On the Atomicity of Crosschain Transactions

Ermyas Abebe
Atomicity in distributed transactions

• A distributed database transaction defines a unit of state change across two or more *data stores* (*shards, databases, partitions*)

• Examples:
  • Updating account balances across shards (transfer Alice -> Bob)
  • Flight, Hotel, Car (long lived transactions, spanning multiple DBs)

• Two important aspects:
  • Isolation: concurrency control (*SS2PL*)
  • Atomicity (*2PC, 3PC*)

• Atomicity is a property that guarantees that either all sub-transactions succeed, or neither does (all-or-nothing)
Two-phase Commit Protocol

Client  Coordinator  Database A  Database B
commit

prepare
vote: yes/no
commit/abort
ack

prepare
vote: yes/no
commit/abort
ack

Limitations
- Blocking Problem: Coordinators could crash, tying resources and blocking progress
- Database nodes could fail causing related problems
- Various mitigation strategies to improve fault tolerance
- Might require manual intervention
- Increases latency, Limits throughput and scalability

In practice:
- Within single administrative domain and often employed across homogenous systems
- For long-lived distributed transactions, approaches such as Saga patterns (with compensating transactions) are often employed in practice
Characterising Crosschain Communication

Interaction Patterns

A → B: state in B depends on state change in A

Example:
- Read balances of all token holders in one network and confer rights on another (DAOs...)

A → B iff B → A: state change in A and B needs to be synchronised. State in A changes if and only if corresponding state changes in B, and vice versa.

Example: Trust-less asset exchange between two or more parties.

Modes

- Asset Exchange
- Asset Transfer
- General Purpose crosschain Transactions
Characterising Crosschain Communication

**Messaging Patterns**
- Messages not directly exchanged between networks
- Coordinated by parties

**Verification**
- Locally verified state

- State change events Relayed across networks
- Relayed or Attested by a set of external parties/validators

- Natively: Light-client headers
- External: A set of entities attest messages across chains (m-of-n scheme, proof-of-authority, stake … ), optimistic/pessimistic
Atomic Crosschain transaction

• Atomicity: ensure that either all sub-transactions in a crosschain transaction occur or none does

• Challenges
  • More failure scenarios: Byzantine faults
  • Finality guarantees (probabilistic, deterministic - instant or eventual)
  • Who coordinates transactions (Transaction parties, bridge intermediaries)?
  • Incentive alignment and stability under dynamic environments (asset volatility, changing gas prices etc.)
  • Transaction costs of multi-step coordination
  • Bridge hacks (de-pegging of assets…)

• Atomic Commit Protocols:
  • Properties: Safety, Liveness
  • Desired Properties: Fairness, Trust-lessness, Throughput, Asynchrony, Privacy
Contract Patterns for Asset Exchange

Hashed Time-Lock Contracts (HTLC) - Crosschain Atomic Swaps

1. Alice creates contract with pre-image of secret known only to Alice
2. Bob creates contract with pre-image of secret known only to Alice
3. Bob and Alice audit each other’s contracts
4. Alice locks tokens in contract for Bob to claim
5. Bob locks tokens in contract for Alice to claim
6. Alice claims tokens by submitting secret to contract
7. Bob inspects secret now shared on ledger to claim tokens

Fund locking
- To initialise an asset exchange, it is common for one or both parties to first lock up funds with a fund-withholding party on his or her own blockchain.
- Temporary fund locking ensures the locked fund cannot be used for other purposes while the exchange is being executed.
- This scheme is often used with a specified timeout to provide flexibility for reclaiming locked funds if the exchange does not take place.

Execution
- In general, the execution requires a pair of transactions to occur on both blockchains.
- The execution of the exchange can be carried out by users themselves, or through other trusted third parties.

Refund
- For protocols that are initialised with a temporary fund-locking, the locked funds can usually be reclaimed by the initial owner after a specified timeout.
Contract Patterns for Asset Exchange

Hashed Time-Lock Contracts (HTLC) - Crosschain Atomic Swaps

- Assumption:
  - All parties have to be online. One party failing could break atomicity.
  - Hash function compatibility

- Problems:
  - Free-option problem and gifting
  - Latency
  - Parties have to be online
  - Not very generalisable beyond asset exchange scenarios

- Numerous works have improved different aspects of the protocol:
  - Addressing the free option-problem (pricing, fractionalising...)
  - Reducing synchrony requirements
  - Extension to multi-party, multi-asset exchanges
  - Extensions to support networks that do not support hashlock or timelock primitives
  - Improve privacy
  - In practice, difficult to use at scale
Contract Patterns for Asset Transfer

1. Alice transfers tokens to a designated bridge contract

   Address A

   Network A

   Structural assurance depends on underlying networks and the bridge validators/operators

2. Corresponding about of wrapped/synthetic assets are minted and transferred to Alice’s address

   Address A

   Network B

   Relayers/Attestors

   • A set of parties relay/attest to destination network
   • Multi-Sig or Threshold signatures (PoS, or PoA)
   • Optimistic schemes

   Alice
Asset Exchanges in Practice (Liquidity Pools)

- Single-asset liquidity pool maintained on each chain
- Asset exchange semantics built on top of asset transfer mechanics
- User seeking to acquire token X on network A for corresponding token Y on network B
  1. Locks X on network A
  2. Message of the locked asset is relayed to destination
  3. User is either a) issued synthetic token which is a claim to an amount in the pool b) directly given the corresponding tokens

- Various failure scenarios
  - Relaying crosschain messages could fail across chains, or delayed (crash, gas spikes etc.)
  - Liquidity in the destination chain might no longer be available
  - User funds could be locked for extended periods, waiting for failure scenario to resolve
  - Costly and time consuming for user to refund transactions
  - Manual admin interventions
Failure Scenarios of non-atomic models

• State updated in one source chain but not in destination

• Failure could occur for a number of reasons
  • Failure of intermediary bridge validators (relayers/attestors)
  • Failure of destination transaction for various reasons (e.g. not enough liquidity in pool)
  • Change in economic costs of completing transaction (e.g. gas price spikes in destination, asset price volatility)

• Atomicity in crosschain in practice:
  • Protocol Guaranteed Atomicity (increase security and resilience, reduce trust assumptions)
  • Externally coordinated “atomicity” (exposes users to various failure scenarios, which are often remediated through trusted operators/admins…)
General Purpose Atomic Transactions

- Address more complex crosschain application interaction scenarios
- Crosschain transactions spanning multiple chains, performing general function calls
- The need for atomicity in this context grows as failure scenarios leading to inconsistent crosschain states could be difficult to remediate
- Challenges of protocol enforced atomicity
  - How do we ensure atomicity of arbitrary state changes?
  - What type of isolation guarantees need to be offered?
  - Who coordinates transactions? Can they perform all legs of a transaction and are they incentivised to do so?
  - How do we reason about incentive alignments across all parties?
  - What application-context independent assumptions can be made?
  - What are the costs?
GPACT Protocol

• The General Purpose Atomic Crosschain Transaction (GPACT) protocol is a call tree commitment scheme that provides atomic crosschain function calls across an arbitrarily deep call tree*

• Example: Orchestrate global trade workflow spanning different networks (trade finance, trade logistics, provenance, payments identity/KYC networks etc.)

* Peter’s talk covers GPACT and incentivisation in crosschain communication in greater depth
Conclusion

• Most crosschain transaction scenarios conceptually involve coordinated state change across networks, requiring atomicity.

• In practice however, these properties are not guaranteed by most crosschain protocols. This introduces various failure scenarios requiring trusted actors for remediation.

• Such trusted actors in bridge design are often the source of security vulnerability, as seen in recent hacks.

• Protocol enforced atomicity, could increase resilience and security, and reduce trust requirements.

• As adoption of crosschain protocols grows, and more complex crosschain use-cases emerge, the need for atomicity guarantees could become more salient.

• Whether the limitations of current atomic protocols can be overcome to offer viable, cost-effective and scalable alternatives to non-atomic approaches is to be seen.