Comparative Analysis of Blockchain Technologies under a Coordination Perspective

Giovanni Ciatto[1] Michael Bosello[1] Stefano Mariani[2]
Andrea Omicini[1]

giovanni.ciatto@unibo.it michael.bosello@studio.unibo.it
stefano.mariani@unimore.it andrea.omicini@unibo.it

[1]Dipartimento di Informatica – Scienza e Ingegneria—Università di Bologna

[2]Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia

2nd International Workshop on
BlockChain Technologies 4 Multi-Agent Systems
Ávila, Spain – June 26, 2019

Outline

1. What & Why
2. Ok, How?
3. Comparison
4. Cool, Then?

Next in Line...

Premises

- a lot of BCT have been developed in the recent years
- some are simply aimed at improving crypto-currencies
- other are conceived to be general and track custom assets
  - through some sort of programmability

  e.g. by means of smart contracts

Some foundational question arise

- can BCT be used to mediate the interaction of { off-chain entities? / on-chain entities? }
- can it be used for coordination?

Premises

- a lot of BCT have been developed in the recent years
- some are simply aimed at improving crypto-currencies
- other are conceived to be general and track custom assets
  - through some sort of programmability

  e.g. by means of smart contracts

Some foundational question arise

- can BCT be used to mediate the interaction of { off-chain entities? / on-chain entities? }
- can it be used for coordination?

What is *Coordination*?

Within MAS

"A framework in which the interaction of active and independent entities called agents can be expressed" [Cia96]

active entities i.e., proactive, computationally autonomous

interaction *dependencies* among agents activities

- e.g. communication, cooperation, competition, etc.

expressed i.e., explicitly represented, or observed

Takeaway

Coordination studies interaction at the fundamental level

Wait: but *why* Coordination & BCT?

BCT provide highly-desirable properties

- *ordering* of events
- *consistency* among replicated data
- *accountability* of actions
- *identity* management
- (byzantine) *fault tolerance*

Especially in MAS!

Accountable, trusted communications, and consistency of information are long-standing issues for open MAS [RHJ04, HM15]

Next in Line. . .

Our test-bed – Tuple-based coordination in LINDA I

LINDA at a glance

Archetypal tuple-based coordination model [Gel85]

4 main elements

- tuples — structured data chunks
- templates — pattern language for tuples
- tuple spaces — blackboards for tuples
- primitives — put (OUT), or access, i.e. read (RD), or consume (IN)

3 main features

- generative — tuples independent from agents
- associative — tuples accessed through templates; no addr. & no ref.
- suspensive — access attempts suspended if no tuple is available

Our test-bed – Tuple-based coordination in LINDA II

Useful in MAS?

TL;DR Yes! [OZKT01]

LINDA over BCT

- How can primitives be implemented on BCT?
- Can we reproduce suspendive semantics?
- Does it work for { off-chain agents? / on-chain agents? }
- Are there differences among BCT w.r.t. interaction?

Comparison Dimensions for BCT

- openness — whether they are permissioned or permissionless
- access control — degrees of freedom in controlling access to assets
- architecture — main roles for nodes composing the BCT network
- state model — how (i.e., in which form) the BCT tracks information
- asset — which sort of assets may be manipulated by users
- programmability — how the programmable elements are called
- meta-primitives — how programmable elements can be manipulated
- primitives — operations letting programmable elements interact
- consensus — how the BCT supports consistency
- termination — how BCT prevent infinite computations

Next in Line. . .

1. What & Why
   - Coordination
2. Ok, How?
   - LINDA as a test-bed
3. Comparison
   - Ethereum
   - HyperLedger Fabric (HLF)
   - Corda
4. Cool, Then?

Ethereum at a Glance

openness → permissionless

access control → none

architecture → light nodes (clients) + full nodes (miners)

state model → accounts

asset → byte strings

programmability → smart contracts (SC)

meta-primitives → Deploy, Call

primitives → Call (inter-SC), Raise Event (SC-to-client)

consensus → Proof of Work (all miners)

termination → gas

Linda over Ethereum

Details

! more in [CMO18]

tuple spaces ↦ smart contracts

coordinated agents ↦ clients

primitives ↦ method calls through the Call meta-primitive

suspensive semantics ↦ listening for events

Peculiarities

- economy of coordination [CMO18]

Shortcomings

- smart contracts cannot be coordinated agents
- no concurrency for primitives

HyperLedger Fabric (HLF) at a glance

- openness → permissioned
- access control → Membership service, Channels, Endorsement policies
- architecture → peers + endorsement peers + ordering nodes
- state model → Key-Version-Value (KVV) triplets
- asset → KVV triplets
- programmability → chaincodes (C)
- meta-primitives → Deploy, Invoke, Query
- primitives → Invoke (inter-C), SetEvent (C-to-peer)
- consensus → pluggable (default: PBFT [CL02])
- termination → execute-order-validate architecture [ABB+18]

LINDA over HLF

Details

tuple spaces $\mapsto$ chaincodes

coordinated agents $\mapsto$ peers

primitives $\mapsto$ method calls through the Invoke meta-primitive

suspensive semantics $\mapsto$ listening for events

Peculiarities

- finer control w.r.t. replication, consensus, tuple space access
- some degree of concurrency w.r.t. primitives

Shortcomings

- smart contracts cannot be coordinated agents
- workaround needed for completing multiple primitives at once

Corda at a Glance

openness → permissioned
access control → Doorman service + Flows
architecture → nodes + notary services (notaries)
state model → unspent transaction output (UTXO) + Vaults
asset → states
programmability → contracts
meta-primitives → none (contracts deployed by admins)
primitives → none (validation only)
consensus → pluggable (custom algorithm)
termination → not a problem (contracts are trusted)

LINDA over Corda

Details

tuple spaces $\mapsto$ vaults

coordinated agents $\mapsto$ nodes

primitives $\mapsto$ state creation, or consumption

suspensive semantics $\mapsto$ flows

Peculiarities

- subjective visibility of tuples in tuple spaces
- some degree of concurrency w.r.t. primitives

Shortcomings

! tight coupling among coordinated agents

Next in Line...

Takeaways

The good

- BCT can in general be used for coordination off-chain
- permissioned BCT allow for a finer control w.r.t. tuple visibility to off-chain entities

What can be improved

- on-chain computational entities may benefit coordination as well
  - because they can be conceived as agents [CMMO19]
  - even if this is currently poorly understood by the community

The bad

- LINDA-based coordination is not fully supported by Corda

Conclusion

Summarising, we

- compared three programmable BCT, namely Ethereum, HLF, & Corda
- w.r.t. the coordination perspective, by implementing LINDA on them
- thus testing capabilities, peculiarities, and limits of BCT
- and showing that smart contracts fall short w.r.t. mutual interaction

In the future, we plan to

- extend our comparison to more technologies
- propose a novel programmable BCT overcoming current limitations

Conclusion

Summarising, we

- compared three programmable BCT, namely Ethereum, HLF, & Corda
- w.r.t. the coordination perspective, by implementing LINDA on them
- thus testing capabilities, peculiarities, and limits of BCT
- and showing that smart contracts fall short w.r.t. mutual interaction

In the future, we plan to

- extend our comparison to more technologies
- propose a novel programmable BCT overcoming current limitations

Comparative Analysis of Blockchain Technologies under a Coordination Perspective

Giovanni Ciatto[1] Michael Bosello[1] Stefano Mariani[2]
Andrea Omicini[1]

giovanni.ciatto@unibo.it michael.bosello@studio.unibo.it
stefano.mariani@unimore.it andrea.omicini@unibo.it

[1]Dipartimento di Informatica – Scienza e Ingegneria—Università di Bologna

[2]Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia

2nd International Workshop on
BlockChain Technologies 4 Multi-Agent Systems
Ávila, Spain – June 26, 2019

References I

Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick.
Hyperledger fabric: A distributed operating system for permissioned blockchains.
In *13th EuroSys Conference (EuroSys '18)*, New York, NY, USA, 2018. ACM.

Paolo Ciancarini.
Coordination models and languages as software integrators.
ACM Computing Surveys, 28(2):300–302, June 1996.

Miguel Castro and Barbara Liskov.
Practical byzantine fault tolerance and proactive recovery.
ACM Transactions on Computer Systems, 20(4):398–461, 2002.

Giovanni Ciatto, Alfredo Maffi, Stefano Mariani, and Andrea Omicini.
Towards agent-oriented blockchains: Autonomous smart contracts.
In *PAAMS 2019*, 2019.

References II

Giovanni Ciatto, Stefano Mariani, and Andrea Omicini.
Blockchain for trustworthy coordination: A first study with Linda and Ethereum.
In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 696–703, December 2018.

David Gelernter.
Generative communication in Linda.
ACM Transactions on Programming Languages and Systems, 7(1):80–112, January 1985.

Yaqin Hedin and Esmiralda Moradian.
Security in multi-agent systems.
Procedia Computer Science, 60:1604–1612, 2015.

Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors.
Coordination of Internet Agents: Models, Technologies, and Applications.
Springer, March 2001.

Sarvapali D. Ramchurn, Dong Huynh, and Nicholas R. Jennings.
Trust in multi-agent systems.
The Knowledge Engineering Review, 19(1):1–25, 2004.