THE QUEST TOWARDS COORDINATION

FROM DISTRIBUTED TO SOCIO-TECHNICAL SYSTEMS

Stefano Mariani

Università di Modena e Reggio Emilia

OUTLINE

1. Why coordination "on the block"(chain)?
2. How? Linda as benchmark
3. Conceptual and technical gaps
4. A prototype of pro-active Smart Contracts

WHY?

The Blockchain provides highly desirable properties from a coordination standpoint:

- partial / total ordering of events
- adjustable consistency of replicated data
- accountability of actions
- identity management, non-repudiation, authentication
- (Byzantine) adjustable fault-tolerance

Name one which is NOT relevant to the Internet of Things vision, or distributed systems in general...

WHY NOT :)

- can the Blockchain **mediate interaction**?
  - amongst off-chain entities
  - amongst on-chain entities
  - amongst both
  - amongst any combination thereof
- can the Blockchain act as a **coordination medium**?
  - fixed coordination laws?
  - application-level coordination laws?

HOW?

Let's use the archetypal Linda model as a benchmark!

- can the Blockchain implement Linda **core abstractions**?
  - tuples
  - templates
  - matching
  - tuple space
  - primitives
- can the Blockchain preserve Linda **essential properties**?
  - generative communication
  - associative access
  - suspensive semantics
  - reference, time, space uncoupling

WHICH BLOCKCHAIN?

"The" Blockchain is a blurred concept

Depending on

- openness (permission-ed/less)
- access control (w.r.t. assets)
- architecture (clients, miners, ...)
- state model (how data is tracked)
- asset (money, ownership, ...)
- programmability (smart contracts, chaincode, ...)
- meta-primitives (deploy, invoke, ...)
- primitives (send, log, ...)
- consensus (degree of consistency / fault-tolerance)
- finitness (finite computations)

radically different blockchains may be defined, with specific functional / non-functional properties

ETHEREUM

- Permission**less** blockchain featuring Smart Contracts
- No access control, nodes are either clients or miners
- Any byte string can be the asset to keep track of
- Consensus via **Proof-of-Work** (resistant to Byzantine failures)
- **Native cryptocurrency** (ETH) and Gas guarantee finite computations

ETHERLINDA

- Mapping:
  - Smart Contracts implement the tuple space API
  - strings are tuples, regular expressions templates
  - generative communication via blocks
  - associative access via regex pattern matching
  - suspensive semantics via Ethereum **events mechanism**
- Calling primitives costs money
  - economic model depends on implementation
  - e.g. advertising model vs. market model
- e.g. Reference, time, space uncoupling preserved

HYPERLEDGER FABRIC

- Permission**ed** blockchain featuring Chaincodes
- **Membership service** and channels for access control, nodes can be endorserers, orderers, peers
- Any serialisable key-value pair can be the asset to keep track of
- Pluggable consensus (default: PBFT)
- Execute-Order-Validate architecture guarantees finite computations

HYPERLINDA

- Mapping:
  - Chaincodes implement the tuple space API
  - strings are tuples, regular expressions templates
  - generative communication via blocks
  - associative access via regex pattern matching
  - suspensive semantics via HyperLedger **events mechanism**
- Fine-grained access control, adjustable consensus
- Dirty implementation (e.g. rd)
- Reference, time, space uncoupling preserved

R3 CORDA

- Permission**ed** blockchain featuring Flows and Contracts
- **Doorman service** for access control, nodes can be notaries or peers
- Any serialisable object can be the asset to keep track of
- Pluggable consensus
- No guarantees for finite computations

CORLINDA

- Mapping:
  - Vaults implement the tuple space API
  - strings are tuples, regular expressions templates
  - generative communication via blocks
  - associative access via regex pattern matching
  - suspensive semantics via Flows
- Reference uncoupling lost
- Subjective view of tuple space
- No first-class "program"

TAKEAWAY

There is no *"one blockchain to rule them all"*

- Small technical details may have huge impact on expressiveness
- Unfortunately, they are often abstracted away when modeling blockchain to prove properties...

Blockchain-based coordination seems nevertheless possible :)

only amongst off-chain nodes, open to application-level laws

MIND THE GAP

- No or limited concurrency
- Full uncoupling not always possible
- Suspensive semantics often needs clever / dirty implementation hacks
- No coordination amongst on-chain programs (e.g. Smart Contracts)

Most issues stem from Smart Contracts pure **reactiveness**

they cannot do anything if not as a consequence to clients (e.g. users) input

FILL THE GAP

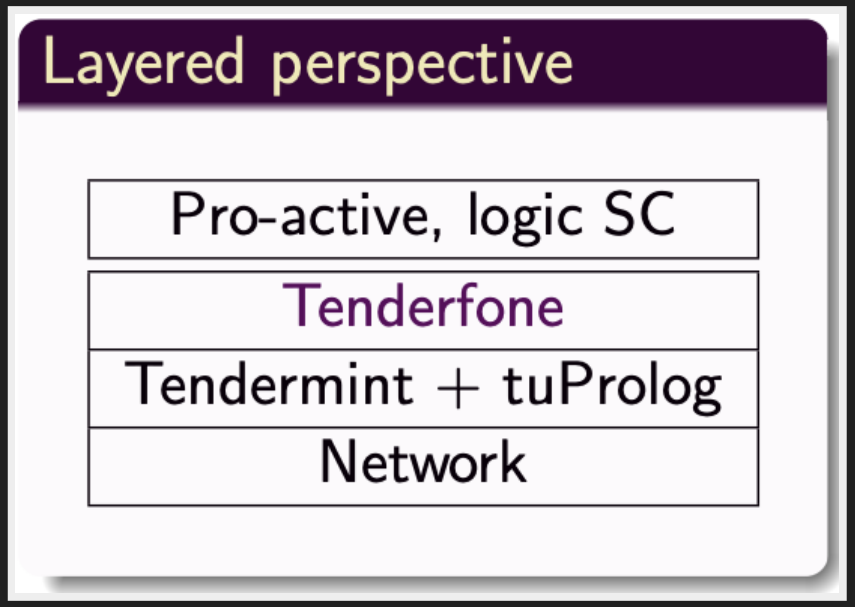
*Coordination needs **pro-active** Smart Contracts*

- own flow of control
- asynchronous interaction means
- time awareness handy, too

OUR PROTOTYPE

- Minimal Smart Contracts language
  - .init, .send, .when, .every
- Suitable Smart Contracts interpreter
- Suitable blockchain as execution infrastructure
  - Tendermint
  - Byzantine fault-tolerant state machine replication infrastructure (ledger + consensus + dev API)

ARCHITECTURE



- tuProlog interprets the Tenderfone language
- Tendermint takes care of consensus amongst replicas
- Tenderfone exposes to developers a declarative API akin to Prolog

.init -> own control flow

.send -> asynchronous interaction

.when, .every -> time-awareness

THANKS
FOR YOUR ATTENTION

QUESTIONS?

Stefano Mariani

Università di Modena e Reggio Emilia