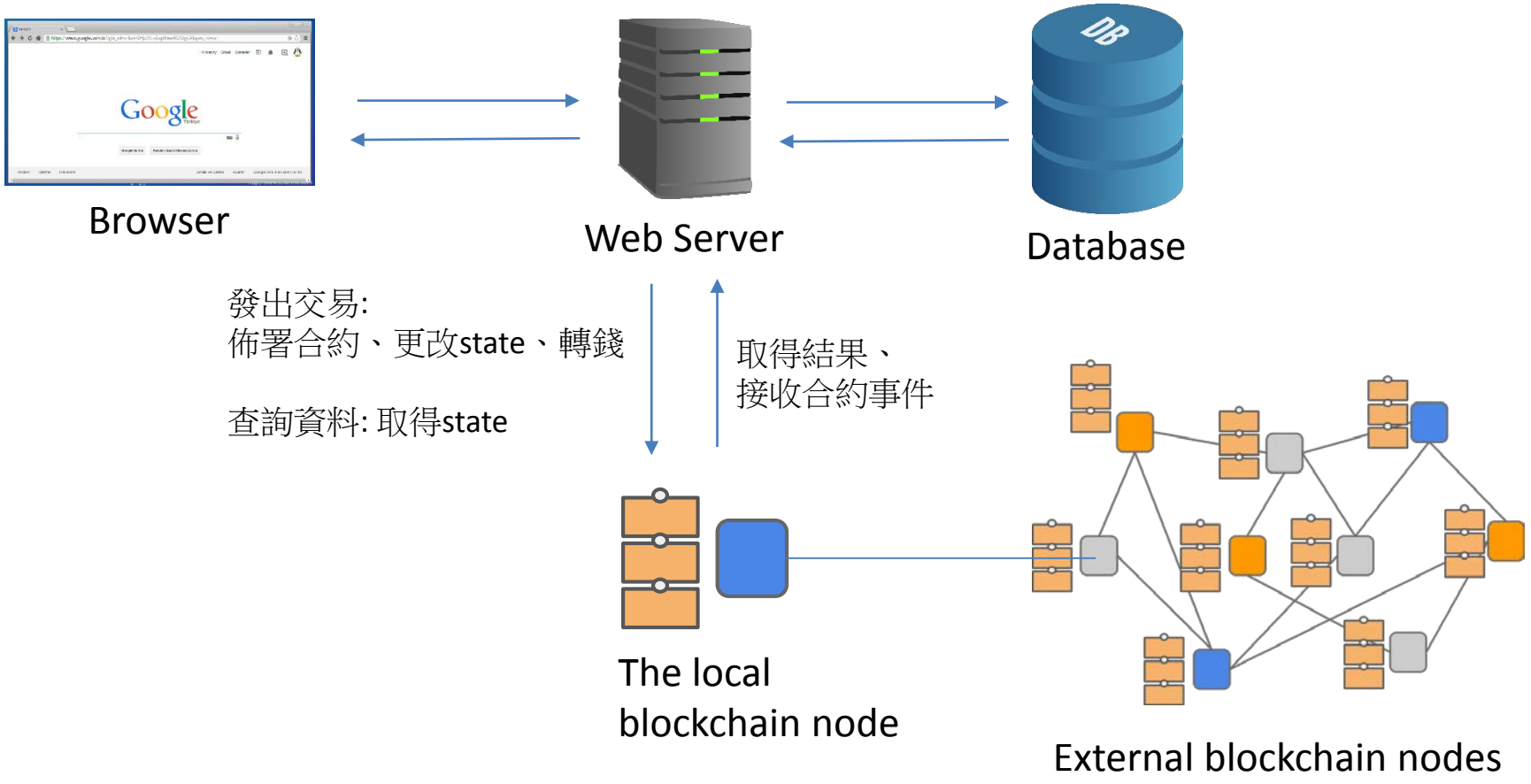
Blockchain應用系統



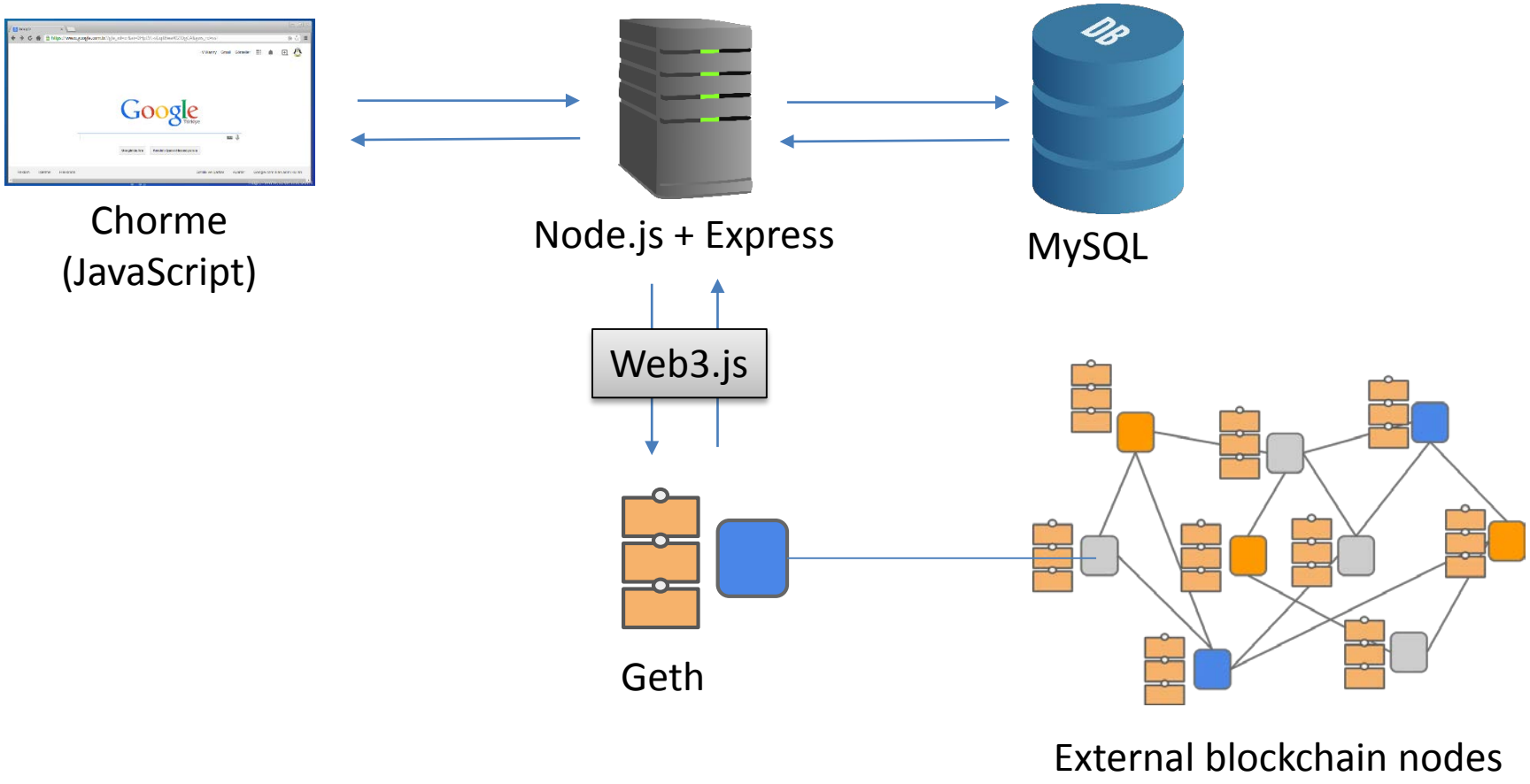
一個典型的Blockchain應用系統



一個典型的Blockchain應用系統



一個典型的Blockchain應用系統



Blockchain交易處理流程

Browser

Web Server

Tx

(execute a contract method)

呼叫者準備一個Message Call Transaction，**將要呼叫的**(Contract) Account**設為**TO

呼叫者將傳入參數放在Transaction**的**data segment**中**

Transaction

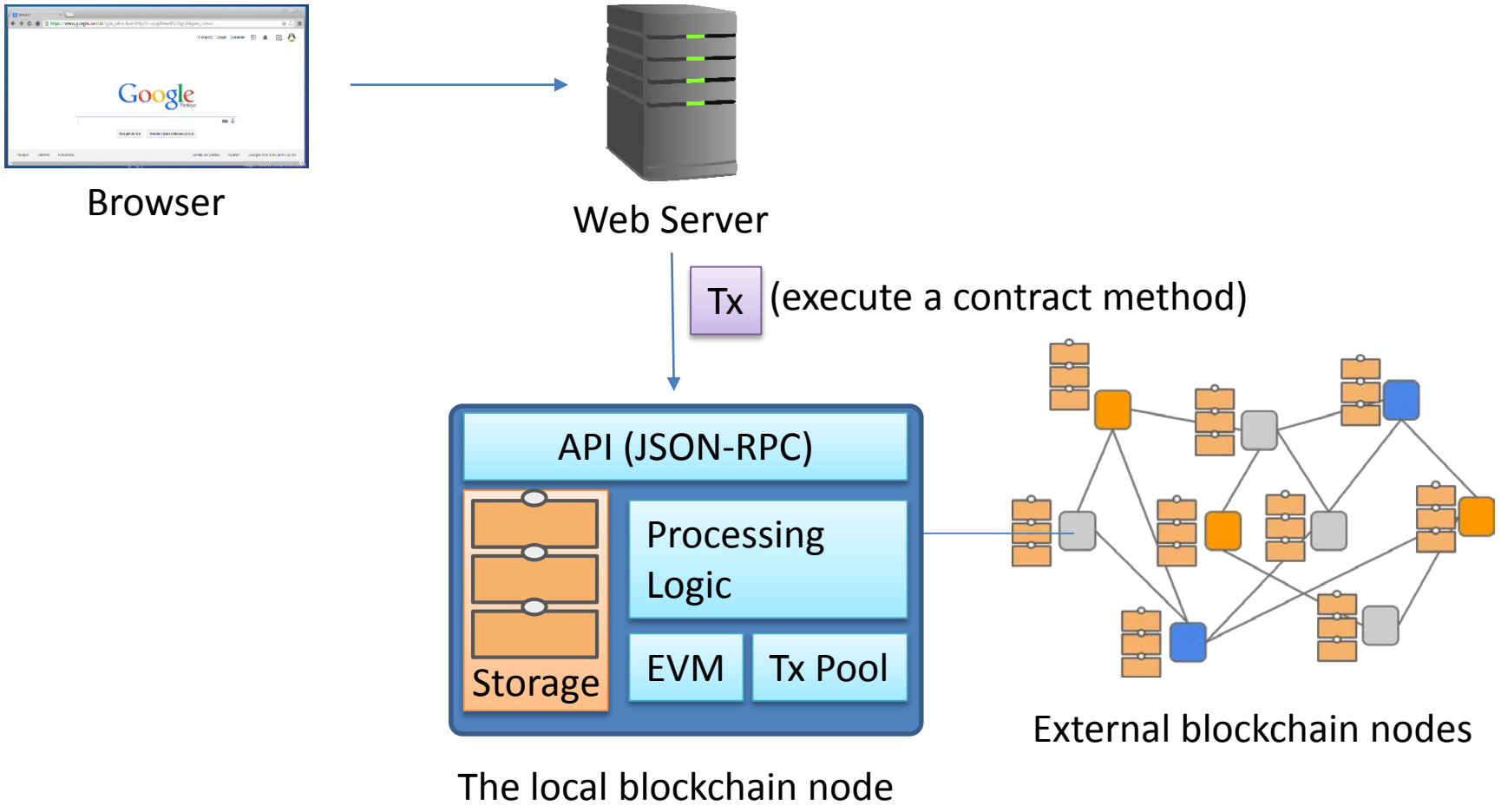
Nonce

TO | FROM | $

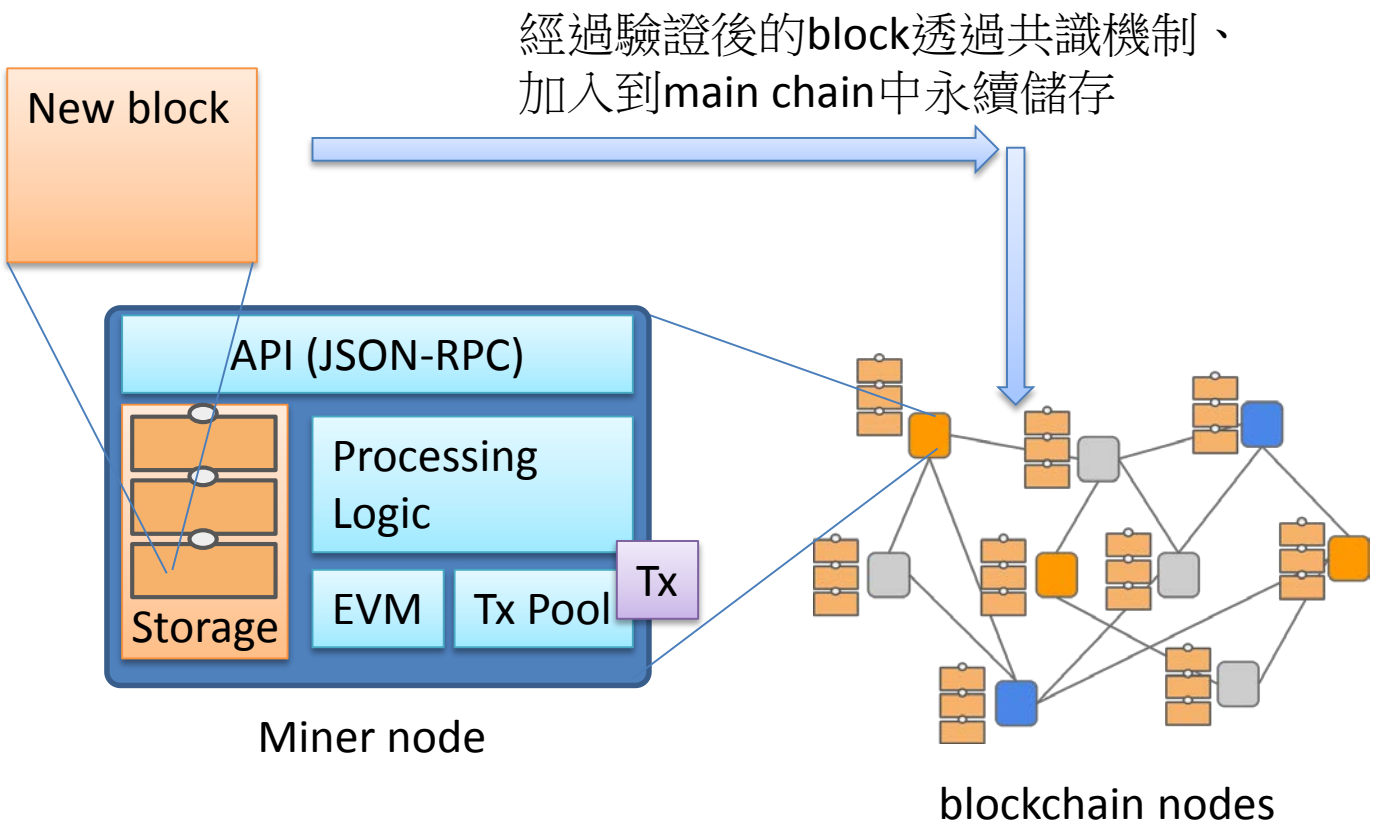
INIT (contract creation)
DATA (message call)

GAS price/limits

Blockchain交易處理流程



Blockchain交易處理流程



大綱

- 簡介
- Blockchain技術
- Blockchain應用
  - Blockchain與智能合約開發
  - Blockchain與物聯網整合
- 未來展望

Note: to make it concrete, in this talk, we use terms and designs of Ethereum when discussing the blockchain concepts

Blockchain 應用系統開發相關議題

- Topics of interests
  - Engineering/ management
    - Agile and Lean processes for Blockchain software development
    - DevOps for Blockchain software
    - Tools for Blockchain software distributed development
  - Design
    - Smart Contract formal specification and verification
    - Smart Contract refactoring/ security /testing
  - Architecture
    - Blockchain software architecture, design and meta-models
    - Architecture for Blockchain and Internet of Things

Our Focus

- DevOps support for blockchain applications
  - BDD-style develop/testing tool for smart contract
  - Middleware for streamlined contract deployment, monitoring, and analyzing
- Blockchain and IoT integration
  - From a software architecture perspective

BDD-style Develop/Testing Tool for Smart Contracts

- Challenges of Smart Contract development
  - Development burdens
    - Must go through a complex installation and configuration process
      - Setup the infrastructure; Create an account (key pairs); Execute miners; Generate and deploy bytecodes
  - Hard to verify the contract logic
    - Must have deep understanding of domain knowledge
    - Whether the contracts accurately reflect the business logic

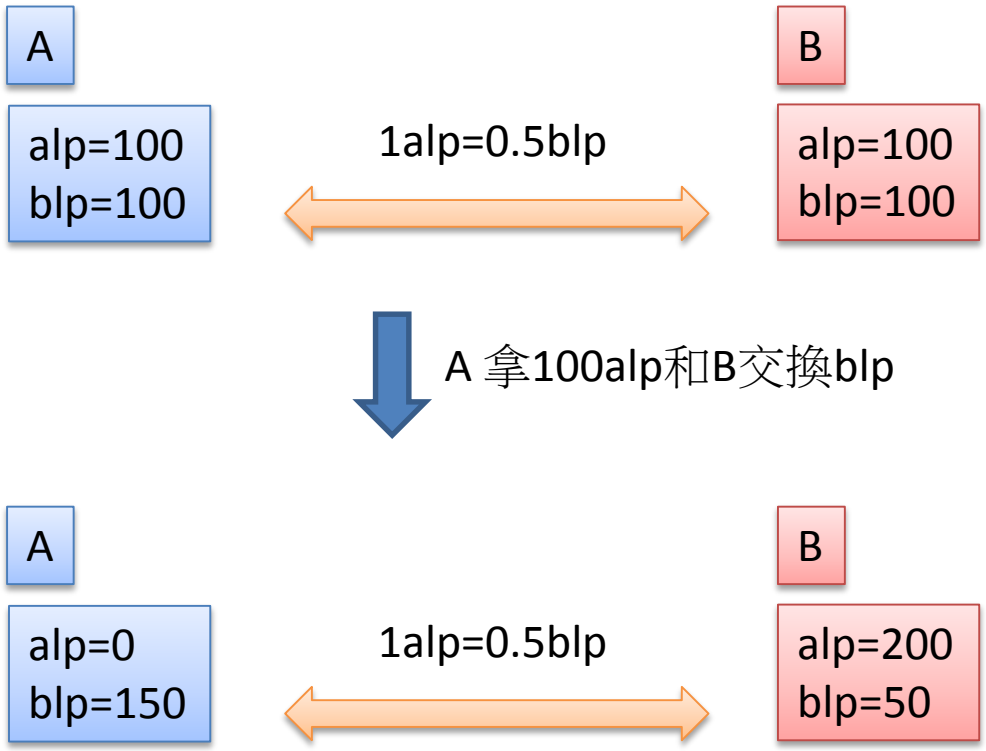
Approach: Behavior-Driven Development

- Popular in the Agile community
  - Based on executable specifications
    - Spec. derived from domain knowledge
    - Specify the expected behaviors in a structured way
    - Complement TDD (Test-Driven Development)
- Validates both “requirement” and “functions”
  - Building the right software (What)
  - Building the software right (How)

Executable Specification

- 需求規格
  - 主要說明「What」而非「How」
- 規格的演進與維護
  - 生命周期長的系統，應該只保留一份需求規格
    - 最精準的規格就是code: 但客戶看不懂，很難察覺需求不符的問題
    - 另外準備一份word or 描述文件: 這樣就有同步問題
  - Executable Specification
    - 以自然語言寫作，因此客戶可確認是否有偏離其需求
    - 以實施例(Examples)來驅動測試
    - 成為Living documentation (反映最新版本功能的說明文件)

範例: 點數交換


A
alp=100
blp=100
1alp=0.5blp
B
alp=100
blp=100
A 拿100alp和B交換blp
A
alp=0
blp=150
1alp=0.5blp
B
alp=200
blp=50

Runnable Specification (Gherkin)

```
Feature: Exchange loyalty points between two parties
  In order to exchange loyalty points between company A and company B
  As a loyalty point user
  I want to exchange loyalty points of A (alp) for loyalty points of B (blp) or
            exchange loyalty points of B (alp) for loyalty points of A (blp)
            according to corresponding preset exchanging rate and rules.
```

Feature name

Feature description (free form)

Scenario name (a feature consists of 5-20 scns)

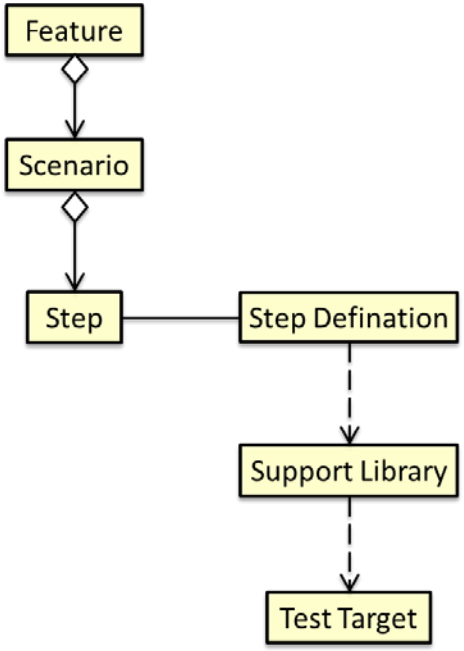
```
Scenario Outline: Company A exchanges alp for blp
   Given the exchange rate is 1alp=0.5blp
   And original alp account of A is 100
   And original blp account of A is 100
   And original alp account of B is 100
   And original blp account of B is 100
   When A want to exchange 100 alp for blp with B
   Then alp account of A should be 0
   And blp account of A should be 150
   And alp account of B should be 200
   And blp account of B should be 50
```

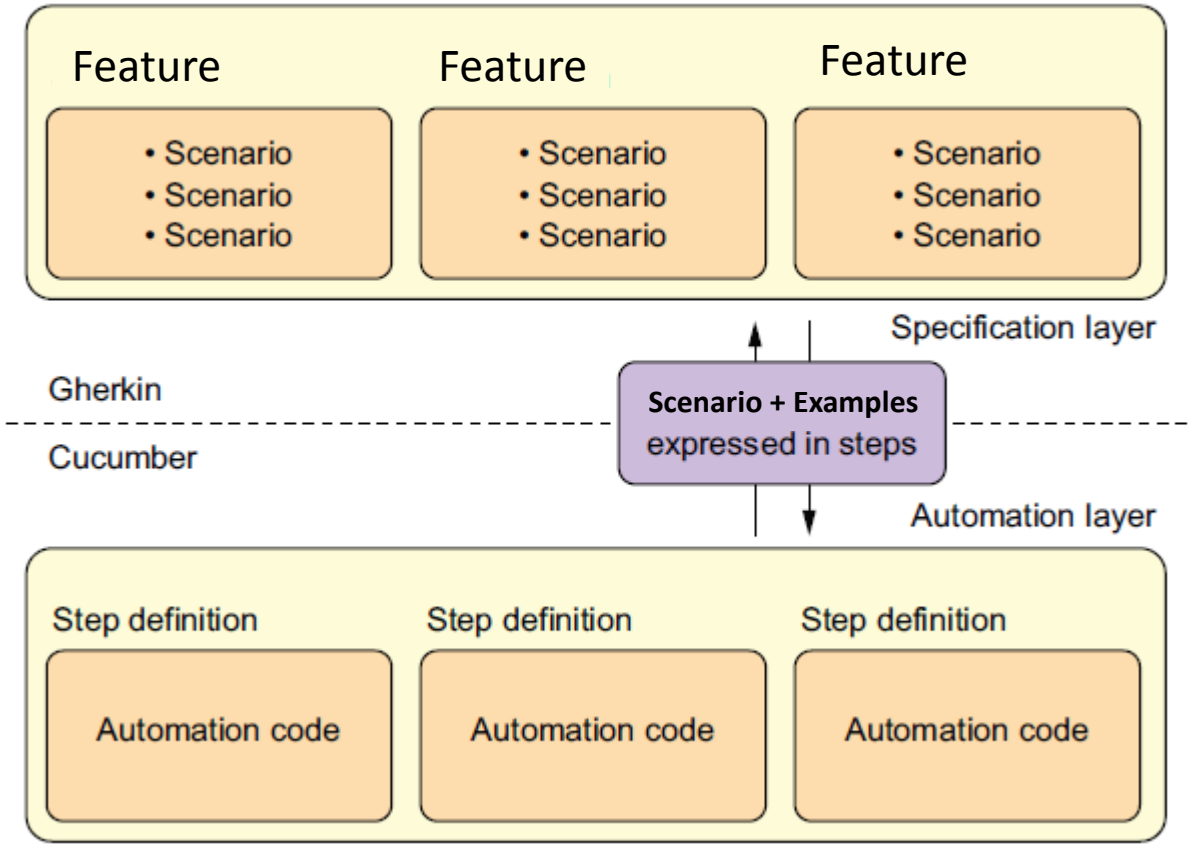
Runnable Specification

```
Feature: Exchange loyalty points between two parties
  In order to exchange loyalty points between company A and company B
  As a loyalty point user
  I want to exchange loyalty points of A (alp) for loyalty points of B (blp) or
            exchange loyalty points of B (alp) for loyalty points of A (blp)
            according to corresponding preset exchanging rate and rules.

  Scenario Outline: Company A exchange alp for blp
    Given the exchange rate is 1alp=0.5blp
    And original alp account of A is <alp account orig>
    And original blp account of A is <blp account orig>
    When A want to exchange <alp to exchange> alp for blp
    Then alp account of A should be <alp account new>
    And blp account of A should be <blp account new>

    Examples:
    | alp to exchange |  alp account orig | alp account new | blp account orig | blp account new |
    | 100             |  100              | 0               | 100              | 150             |
    | 20              |  100              | 80              | 100              | 110             |
    | 110             |  100              | 100             | 100              | 100             |
```

Example中以表格提供測試數據

Specification如何被執行





Implementation

- Web UI
  - BDD-style on-line contract development
  - Guiding the developer through the steps of the process
- Web API
  - Project, Testing and Contract management
  - Adapter to the baseline frameworks
- Infrastructure
  - Baseline frameworks (cucumber/mocha/solc/Web3.js)
  - Ethereum node

Implementation

- Target platform
  - Ethereum
  - Contract Language: Solidity
  - Integration: Web3.js
- Baseline testing frameworks
  - Unit Test: Mocha
  - Integration Test: Cucumber.js
- Implementation platform
  - Node.js (LTS)
  - Express 4.x

Implementation

Gherkin IDE (Executable Spec.)

Step Defs IDE (Integration Test)

Mocha Tests IDE (Unit Test)

Contract Management GUI

Process Controller (BDD流程控制)

Admin Console

Web-based BDD合約整合開發測試環境

Dev. Data Repository (專案資料、合約metadata、測試報告、account、系統設定)

Dev. Services

Integration Testing Services

Unit Testing Service

Contract Management Service (Gas調控輔助、abi/bin產生/合約佈署、尋找與呼叫)

BDD合約整合開發測試服務

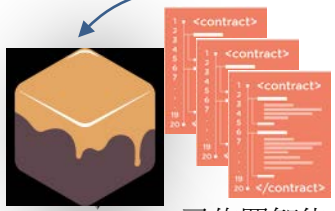
Cucumber

Mocha

Solc

Web3 JavaScript Đapp API

現有工具與技術



Ganache (Mock)

已佈署智能合約

eNode

已佈署智能合約

Ethereum Blockchain Network

小結

- 實作基於BDD的智能合約整合測試平台
  - 確保業務邏輯正確性
    - BDD以業務場景為核心，與TDD互補確保合約符合立約雙方需求
  - 兼具合約開發與測試功能
    - 藉由GUI與自動化，降低合約開發與測試難度
  - 有效步驟導引回饋
    - 降低程式除錯與修正成本
    - 節省整體開發時間

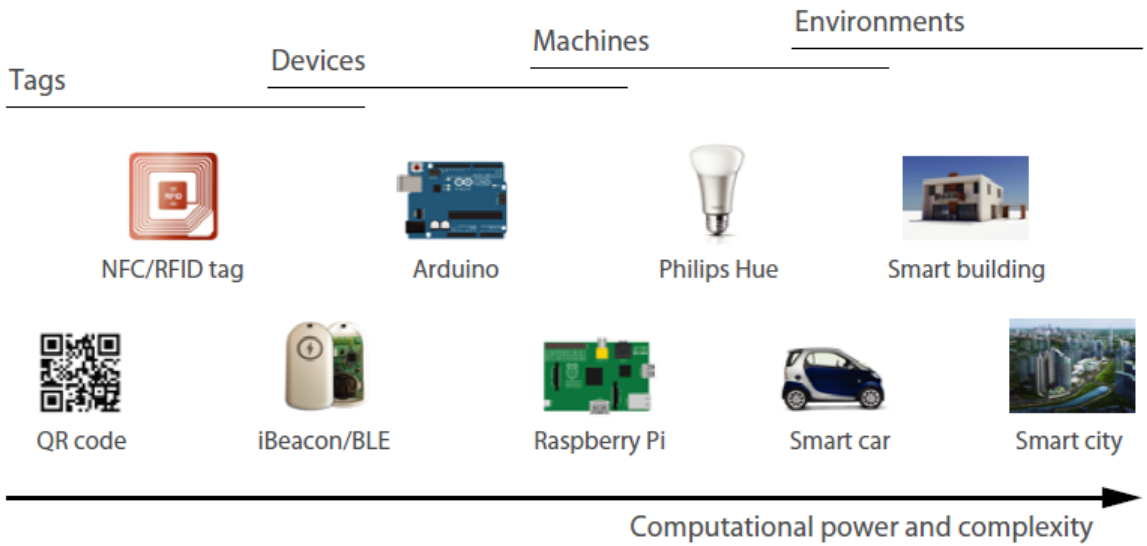
大綱

- 簡介
- Blockchain技術
- Blockchain應用
  - Blockchain與智能合約開發
  - Blockchain與物聯網整合
- 未來展望

Note: to make it concrete, in this talk, we use terms and designs of Ethereum when discussing the blockchain concepts

Smart Things

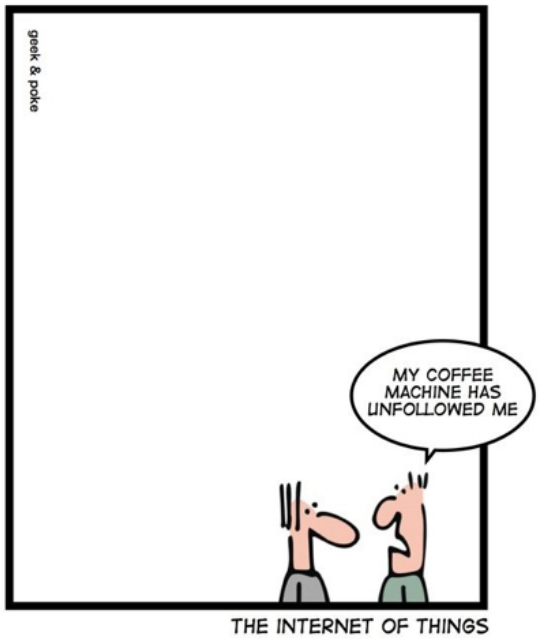
- Smart Things
  - 具有計算能力的日常物品
- 具備「計算能力」的意函
  - 感測能力
  - 致動能力
  - 邏輯與運算能力



圖片來源: D.D. Guinard & V. M. Trifa, Building the Web of Things, Manning, 2015

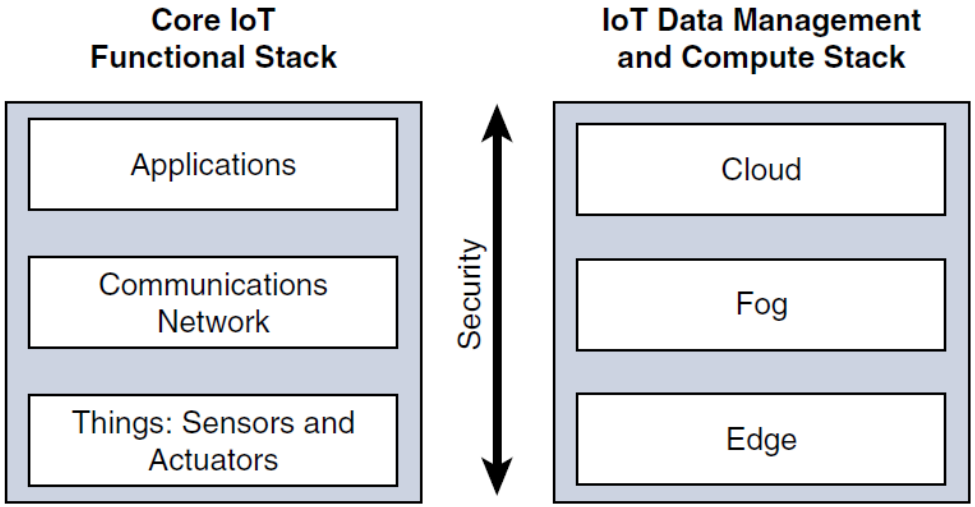
Internet of Things

- 願景
  - 所有實體裝置和網頁一樣，能透過網路互連
- 定義
  - 一群智慧物件(Smart Things)，它們
    - 除設置時期之外，不需要人為操作
    - 可透過Internet互連
    - 可透過ICT技術被發現、監控、互動



IoT Reference Architecture

- oneM2M
  - ETSI M2M Technical committee (2008)
- IoT Reference Model
  - A IoTWF (IoT World Forum) standardized Architecture



A simplified IoT ref. architecture suggested by Hanes et al. (2017)

IoT技術應用面臨問題

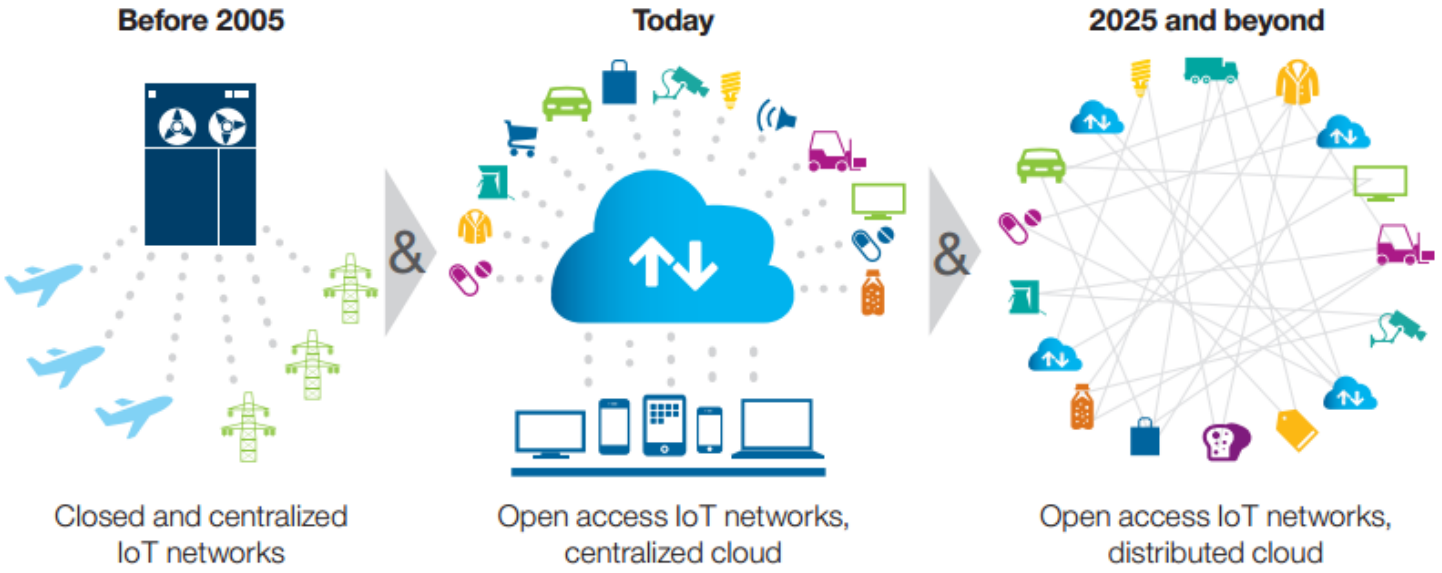
- Success only in high-value applications
  - Jet engine monitoring
  - Smart metering
  - Healthcare management
- Demands are slow to take off in other areas
  - Not so good: heavy industrial and home automation
  - Failed: consumer electronic
    - smart toothbrushes and refrigerators
- The market expects 10-20 times revenue!

IoT技術應用面臨問題

- Cost of maintenance and network connectivity
  - Software/ firmware updates
- Ensure privacy and security in IoT
  - Perfect trust is not possible in current Internet
    - “security through transparency” better than “security through obscurity”

P2P and Trustless IoT

- Device democracy
  - Trustless p2p messaging
  - Secure distributed data sharing
  - Robust and scalable form of consensus



Blockchain-based IoT Services (B-IoT)

- Unlocking excess capacity of physical assets
  - IoT
    - Enables the digitization of sell and deliver of physical assets
  - Blockchain
    - Facilitates distributed and secured transactions
- Creating liquid, transparent marketplaces
  - Three key elements of e-commerce became instant and comprehensive : search、use、pay
- Radical re-pricing of credit and risk
  - personalized risk and credit profiles

Case Slock.it

- 一個基於Blockchain與Smart Contract的電子鎖系統

Contract

```
Contract {
        Owner_address
        User_address

        deposit
        price

        open()
        close()
        rent()
        returnLock()
        ...
}
```

Owner_address：為鎖的擁有者address。
User_address：租借人(能夠控制鎖的人)的address

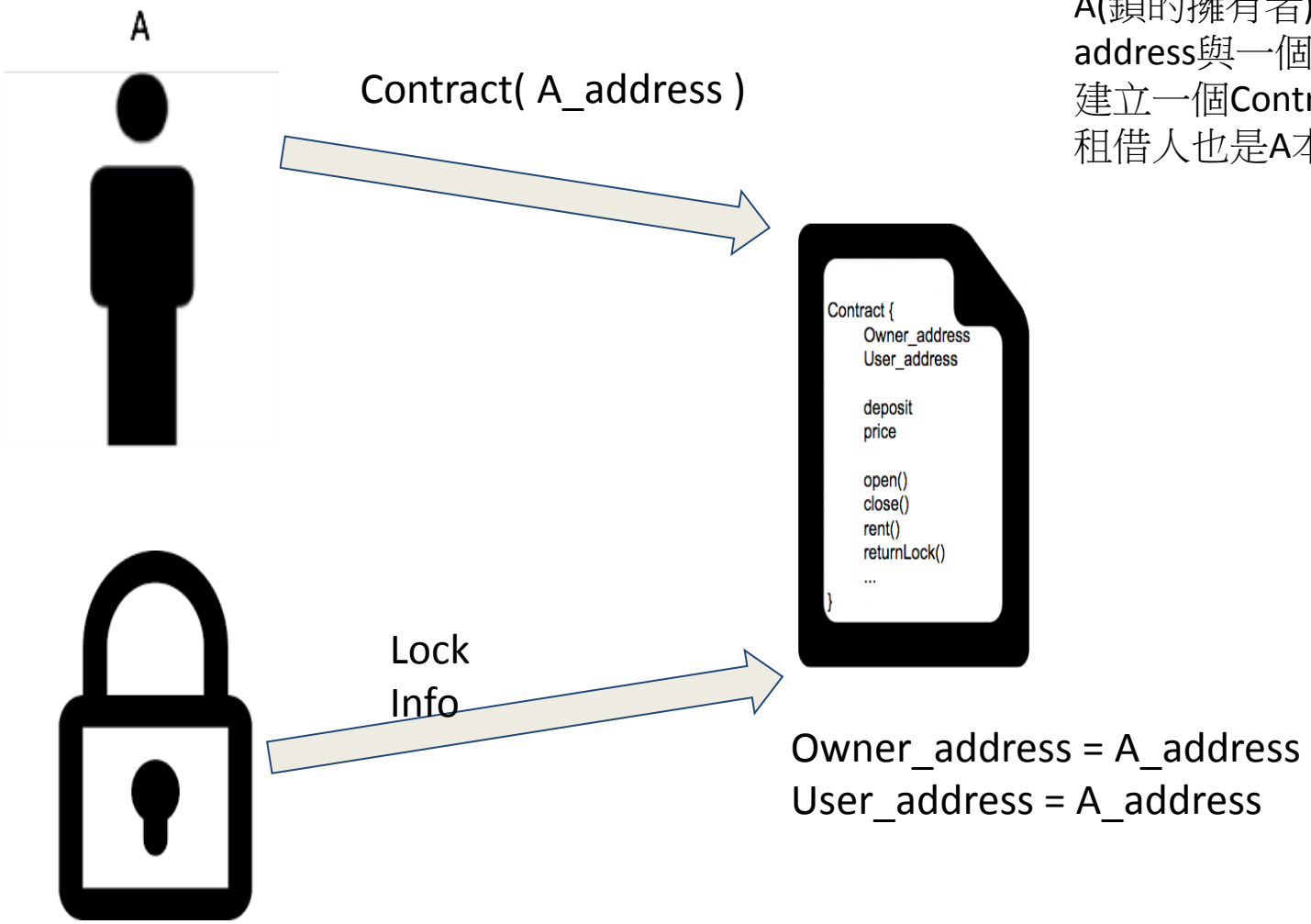
deposit：擁有者要求使用者需預先付的押金。
price：租用價格

open()：打開鎖(僅限使用者)
close()：鎖住鎖(僅限使用者)

rent(address)：提供使用者的address與押金並租借鎖。如果支付的押金足夠就將租借人記為此人。

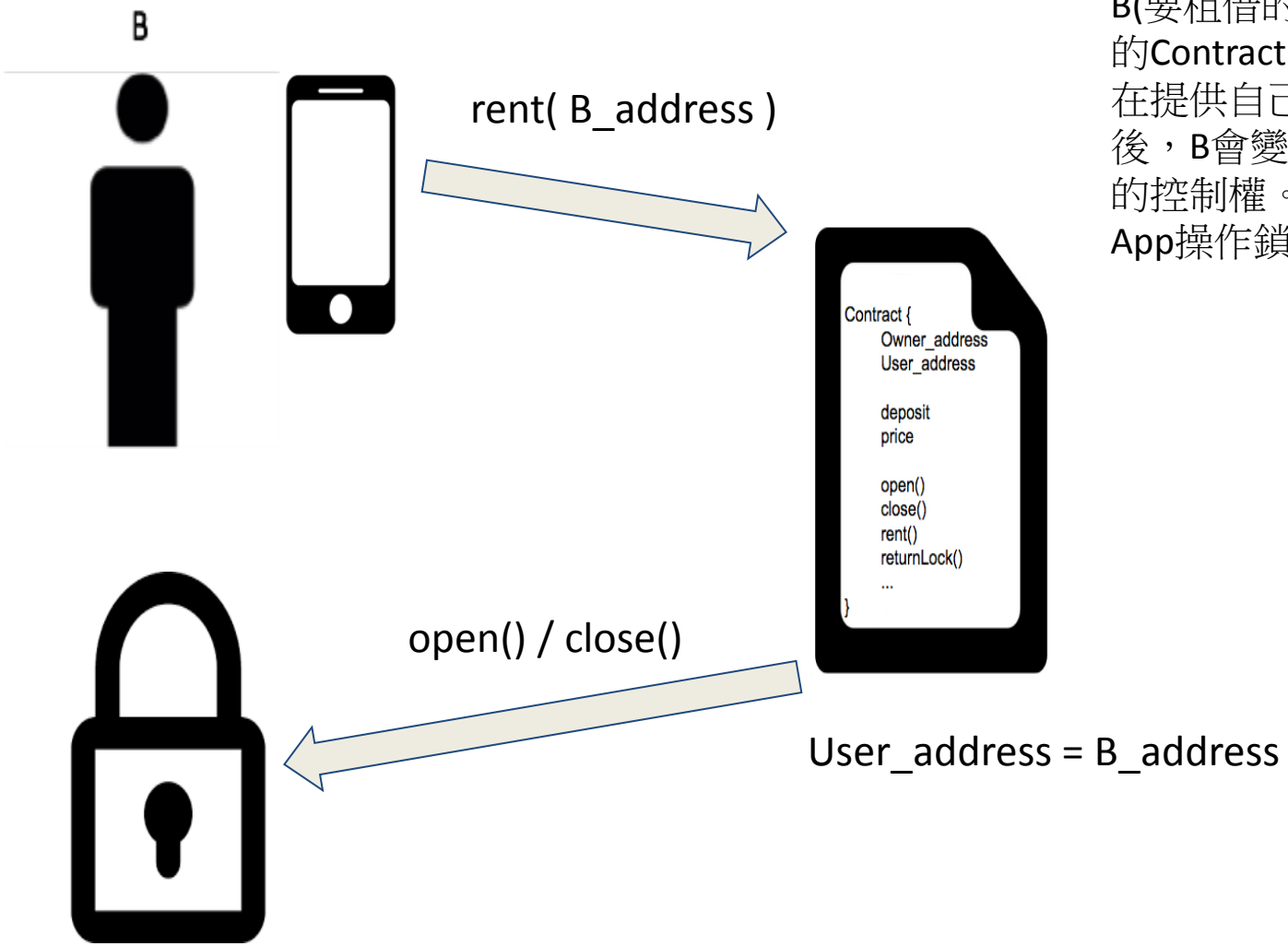
returnLock()：將鎖的擁有權還給擁有者(僅限使用者)

Owner Create Contract



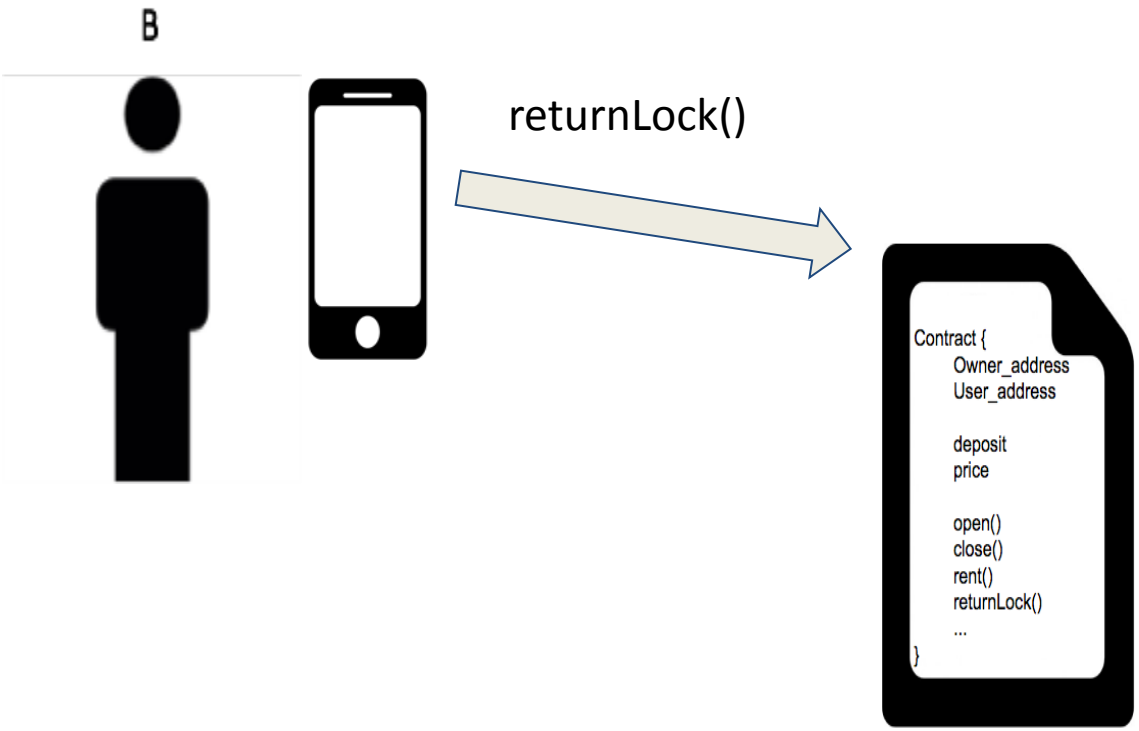
A(鎖的擁有者)提供自己的address與一個鎖的資訊後，建立一個Contract。(此時的租借人也是A本人)

User Rent Lock



B(要租借的人)透過App找到鎖的Contract。
在提供自己的address與押金後，B會變成User，拿到了鎖的控制權。這時B就可以透過App操作鎖。

User Return Lock

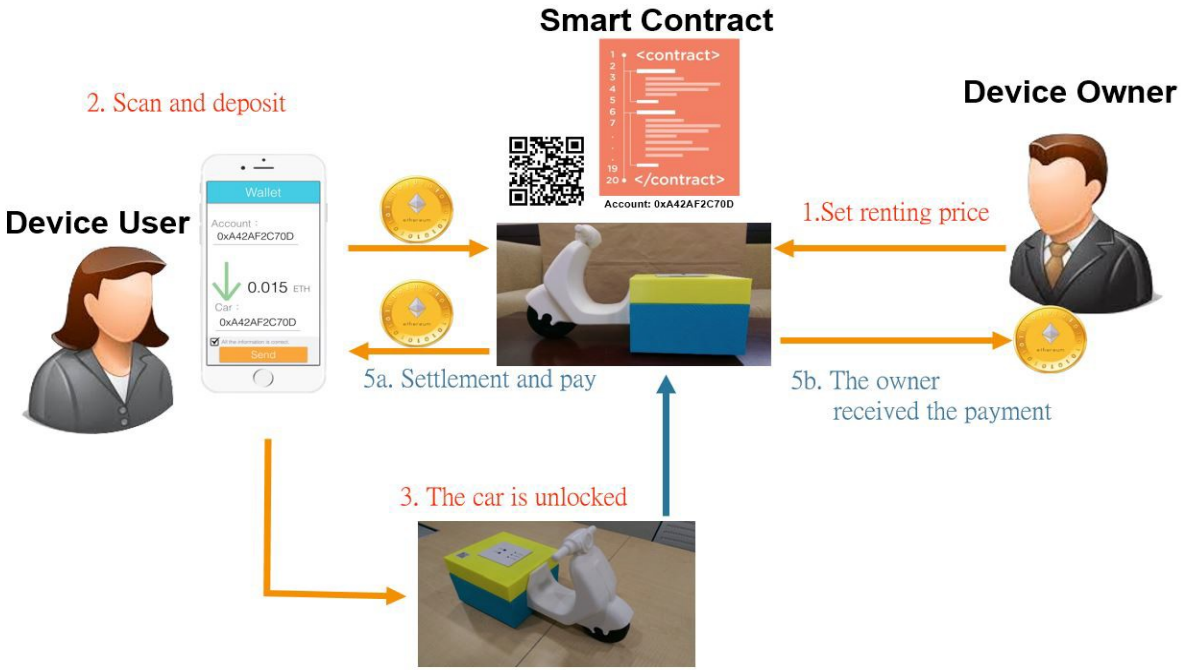


B使用完後，可以透過App將鎖還給A，這時Contract會透過使用時間與價格自動計算B需要付出的費用，將押金扣除費用後的餘額還給Ｂ，將費用給Ａ。同時將User變成A，表示使用權限回到Ａ。

User_address = A_address

Example: 智能租車

- Internet of Things Renting Platform based on Smart Contracts on Blockchain



https://www.youtube.com/watch?v=CbsAFStbyBg

政大校園點數系統


校內行政單位
發行各種點數
T O K E N
取得點數,
存入錢包
我的帳戶
我的點數
教務處點數 29000
置物櫃點數 27000
宿舍點數 28300
學務處點數 27600
政大點數系統
交換
轉出
轉入
好友
交易歷史
我的帳戶
我的賣單
移轉點數
給好友
P2P交換
點數
消費點數
使用校內
IoT服務

Main Issues of B-IoT

- Burdens after becoming a blockchain endpoint
  - Storage
    - Blockchain endpoints store entire global states and their history!
  - Bandwidth
    - Must continuously keep track of global states
  - Power consumption
    - A blockchain endpoint does no have time to sleep

Summary

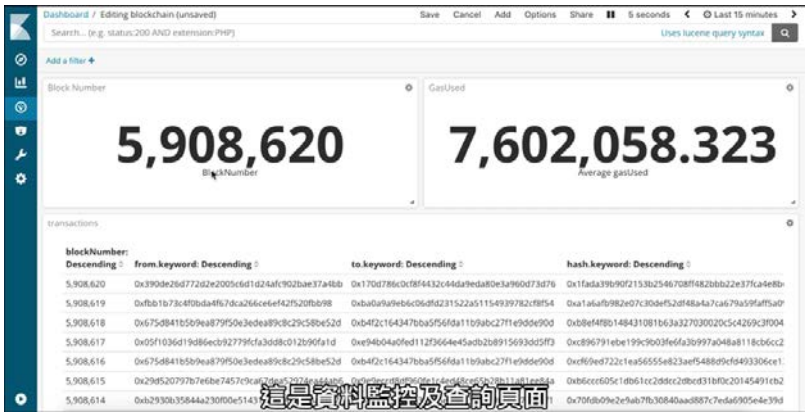
- “Things” Democracy
  - Trustless p2p messaging
  - Secure distributed data sharing
  - Robust and scalable form of consensus
- B-IoT有機會成為IoT進一步發展的解決方案
  - 降低了營運成本 (避免集中的大server)
  - 提高安全與隱私 (DDos)
  - 提供原生Billing layer

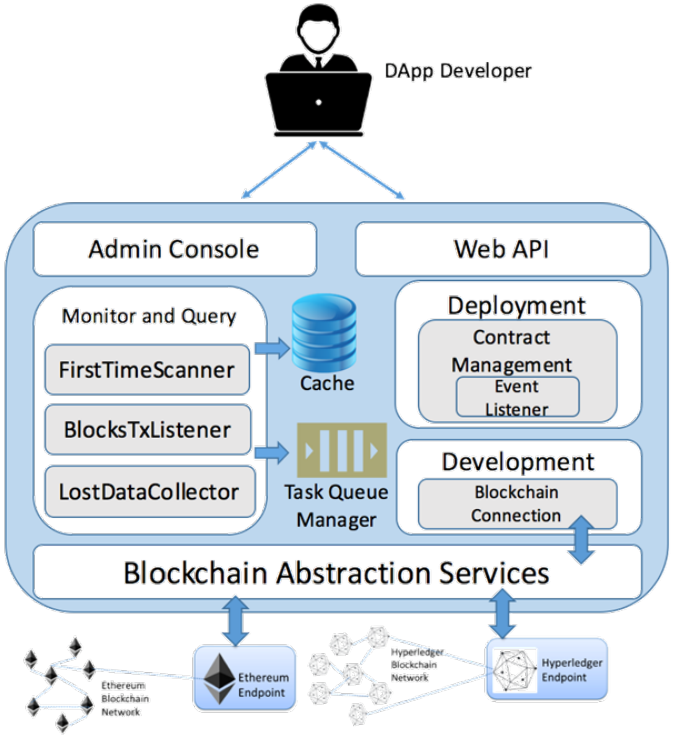
Ongoing Works

- 多元化應用實驗計畫: 打造政大為區塊鏈校園 (Crypto Campus @NCCU)
  - 區塊鏈點數錢包：校內學生獎勵機制的實驗用
  - 區塊鏈IoT應用：電子置物櫃，共享單車
  - 區塊鏈網路投票：學生會選舉，結束後毋須監票開票，投票人可於結束後自行驗票
  - 電子成績單於區塊鏈的存證與查證

Ongoing Works

- 支援智能合約開發、佈署與監控的中介軟體
- 主軸
  - 提供簡便的部署智能合約方法
  - 提供自動化合約事件追蹤機制
  - 提供有效率的查詢及監控平台





https://www.youtube.com/watch?v=1Es80FCHQ14