HyperService: Interoperability and Programmability Across Heterogeneous Blockchains

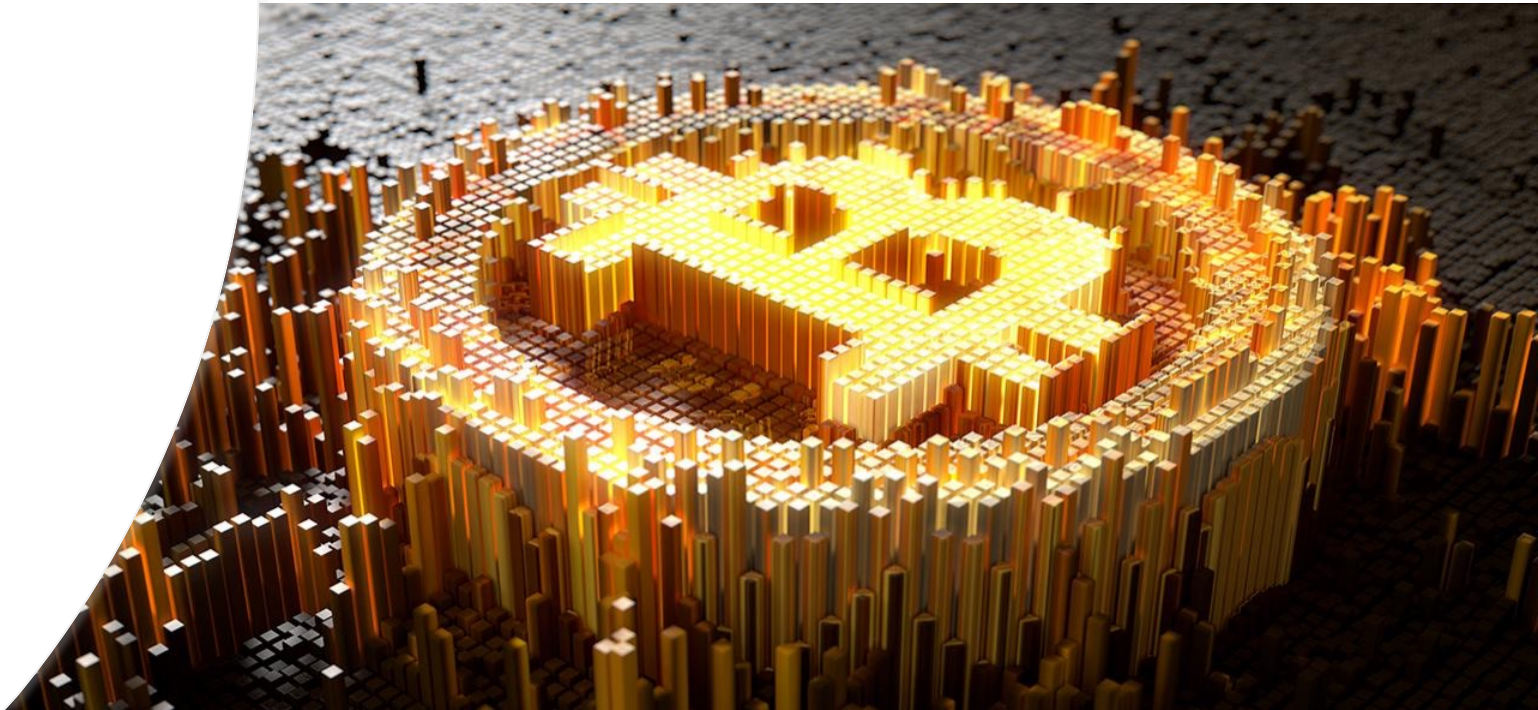
- HyperService-跨异构区块链的互操作性和可编程性

- 汇报人：贺港龙

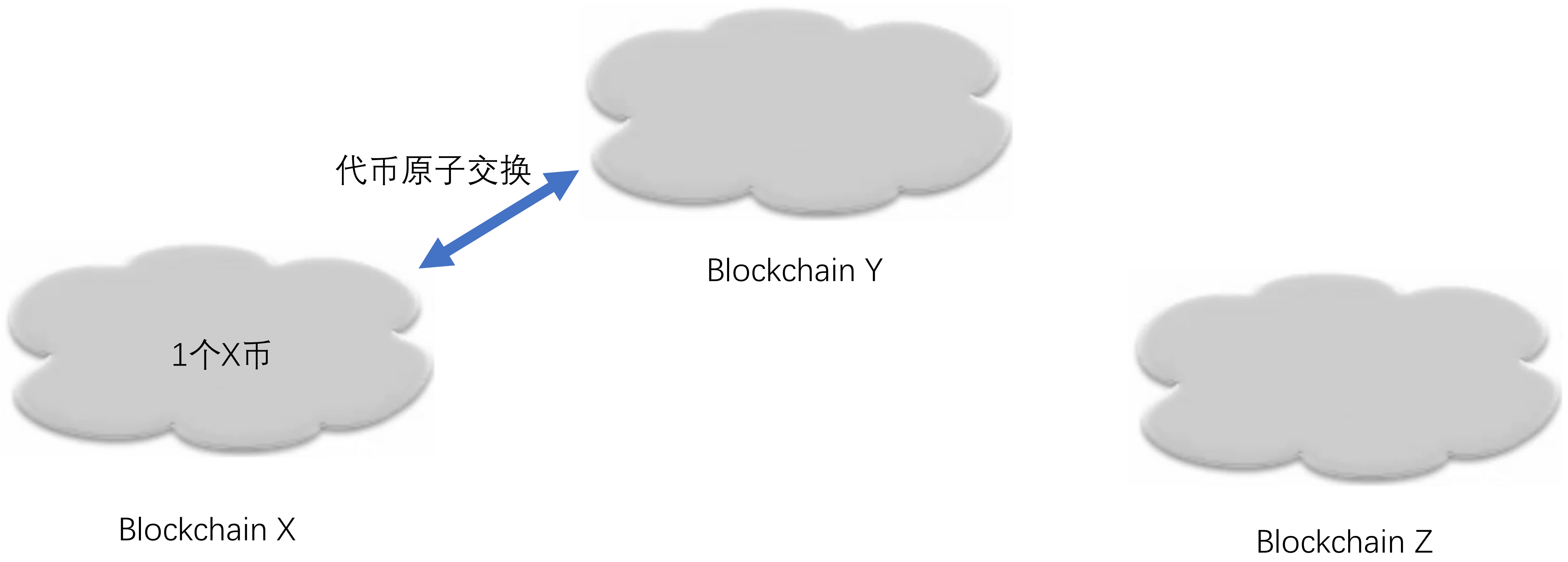
- 2021年4月13日

区块链两大职能

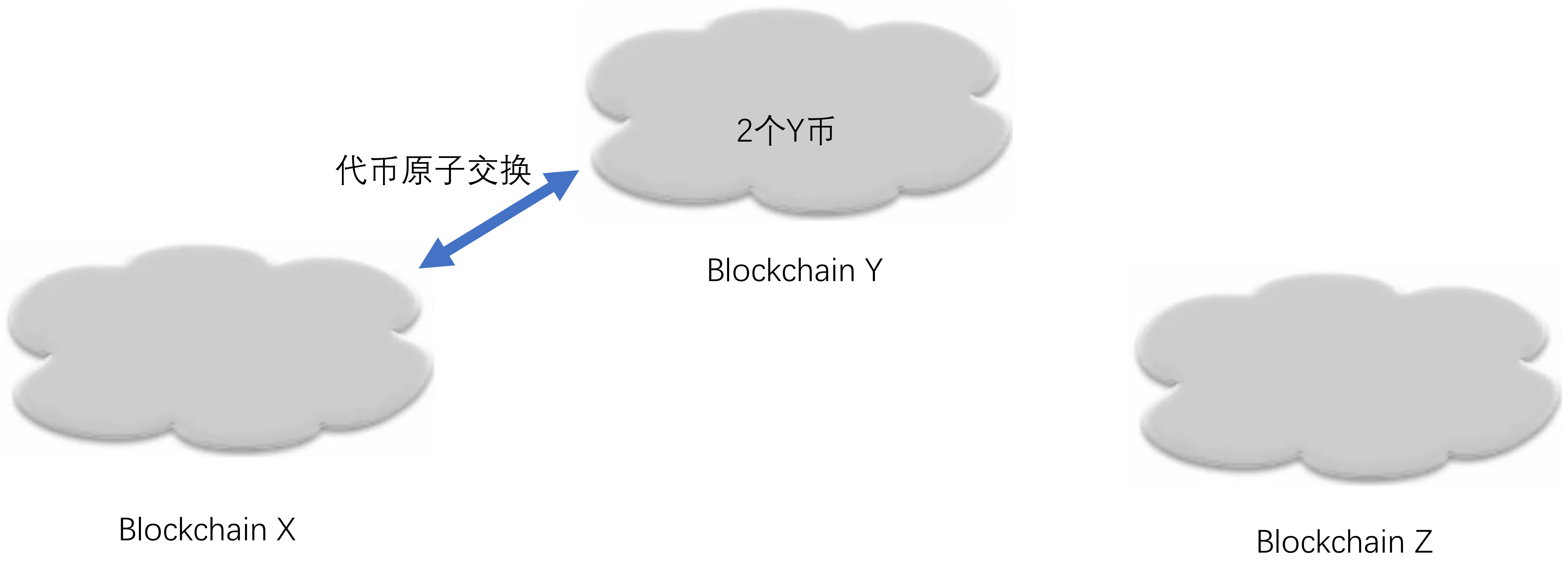
- 交易网络（加密数字货币）
- 智能合约平台



跨链代币原子交换



跨链代币原子交换



跨链智能合约调用

Blockchain Y

genuinePrice = StrikePrice

def CashSettle(sbareCount: uint256, genuinePrice: wei_value):

Blockchain X

uint public StrikePrice;

StrikePrice = $10

Blockchain Z

智能合约：　分布式账本 → 可编程状态机

互操作性和可编程性的挑战：

- 区块链的异构性

Contract language

Blockchain X

Consensus Efficiency & Finality

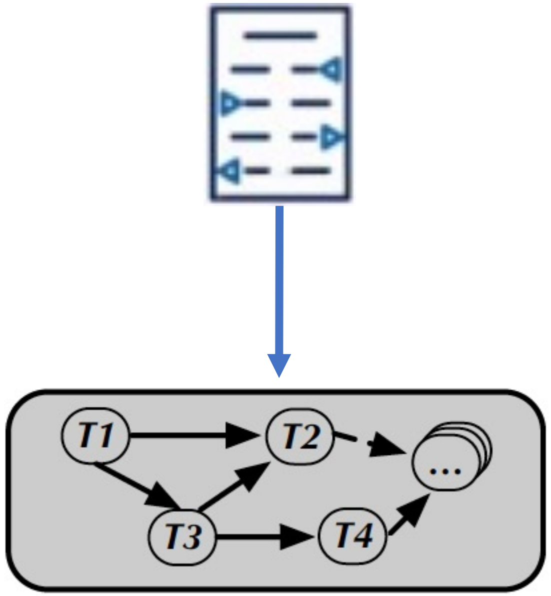
Blockchain Y

Transactions Not-Synchronized

Blockchain Z

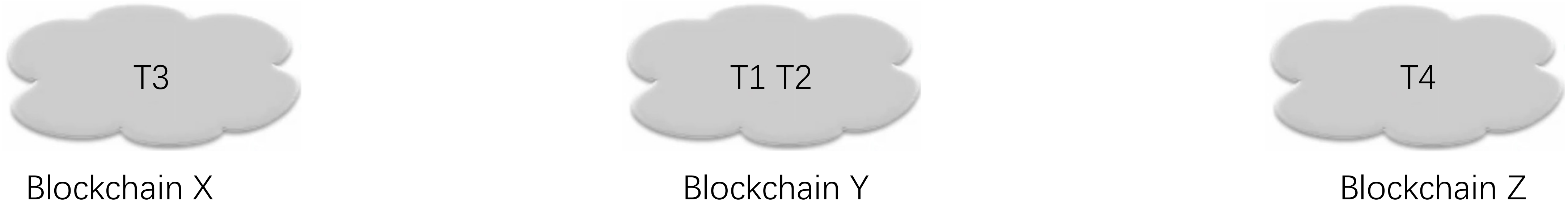
互操作性和可编程性的挑战：

跨链dApps：不仅包含令牌转移，还包含更复杂的操作



dApp交易执行：

·不同区块链上的交易
·按指定顺序执行交易
·下游交易依赖上流交易产生的区块链状态



HyperService架构

- 面向开发者：一个编程框架
  - Universal State Model（USM，通用状态模型）：封装区块链的异构性
  - HyperService Language（HSL）：编写dApps的高级语言

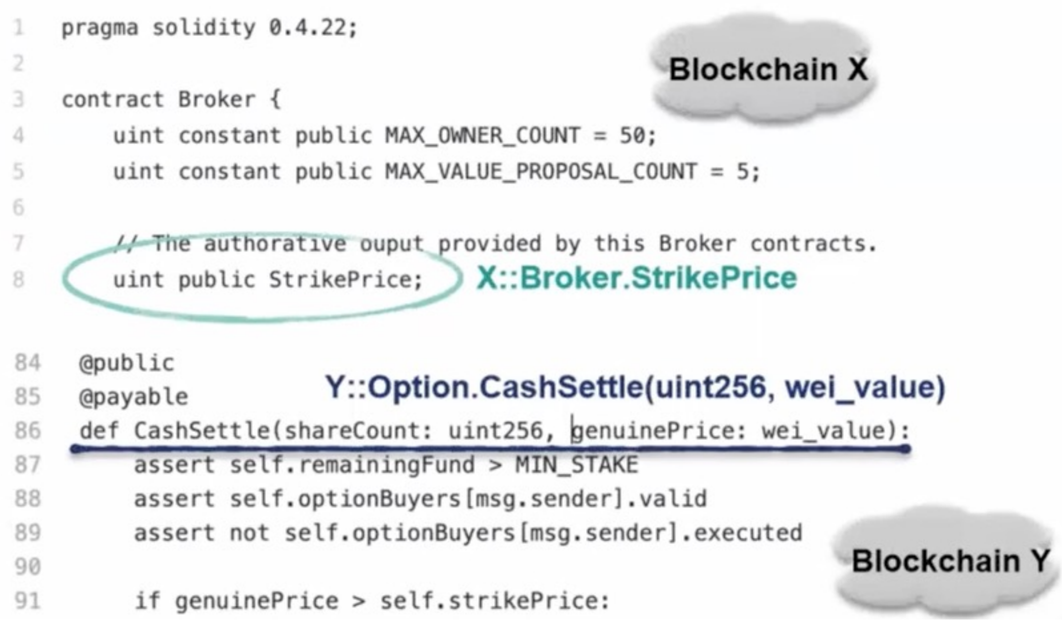
- 面向底层区块链：
  - Network Status Blockchain（NSB，网络状态区块链）：为跨链操作提供去中心化的trust anchor
  - Insurance Smart Contract（ISC，保险智能合约）：保证跨链操作正确执行

Programming Framework – Universal State Model

$$\mathcal{M} = \{\mathcal{E}, \mathcal{P}, \mathcal{C}\} = \{Entities, Operations, Constraints\}$$

实体：可以参与到跨链操作的独立对象

Entities	Attributes
account	address, balance, unint
contract	state variables[], interfaces[], source

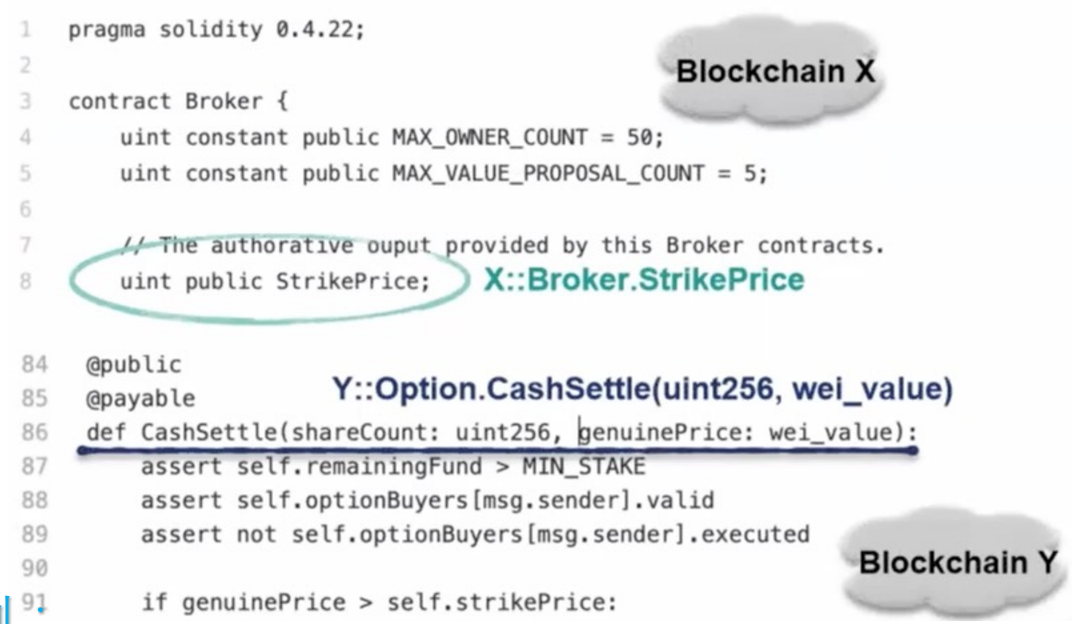


Programming Framework – Universal State Model

$$\mathcal{M} = \{\mathcal{E}, \mathcal{P}, \mathcal{C}\} = \{Entities, Operations, Constraints\}$$

操作：多个实体执行

Operations	Attributes
payment	from, to, value, exchange rate
invocation	interface, parameters[], invoker



调用实例：

Y::Option.CashSettle(10, X::Broker.StrikePrice)

Programming Framework – Universal State Model

$$\mathcal{M} = \{\mathcal{E}, \mathcal{P}, \mathcal{C}\} = \{Entities, Operations, Constraints\}$$

限制：操作之间依赖关系

Entities	Attributes	Operations	Attributes	Dependency
account	address, balance, unint	payment	from, to, value, exchange rate	Precondition
contract	state variables[], interfaces[], source	invocation	interface, parameters[], invoker	deadline

HyperService Programming Language - HSL

import

实体：可以参与到跨链操作的独立对象
account & contract

操作：多个实体执行
payment & invocation

限制：操作之间依赖关系
before, after & deadline

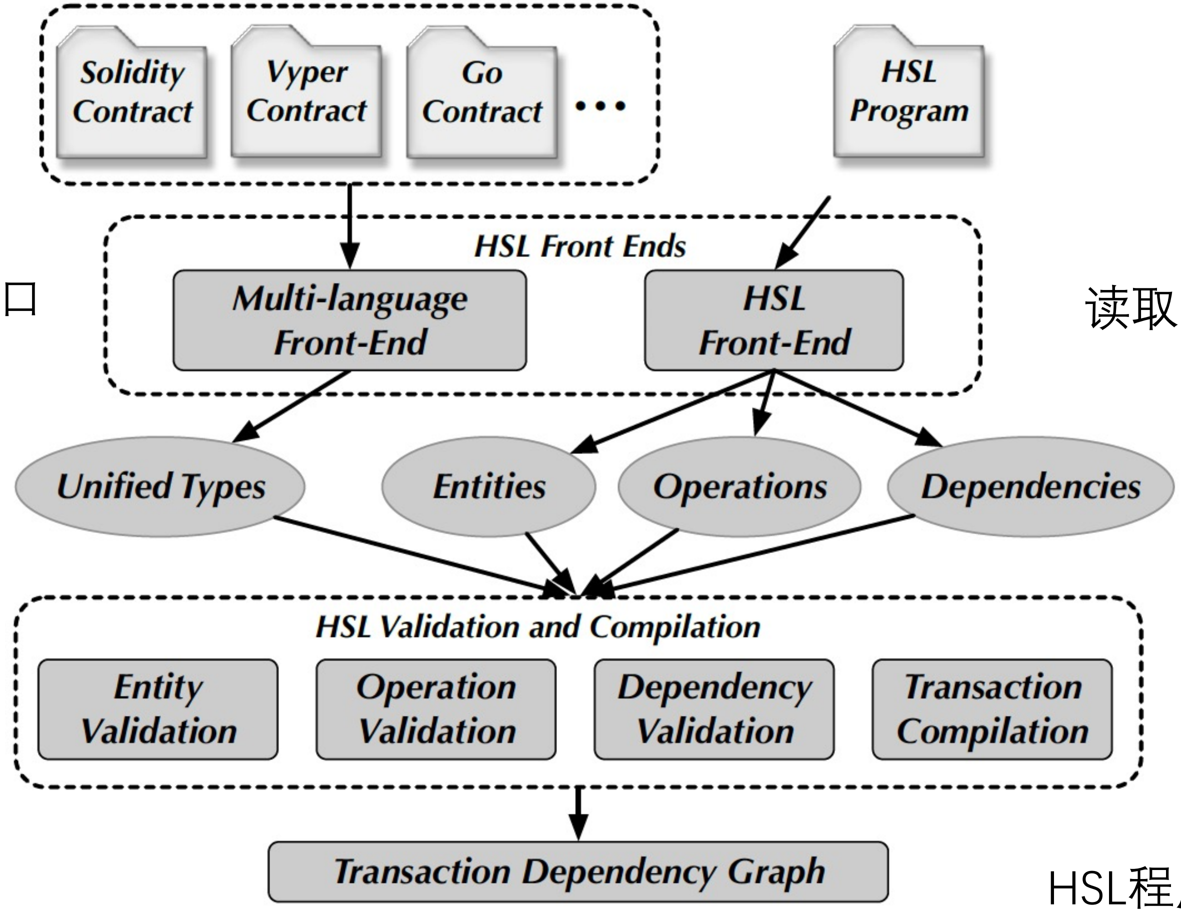
```
# Import the source code of contracts written in different languages.
import ("broker.sol", "option.vy", "option.go")
# Entity definition.
# Attributes of a contract entity are implicit from its source code.
account a1 = ChainX::Account(0x7019..., 100, xcoin)
account a2 = ChainY::Account(0x47a1..., 0, ycoin)
account a3 = ChainZ::Account(0x61a2..., 50, zcoin)
contract c1 = ChainX::Broker(0xbba7...)
contract c2 = ChainY::Option(0x917f...)
contract c3 = ChainZ::Option(0xefed...)
# Operation definition.
op op1 invocation c1.GetStrikePrice() using a1
op op2 payment 50 xcoin from a1 to a2 with 1 xcoin as 0.5 ycoin
op op3 invocation c2.CashSettle(10, c1.StrikePrice) using a2
op op4 invocation c3.CashSettle(5, c1.StrikePrice) using a3
# Dependency definition.
op1 before op2, op4; op3 after op2
op1 deadline 10 blocks; op2, op3 deadline default; op4 deadline 20 mins
```

Programming Framwork Core – HSL Program Compilation

从导入的合约提取状态变量和接口

统一不同的合约语言

兼容性：类型检查
可验证性：状态变量
可行性：无环



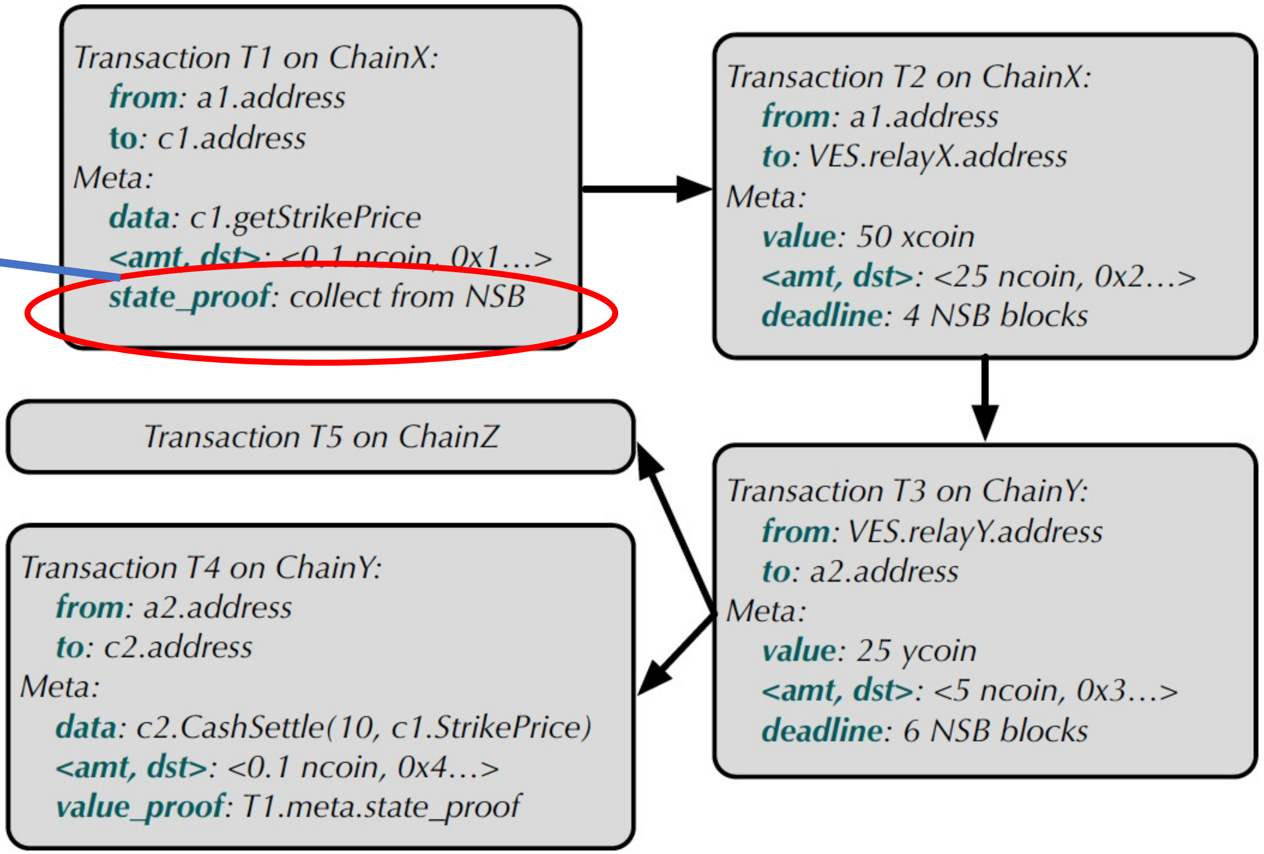
读取并解析HSL程序

从HSL程序提取{E, O, D}

HSL程序可执行文件

Transaction Dependency Graph(TDG) – HSL程序可执行文件

T1产生的状态会在后续使用，
需要提交状态变量证明



节点定义：
执行区块链交易的全部信息
确保正确执行的元信息
边定义：交易执行顺序

Universal Inter-Blockchain Protocol (UIP)

- 所有跨链dApps需要遵守
- 完全去中心化：无权威，相互之间无需信任

- 提供安全属性：正确性、金融原子性、可结算性（问责）

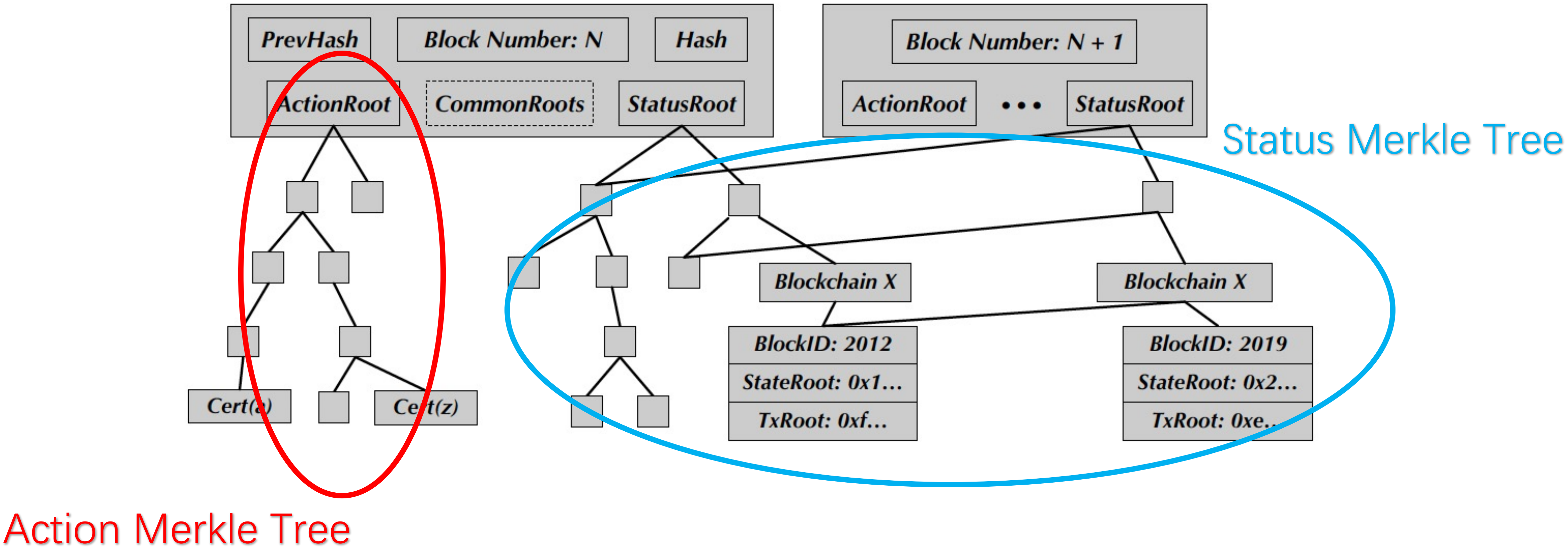
- 两大组件：
- Network Status Blockchain（NSB）：去中心化trust anchor
- Insurance Smart Contract（ISC）：无需信任的仲裁决策

Universal Inter-Blockchain Protocol (UIP)

UIP安全属性

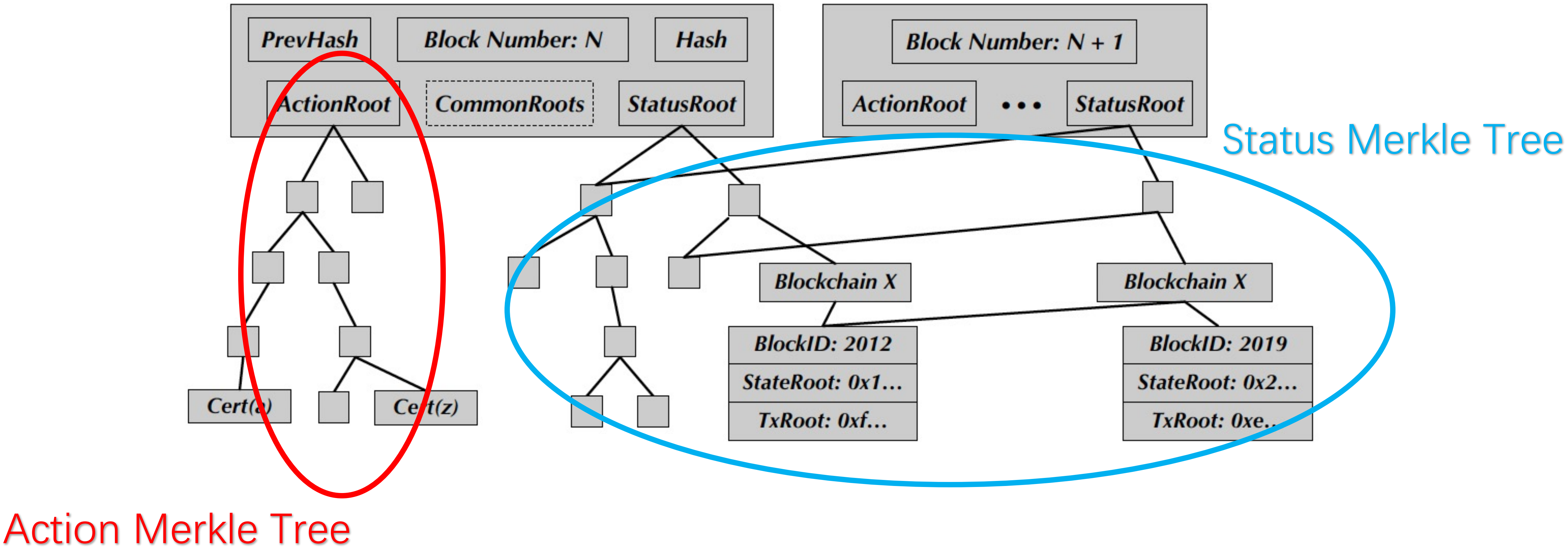
- 金融原子性：dApp执行要么正确执行，要么金融可逆
- 可结算性：dApp执行到任意状态中止都可以进行结算和追责
- 正确性：如果所有交易都在有限时间内完成，那么遵守UIP的可信的实体可以保证dApp会正确执行

Network Status Blockchain – NSB设计



Proof of Actions(PoAs):对交易执行状态提供证明

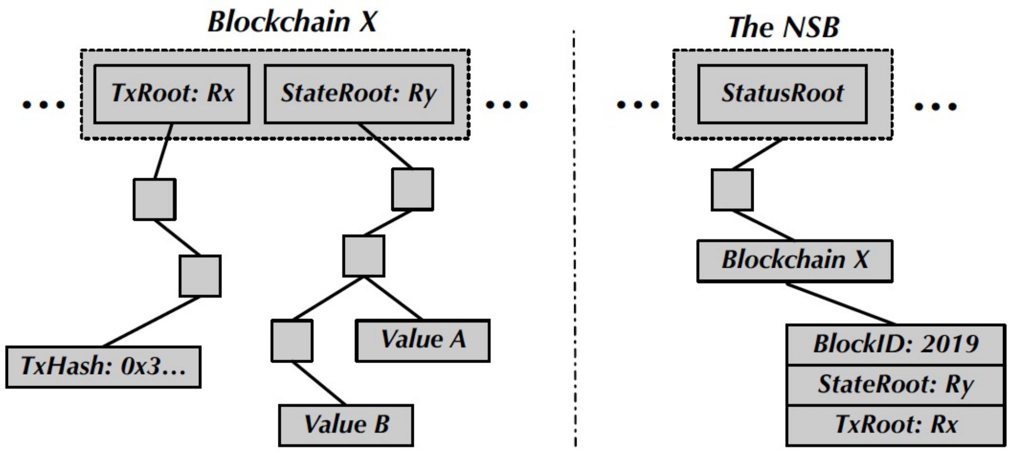
Network Status Blockchain – NSB设计

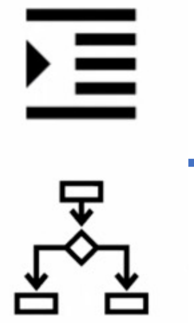


Proof of Actions(PoAs):对交易执行状态提供证明

Insurance Smart Contract （ISC）

Merkle Proofs

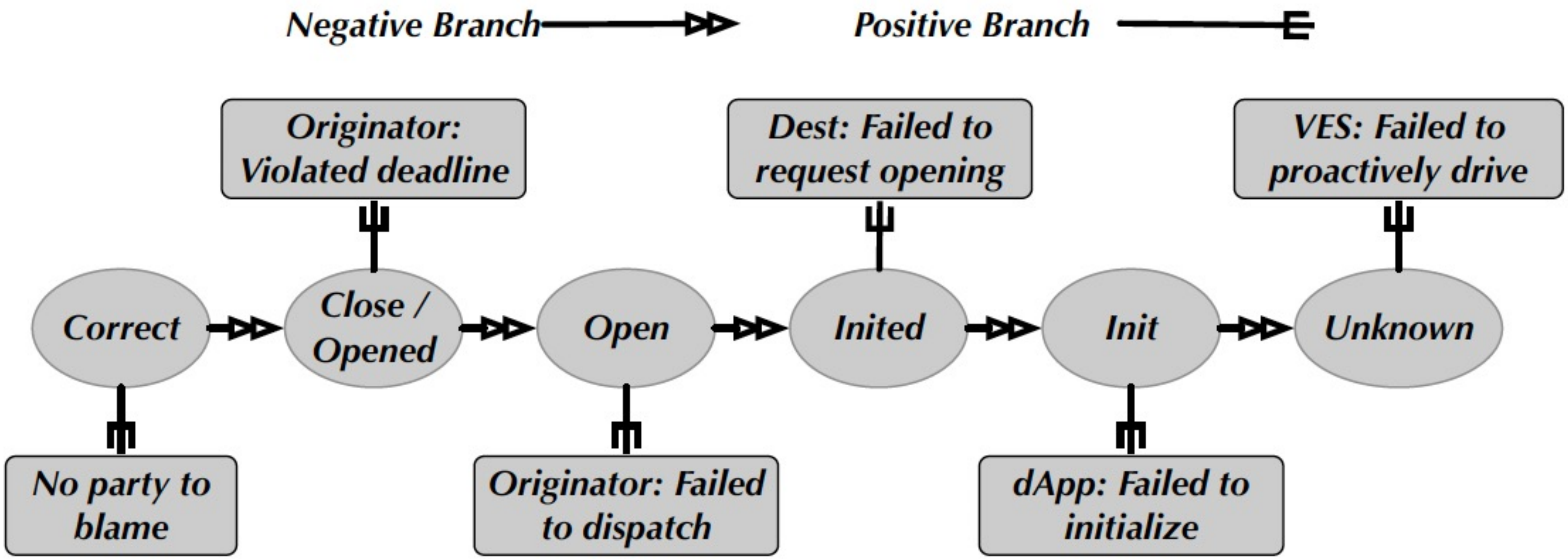




```
if CorrectExecution:
    Pay service fee
else:
    Revert effective fund
    Enforce accountability
```

Decision Logic

Insurance Smart Contract （ISC）



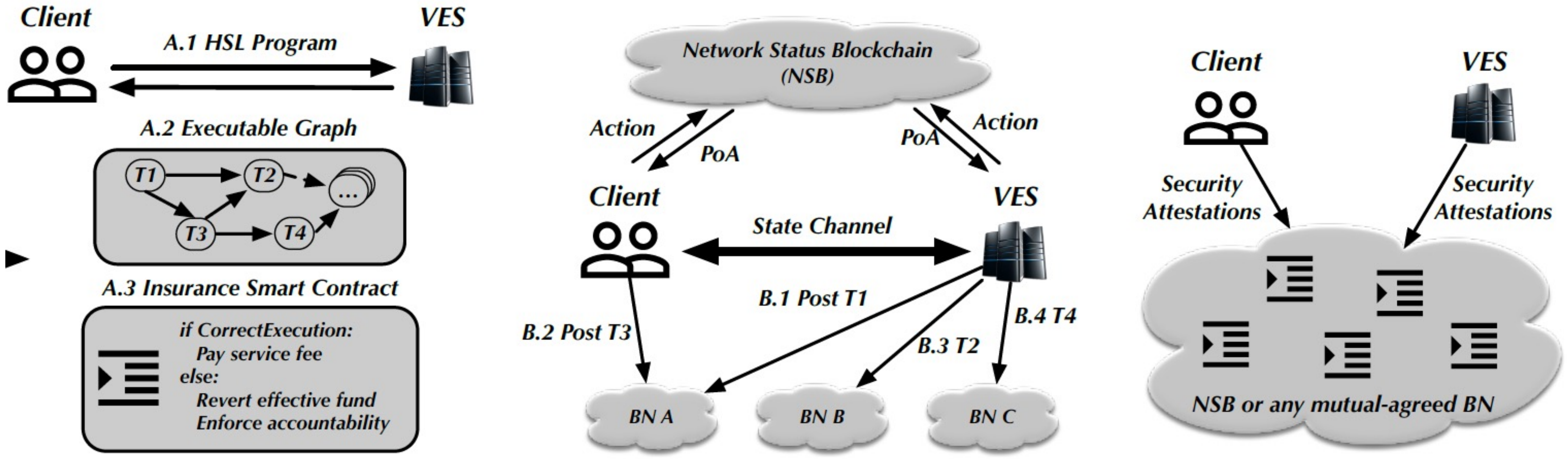
Verifiable Execution Systems （VES）

ISC执行中，承担桥接作用

相当于用户实体+链下服务器

dApps和VES都运行UIP协议

HyperService工作流



Phase A. HSL Program Compilation

Phase B. Cross-Chain Execution

Phase C. Insurance Claim

HyperService应用场景

不同链之间参数转移

区块链分片

小型私链与大型公链交互

实验

- 逻辑代码实现：
- 论文发表时：3w+行代码
- https://github.com/HyperService-Consortium