On Multilateral Agreements And Multidomain Applications

Reggie Cushing
r.s.cushing@uva.nl
28/06/2021
Story so far...

- **Actors** = Containers
- Actors cryptographically addressed
- Multidomain communication through MQ using actor keys as topics.
- **Auditor** actors give permission to actors to carry out actions
- **Planner** actors encapsulate the notion of a workflow
  - Planners coordinate with auditors to execute workflow
Moving forward...multi-domain coordination

● A multidomain application is a workflow whereby the (data|control)flow crosses domain boundaries.
● Domain boundaries are controlled through rules/agreements derived from policies.
● A use case can be considered as having multiple facets.
  ○ The application functional components (functions)
  ○ The data assets
  ○ The coordination logic (controlflow)
● Controlflow is a program in itself that is owned by multiple domains.
● The challenge is:
  ○ How to execute a control program owned by multiple domains?
ArenA use-case multi-domain process model in
ArenA use-case multi-domain process model in

- Track, control, coordinate cross-border processes.
- Traditionally a static layer using API keys etc.
- In a marketplace we propose a programmable layer.
- We need to capture and coordinate these set of rules in a transparent and secure way.
- We propose state machines to keep track of the state of the border.
- Each party/domain updates the state machine thus signaling the other parties to take action.
Process model to infrastructure

Use-case BPMN

Coordination through smart contract/s

Graph interpreter

Functional containers

Multidomain infra
Functional containers

Process model to infrastructure

- **Generic** dataflow/petrinet executor running on a blockchain i.e. every peer is running the executor.
- Domains/actors are assigned a set of tokens.
- Actors define functions as a task with token input, token outputs and webhooks to interact with the outside world.
- So actors own tokens and tasks
- A task needs certain amount of tokens to fire
- Blockchain transactions copy tokens between actors.
- When a task has enough input tokens it will fire which in turn generates blockchain events.
- Containers monitor the ledger to trigger a process inside a container (the task).
- The container will make blockchain transactions to signal the task is completed and move the state machine.
Process model to infrastructure

- a petrinet that regulates how multi parties collaborate.
- the rationel is that a party can only perform a certain task given a certain context.
- Context is a set of multi domain preconditions that have to met.
- The movement of tokens within the graph changes the context.
- Tasks running outside the their agreed context are deemed as illegal.
- The approach should be generic enough that it can be applied to event driven scenarios.
- Petrinets have strong mathematical foundations.
- Can be analysed for behaviour and structure properties e.g. reachability, boundedness liveness, reversibility, coverability...
Process model to infrastructure

Coordination through smart contract/s

Use-case BPMN

Graph interpreter

Functional containers

Multidomain infra

chaincode container A
smart contract 1
smart contract 2
smart contract 3

chaincode container B
smart contract 3
smart contract 4
Beneath the blockchain buzz words; a computer scientist’s view

● Is a distributed database.
● Instead of storing the DB data, store the transactions the made the data.
● Data ‘asset|token’ is cryptographically signed data struct by users ‘owners’.
● Changing owner’s signature of data is a ‘transaction’.
● Users have pki keys. ‘accounts|wallets’.
● Use a linked list to store the transactions ‘blockchain’.
● Reference(hash) the previous list’s recordset ‘block’ in the new block.
● Multiple nodes need to agree on recordset order ‘consensus’.
● Multiple nodes can rebuild the data from the linked list.
● Since multiple nodes can do something then they can also run scripts ‘smart contracts’.
● End result is a distributed network that can run deterministic scripts to manipulate a shared linked list where records are owned by different users.
Blockchain primitives

- **Participants**
  - Users with an x509 cert given by a CA peer on the network.

- **Assets**
  - User defined data structs owned by a participant.
  - Cryptographically signed data structs.

- **Transactions**
  - Move assets between participants

- **Chaincode(smart contracts)**
  - Javascript/go/java programs to create programs with these primitives.
  - The chaincode runs on all/multiple peers of the network
  - Transactions are recorded in the DB(Ledger)

- **The challenge:**
  - How to map the controlflow program to a chaincode.
  - Make it generic.
  - How to interface actors to the chaincode (we want actors to affect state changes in the controlflow)
Petrinet to blockchain mapping

- A place receives is a placeholder for tokens.
- It is owned by a domain.
- Can be represented as an Asset.
Petrinet to blockchain mapping

- Tokens are passed between places.
- They are owned by domains.
- They are represented as assets.
- Tokens change ownership when moved between places.
- As is with web tokens, tokens also represent authorization. A function can only execute if it has to correct tokens from the different domains.
Petrinet to blockchain mapping

- Transitions are what move tokens between places.
- They are represented as an asset.
- They are owned by domains.
- They map to container functions.
- A transition fire implies a container function execution.
Arrows show the control flow of the network.
They indicate the required input tokens for a transition and the number of output tokens.
A transition (container function) fires when the required input tokens are ready.
Petrinet life cycle

Development & Analyses

Develop the petrinets as ‘smart contracts’. Analyse petrinets. We can only deploy once to a blockchain.

Authorization & Deployment

Deployment needs authorization from multiple peers on the network. This will need an audit layer to authorize deployments.

Activate

Once deployed it is in a start state. Moving from the start state activates the petrinet.

Terminate

A petrinet can terminate it can not move to any other state.