CAPER: A Cross-Application Permissioned Blockchain

Mohammad Javad Amiri, Divyakant Agrawal, Amr El Abbadi

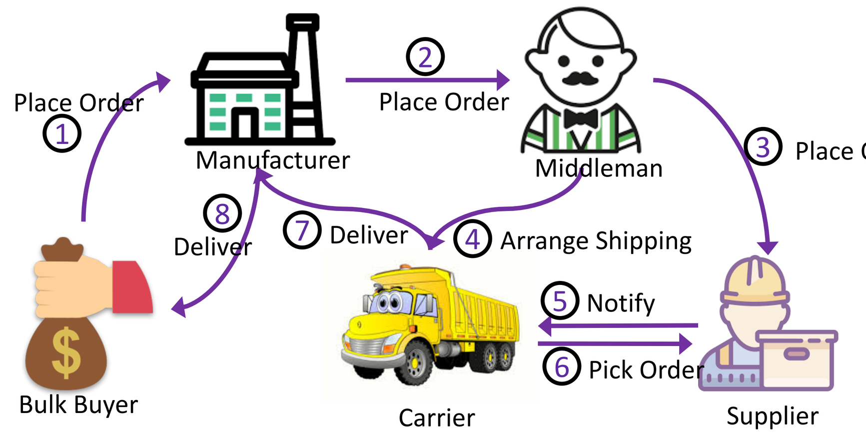
University of California Santa Barbara

ABSTRACT

- Distributed applications collaborate with each other following *service level agreements* (SLAs) to provide different services.
- While collaboration between applications, e.g., *cross-application transactions*, should be visible to all applications, the internal data of applications, e.g., *internal transactions*, might be confidential.
- CAPER: a permissioned blockchain system to support both internal and cross-application transactions of collaborating applications.
- Each application orders and executes its internal transactions locally while cross-application transactions are public and visible to every node.
- The blockchain ledger is formed as a *directed acyclic graph*: each application maintains its own view of the ledger including its internal and all cross-application transactions.
- We introduce three consensus protocols to globally order cross-application transactions among applications with different internal consensus protocols.

INTRODUCTION

- Blockchain is a distributed data structure for recording transactions maintained by nodes *without a central authority*. In a blockchain, nodes agree on their shared states across a large network of *untrusted* participants.
- A permissioned blockchain consists of a set of *known*, *identified* nodes that might *not fully trust* each other.
- Distributed applications collaborate with each other to provide different services. Collaborations are defined in service level agreements (SLAs) which are agreed upon by all involved applications. SLAs can be written as self executing computer programs, called *smart contracts*.
- The collaboration is realized using cross-application transactions that are visible to all applications.



Supporting Collaborative Workflow using Blockchain: Existing Solutions

1. Deploy all applications on the same blockchain system
Smart contracts might be confidential, however, transactions data and blockchain ledger are replicated on every application (single-channel Fabric) **Confidentiality issue**

2. Deploy each application on a separate blockchain system
Use another blockchain system for the cross-application transactions **Data Integrity issue**

3. Deploy each application on a separate blockchain system
Use (atomic) cross-chain operation **Performance issue**

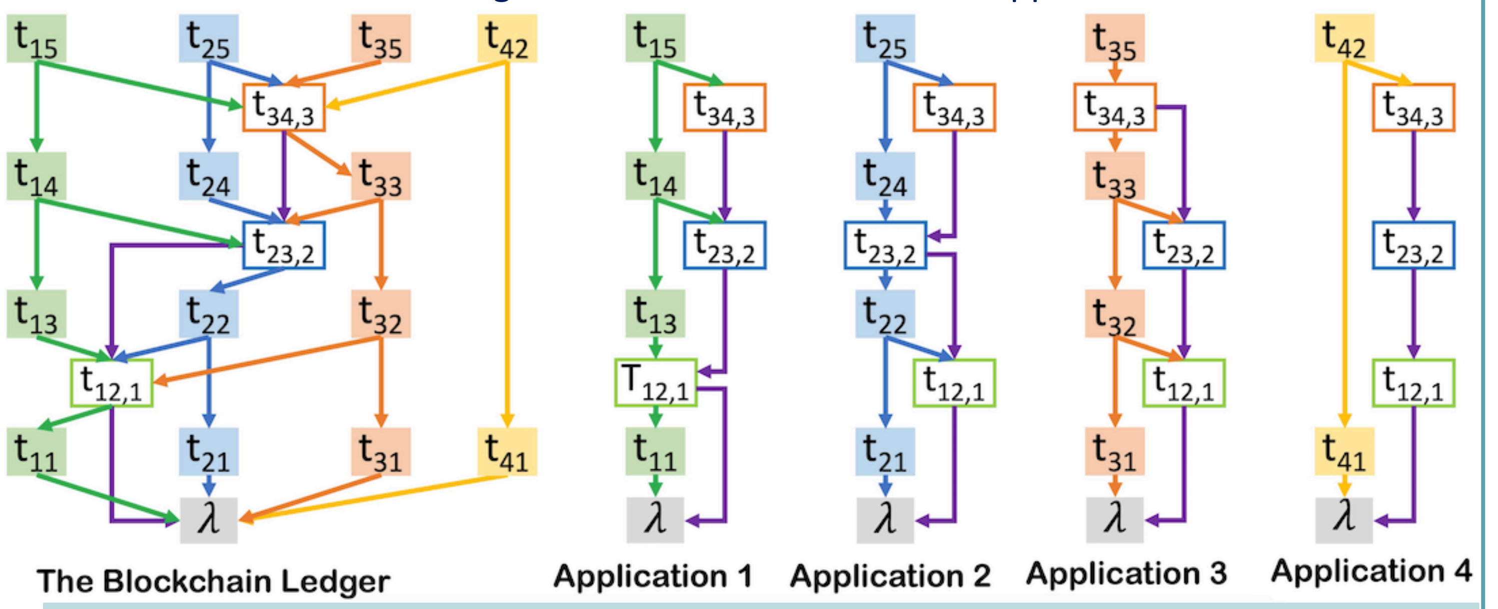
THE CAPER MODEL AND ARCHITECTURE

- CAPER consists of a set of collaborating distributed applications.
- Each application maintains two sets of *private* and *public* records.
- The private records of an application are accessible *only* to the application.
- The public records are *replicated on all* applications.
- CAPER supports both internal and cross-application transactions.
- Internal transactions are performed within an application following the application logic.
- Internal transactions read and write private records, however they can only read the public records. Cross-chain transactions read and write only the public records.
- Cross-application transactions follow SLAs among applications.

Blockchain Ledger

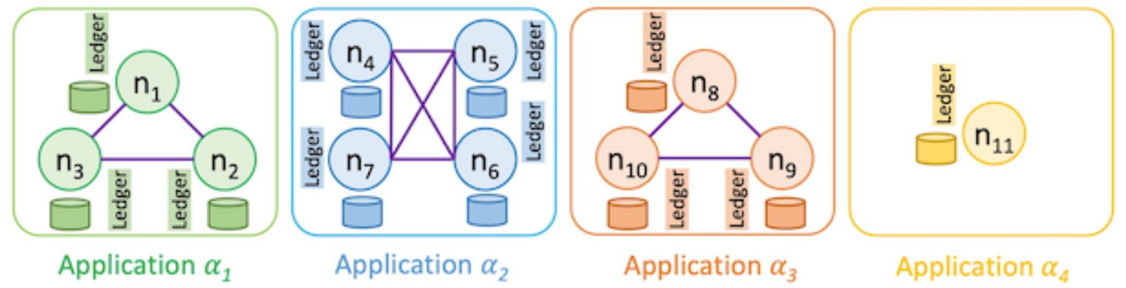
- In CAPER, each block consists of a *single* transaction.
- The blockchain ledger is formed as a *directed acyclic graph (DAG)*.
- The blockchain ledger has three properties:

1. There is a *total order* between all transactions (internal as well as cross-application) that are *initiated by an application*.
2. There is a *total order* between *cross-application transactions.*
3. An internal transaction might include the hash of a cross-application transaction.



Blockchain Architecture

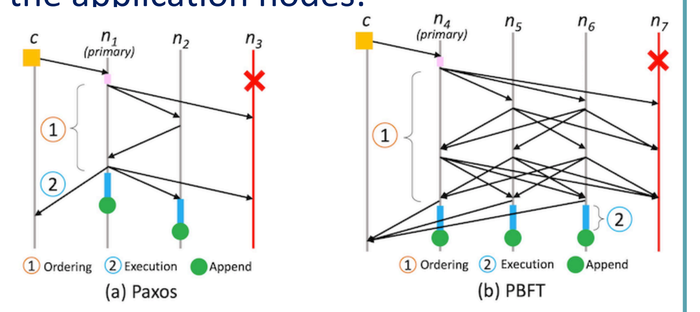
- Each application maintains: (1) its view of the blockchain ledger, (2) a private smart contract, (3) a public smart contract, and (4) the datastore.
- Nodes in CAPER might crash, behave maliciously, or be reliable.
- Applications do not trust each other: we model application failures as Byzantine failures.
- Two levels of behavior are defined in the system: *node level* and *application level.*
- We assume that at most *one-third* of the applications might be malicious.



CONSENSUS IN CAPER

Local Consensus

- Pluggable and depends on the failure model of the application nodes:
- Crash-only failure
  Paxos
- Byzantine failure
  PBFT

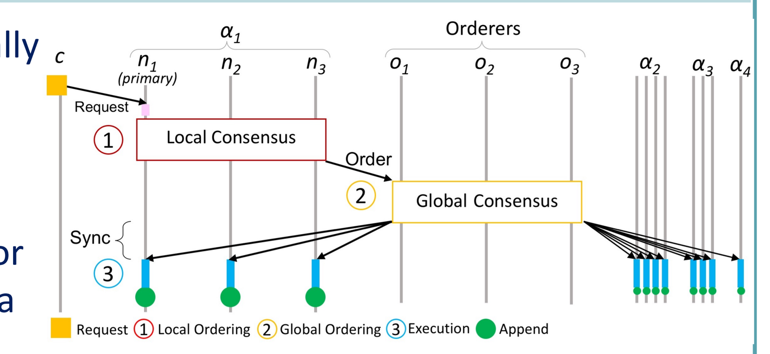


Global Consensus

- Needs the participation of all of the applications. Three protocols are introduced.

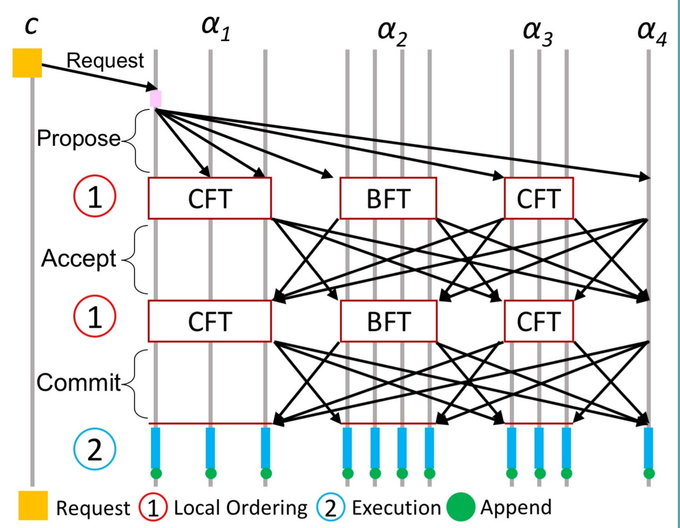
Global Consensus using a separate set of orderers

- A disjoint set of nodes, *orderers*, globally orders cross-application transactions.
- Cross-application transactions are first ordered *locally* and then ordered *globally.*
  - To ensure that the agents of the initiator application agree on the local order of a transaction



Hierarchical Global Consensus

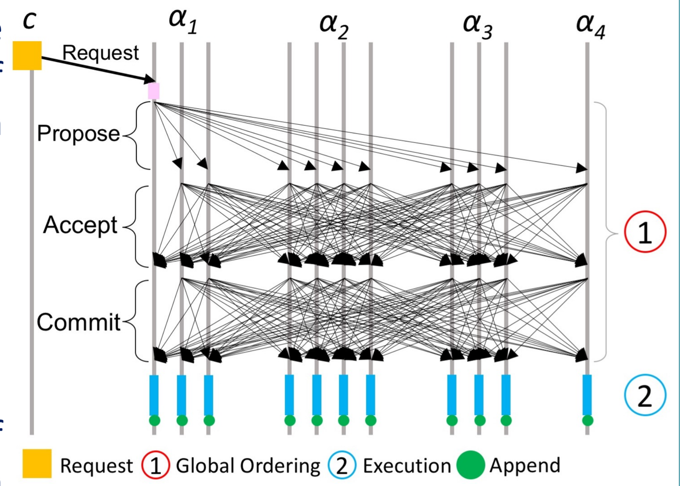
- Using orderers comes with an extra cost of adding orderers to the system.
- CAPER distinguishes between trust at the node level and trust at the application level
- In each phase of the global consensus, every application runs its local consensus protocol to internally decide on the application vote.
- CAPER ensures that the initiator application agrees with the ordering.



One-Level Global Consensus

- Hierarchical consensus requires an expensive *two-level* consensus protocol: Each step of global consensus needs local consensus within each application.
- One-Level Consensus: all agents of all applications talk to each other.
- Each phase needs *local-majority* of *two-thirds* of the applications

Local-majority: the required number of matching messages from the agents of an application, e.g., $f+1$ (Paxos) or $2f+1$ (PBFT)

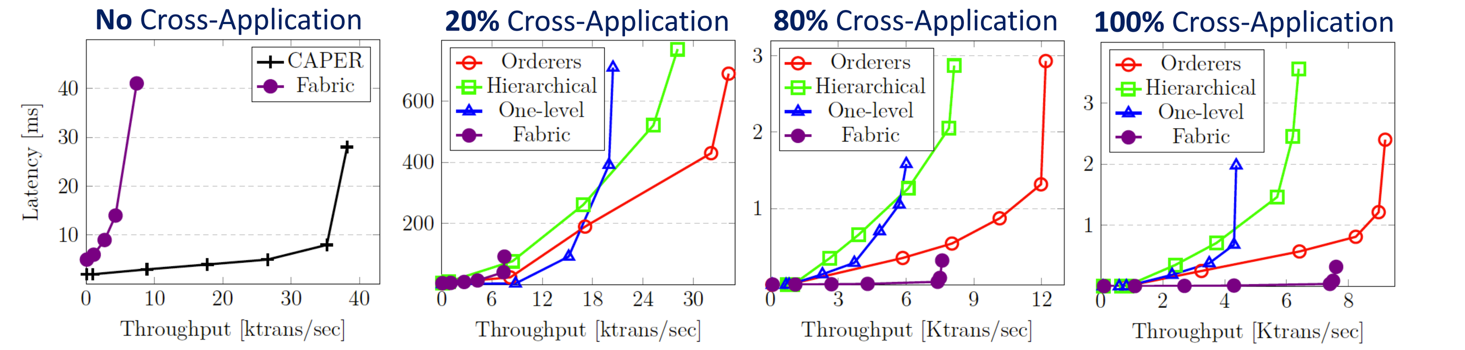


EXPERIMENTAL EVALUATION

Experimental Settings:
Accounting applications, each application has three agents and uses Paxos (f=1), the load is equally distributed among the applications.

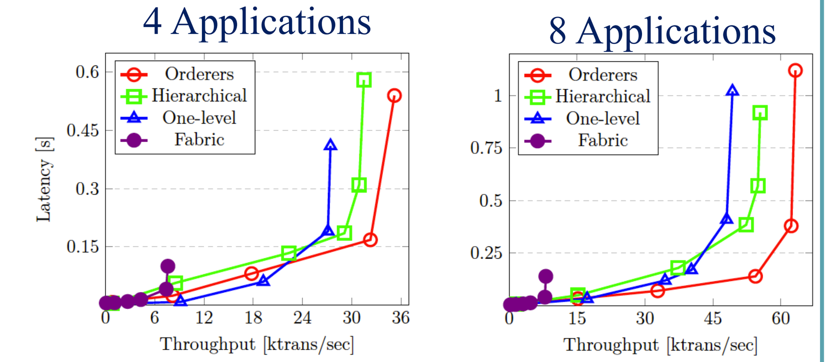
Workloads with Cross-Application Transactions (4 Applications)

- For *lightly loaded* applications *one-level* consensus shows better performance.
- Using a set of *orderers* is more beneficial for *heavily loaded* applications.
- In the absence of extra resources for orderers, the hierarchical approach can provide better performance in heavily loaded applications.
- With *high percentage* of cross-application transactions *Fabric* has less latency.
- The performance of Fabric remains unchanged in different workloads.



Performance with Multiple Applications (10% cross-application transactions)

- The overall throughput of CAPER improves near-linearly.
- One-level and Hierarchical: higher latency for the same throughput
- The performance of Fabric does not improve significantly.



CONCLUSION

- We proposed CAPER, a permissioned blockchain system that supports both internal and cross-application transactions of collaborating distributed applications.
- CAPER targets both performance and confidentiality aspects of blockchain systems.
- To achieve better performance, CAPER orders and executes internal transactions of different applications simultaneously.
- To achieve confidentiality, the blockchain ledger is *not maintained* by any node and each application maintains its own local view of the ledger.
- CAPER distinguishes between trust at the node level and application level and allows an application to behave maliciously for its benefit while its nodes are non-malicious.
- CAPER introduces 3 consensus protocols to globally order cross-application transactions:

(1) using a separate set of orderers, (2) hierarchical consensus, and (3) one-level consensus.

Acknowledgement

This work is funded by NSF grants CNS-1703560 and CNS-1815733.