Making Blockchain Real for Business

Explored

V2.3, 11 Nov 16

© 2016 IBM Corporation
Demo: Trade Solution

URL: https://www.youtube.com/watch?v=r0LsnzAe1Yg
Actors, Documents & Events

Flow of goods from **Exporter** Malay Grower (Malaysia) to **Importer** Berlin’s Fresh Cuts

- Malay Grower
- Berlin’s Fresh Cuts

Registers 3 documents on the private shared ledger

- **Letter of Credit**
- **Bill of Lading**
- **Phyto-sanitary Certificate**

Events

- Letter of Credit issuance
- Shipping & Inspection Progress
- Money Transfer
Blockchain Authentication Nodes

- International Shipping
- Bank of Berlin
- Port of Hamburg
- Port Kelang

Blockchain Ledger Participants

- Shipping Company: International Shipping
- Exporter: Malay Grower
- Importer: Berlin's Fresh Cuts
- Port Authorities: Port of Hamburg, Port Keland
- Customs Authority: Berlin Customs
- Importer's & Export's Banks: Bank of Berlin, Malay Bank
Key Blockchain Benefits

- **Increased Speed & Reduced Errors**: Blockchain-based document store (including Smart Contracts) replaces physical documents and processes, so physical transportation of shipment documents not required, leading to fewer delays.

- **Privacy & Transparency**: Reduced errors: Rekeying of information eliminated across the ecosystem.

- **Scalable Ecosystem**: Real-time: Shipment information and status accessible to relevant stakeholders instantaneously and simultaneously.
Key Blockchain Benefits

- Increased Speed & Reduced Errors
- Privacy & Transparency
- Scalable Ecosystem

- Privacy: Limits views of information to the appropriate parties
- Transparency: Provides end-to-end visibility of cargo information for all relevant stakeholders (customs/govt., insurers, etc.)
  - Shipping Information: Contents, Destination, etc.
  - Status: Location, Ownership (can change during process), Insurance, etc.
Key Blockchain Benefits

- Increased Speed & Reduce Errors
  - Network neutrality: No single owner of network; participants can easily ‘plug-in’ to the information value chain to enable trade

- Privacy & Transparency
  - Ecosystem adoption: Neutral network lowers adoption barrier for key participants with ‘veto power’ (e.g., governments, insurers, etc.) vs. centralized solution, because
    - Benefits of reducing paper and digitization accrue to overall industry
    - No one player gains unfair advantages

- Scalable Ecosystem
  - Resiliency: Blockchain’s decentralized ledger is immutable and validated by all participants, fostering trust
The Participants in a Blockchain Network

Systems Context
The Participants in a Blockchain Network

- **Regulator (R)**: performs oversight
- **Blockchain User (U)**: accesses security certificates
- **Blockchain Developer (D)**: creates applications
- **Blockchain Network Operator (O)**: operates
- **Traditional Data Sources**

B2B transactions flow to the Certificate Authority (CA), which issues security certificates. These certificates are used to access the data on traditional processing platforms. The blockchain network allows access to logic, creating applications operated by the blockchain network operator. The regulator performs oversight to ensure compliance with regulations.
## Blockchain Participants

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Blockchain User</strong></td>
<td>the business user, operating in a business network. This role interacts with the Blockchain using an application. They are not aware of the Blockchain.</td>
</tr>
<tr>
<td><strong>Blockchain Regulator</strong></td>
<td>the overall authority in a business network. Specifically, regulators may require broad access to the ledger’s contents.</td>
</tr>
<tr>
<td><strong>Blockchain Developer</strong></td>
<td>the developer of applications and smart contracts that interact with the Blockchain and are used by Blockchain users.</td>
</tr>
<tr>
<td><strong>Blockchain Network Operator</strong></td>
<td>defines, creates, manages and monitors the Blockchain network. Each business in the network has a Blockchain Network operator.</td>
</tr>
<tr>
<td><strong>Certificate Authority</strong></td>
<td>manages the different types of certificates required to run a permissioned Blockchain.</td>
</tr>
<tr>
<td><strong>Traditional Processing Platform</strong></td>
<td>an existing computer system which may be used by the Blockchain to augment processing. This system may also need to initiate requests into the Blockchain.</td>
</tr>
<tr>
<td><strong>Traditional Data Sources</strong></td>
<td>an existing data system which may provide data to influence the behavior of smart contracts.</td>
</tr>
</tbody>
</table>
The Components in a Blockchain
Component Model
Blockchain Components

- **Ledger**: contains the current world state of the ledger and a Blockchain of transaction invocations.
- **Smart Contract**: encapsulates business network transactions in code. Transaction invocations result in gets and sets of ledger state.
- **Consensus Network**: a collection of network data and processing peers forming a Blockchain network. Responsible for maintaining a consistently replicated ledger.
- **Membership**: manages identity and transaction certificates, as well as other aspects of permissioned access.
- **Events**: creates notifications of significant operations on the Blockchain (e.g. a new block), as well as notifications related to smart contracts. Does not include event distribution.
- **Systems Management**: provides the ability to create, change and monitor Blockchain components.
- **Wallet**: securely manages a user’s security credentials.
- **Systems Integration**: responsible for integrating Blockchain bi-directionally with external systems. Not part of Blockchain, but used with it.
Blockchain Components

- Application
- Membership
- Wallet
- Smart Contract
- Systems Management
- Ledger
- Events
- Consensus Network
- Systems Integration

Users Interests

Developers Interests

Time when topic is investigated in detail

Blockchain

f(abc);
How Applications use the Ledger

The key elements of a Blockchain application
Blockchain Applications and the Ledger

Blockchain developer develops 

**Smart Contract** implemented using chain code

Blockchain developer develops each 'put' or 'delete' invoke recorded

Application invokes

* get

* put, delete

each 'put' or 'delete' invoke recorded

Ledger

World/Ledger state

application

Blockchain

Accesses (Rest API)

* event emits

* Smart Contract implemented using chain code
Blockchain Applications

• Application
  – Focuses on Blockchain user business needs and experience
  – Calls smart contract for interactions with ledger state
  – Can access transaction ledger directly, if required
  – Can process events if required

• Smart Contract
  – Chain code encapsulates business logic. Choice of implementation language
  – Contract developer defines relevant interfaces (e.g. queryOwner, updateOwner ...)
  – Different interfaces access ledger state accordingly – consistent read and write provided
  – Each invocation of a smart contract is a “Blockchain transaction”

• Ledger
  – World/Ledger state holds current value of smart contract data
    • e.g. vehicleOwner=Daisy
  – Blockchain holds historic sequence of all chain code transactions
    • e.g. updateOwner(from=John, to=Anthony); updateOwner (from=Anthony, to=Daisy); etc
Operating Blockchain Networks

Configuring for a replicated ledger
Maintaining a consistent ledger

- Keep all peers up-to-date
- Fix any peers in error
- Quarantine all malicious nodes
Blockchain Operator configures, operates Smart Contract Application deployed to and executes on peer nodes

Consensus Network

Exact network structure depends on consensus mechanism. e.g. PBFT has leader, validating and non-validating peers

consensus messages flow between appropriate peers to ensure the Blockchain SC transactions are kept in order; world state is kept consistent through local transaction replay
How a PBFT Network Works (1/4) – Submission

Non-validating peer
Validating peer
Validating Leader

Transaction submitted to network

Transaction submitted to network
How a PBFT Network Works (2/4) – Ordering

Consensus network establishes order as

Non-validating peer
Validating peer
Validating Leader
How a PBFT Network Works (3/4) – Execution

Non-validating peer
Validating peer
Validating Leader

Consensus network establishes order as
How a PBFT Network Works (4/4) – Distribution

Non-validating peer
Validating peer
Validating Leader

Consensus network establishes order as
Blockchain Networks

• Blockchain Network
  – Comprises a connected set of peer nodes, each owning a copy of the ledger
  – Peers collaborate to maintain consistent replicated copies of the ledger
    • Different mechanisms for collaboration – so-called “consensus protocols”
  – Peers managed by key network participants

• Consensus Protocol Options
  – PBFT excellent first choice. NOOPs (No Operation) available for starter networks
  – Other protocols can be added (non-trivial!)

• PBFT Overview
  – Defines non-validating peers, validating peers, with 1-validating leader
  – Leader receives transactions from connected applications
  – Leader organizes and distributes transactions with validator network
    • Copes with erring and malicious validators at very low compute cost
  – Each v-peer executes transactions to bring local ledger copy up-to-date
  – Non-validating-peers’ ledgers maintained from connected validating-peer’s
  – Castro & Liskov’s paper ‘Practical Byzantine Fault Tolerance and Proactive Recovery’
Synchronization

If a node drops from the network or receives different transaction results or a new node joins, there are 3 options for getting back up to speed:

- **Get Deltas (Smallest)**
  - The node receives the deltas of the changes from previous transactions to update its world state
  - Smallest transfer size

- **Get Blocks**
  - The Node receives the sequential set of blocks it is missing

- **Get Snapshot**
  - The Node receives a snapshot of the World State
  - Largest transfer size
Consensus Considerations
Transaction and identity privacy
Non-Deterministic Number Errors

– If each smart contract calls out to external system to receive information or uses a non-deterministic variable, it will prevent consensus being achieved.

– All external data sources should return the same result for the same calls and variables must be deterministic, any non-deterministic values must be defines as part of the transaction
Duplicating Results

– If each smart contract stores calls out to alter information in an external system, it could lead to duplication
– The UUID of each transaction or the nonce can be used so only one instance of the call will be added to any external system
Permissioned Ledger Access
Transaction and identity privacy
Permissioned Ledger Access Today

Blockchain User A
- Uses Ecert and Tcert
- Invokes SC txn (signed with TkeyA, encrypted with Vkey)

Enrollment certificates (Ecerts) and Transaction certificates (Tcerts) can only be linked by CA and user

Consensus Network

Blockchain User B
- Accesses ledger

Application
- Uses /checkbld sc
- Membership Certificate Authority (stored in wallet)

Smart contract
- Deployed on every validating peer

Certification Authority
- Ecert
- Tcert

Accesses ledger
Transaction and Identity Privacy

• Transaction Certificates, Tcerts
  – Disposable certificates, typically used once, requested from Transaction CA
  – Tcert derived from long term identity - Enrollment Certificate, Ecert
  – Only Transaction CA can link Ecert and Tcert

• Permissioned Interactions
  – Consumer shares public Tcert to provider
  – Provider invokes chain code transaction as usual, but
    • Signs with provider’s private Tcert for authentication
    • Encrypts with provider and consumer Tcerts for subsequent access
  – Consumers can subsequently access ledger data using their private key

• Secure chain code
  – CC can also be signed and encrypted, to keep verify and secure contract details
  – Signing is by contract owner/author
  – Encryption ensures only validators can see and execute transaction chain code
Integrating with Existing Systems
Integrating with Existing Systems

1. Call out to existing systems to enrich smart contract logic
2. Call into Blockchain network from existing systems
3. Blockchain events
4. System events

Smart contract
Blockchain network
Integrating with Existing systems

• Blockchain is network System of Record

• Smart contracts can call out to existing systems
  – Query is most likely interaction for smart decisions
    • e.g. all payments made before asset transfer?
  – Transactions execute on every peer in the Blockchain network
    • Care over predictability… transaction must provide same outputs each time it executes

• Two-way exchange
  – Events from Blockchain network create actions in existing systems
  – Cumulative actions in existing systems result in Blockchain interaction

• Transformation between Blockchain and existing systems’ formats
  – GBO, ASBO is most likely approach
  – Standard approach will be for Gateway products to bridge these formats
  – Gateway connects to peer in Blockchain network and existing systems
Where are we now?
Current Architecture (v0.6)

- **SDK**: keys
- **membership**: ECA, TCA, TLS-CA
- **Consensus Ledger Events Chaincode**: state
- **peer**: enroll, transact
What We Have Learned

Better support for confidentiality
Scalable in number of participants and transaction throughput
Eliminate non-deterministic transactions
Enable pluggable data store
Ability to upgrade fabric and chaincode
Remove SPF and enable multiple providers of Membership Services
v1.0 Architecture

- **SDK**
  - keys
  - enroll
  - Submit Tx

- **membership**
  - No SPoF
  - No SPoT

- **peer**
  - Endorser
  - Committer
  - Ledger
  - Events
  - Chaincode
  - state

- **orderer**
  - Order TXs in a batch according to consensus

- Proposal
- Relay
- Batch
Hyperledger Fabric: Updated Roadmap and Releases

- Custom events
- Version indicator (log and cli)
- CC deploy SDK API

- Member services 1
- Enhance Ledger API
- Status codes & msg’s
- Event listener SDK

- Consensus 1
- Life-cycle SCC
- Error handling
- Tx simulation rw-set
- File-based datastore

- Enhance protocol
- Life-cycle SCC
- SDK specification
- SDK submitting TX

- ACL
- Kafka 1
- Multichannel 1
- HSM support PKCS11

- Auditability API
- State cache
- Upgrade chaincode
- Kafka 2
- Multichannel 2

- Sec code hardening
- Bug fixes

- Bug fixes

Skeleton drop - 11/11/16
Complete basic transaction flow of v1.0 architecture, from node.js SDK to commit on block.

Alpha drop of v1.0 - 12/17/16
Previous Alpha features move to Beta, additional features introduced as Alpha.

Beta driver of v1.0 – 1/31/17
Previous Alpha features move to Beta, additional features introduced as Alpha.

GA - 3/31/17
Next Generation Consensus
Details of the proposed HyperLedger next generation consensus protocol
Next Generation Consensus Protocol

• Validation role will be split into 2 independent roles:

  – **Endorsement**
    • Endorsing a transaction verifying that its content obeys a given smart contract. Endorsers “sign” the contract

  – **Consensus**
    • Consenting the inclusion of a verified transaction in the ledger. Consensus controls what goes in the ledger making sure that the ledger is consistent
Peers and roles

• Generic peer: observes and stores the ledger

• Endorsing peer: receives a transaction proposal for endorsement, responds granting or denying endorsement

• Consenting peer: approves the inclusion of transaction blocks into the ledger
1. The Client App proposes a transaction for **Smart Contract A** to the Endorsing peer $E_0$. Endorsement policy: “$E_0$, $E_1$ and $E_2$ must sign”. $E_3$ is not part of the policy.
2. Endorsing peer $E_0$ endorses a tx and (optionally) “anchors it” with respect to the ledger state version numbers. An “anchor” contains all data read and written by contract that are to be confirmed by other endorsers.
3. The client requests further endorsement from $E_1$ and $E_2$. The client may decide to suggest an anchor obtained from $E_0$ to $E_1$ and $E_2$. 
4. The Endorsing peers E₁ and E₂ send the endorsement to client.
5. Client formats the transaction and broadcasts it to the consenters for inclusion in the ledger
6. The consensus service delivers the next block in the ledger with the consented transaction.
Summary and Next Steps

For users
Summary and Next Steps

• We are at the beginning of the Blockchain journey!

• Apply shared ledgers and smart contracts to your Business Network

• Think about your participants, assets and business processes

• Spend time thinking about realistic business use cases

• Get some hands-on experience with the technology

• Do a First Project in 2016!

• IBM can help with your journey
Thank You!