ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

□□□

BÁO CÁO ĐỒ ÁN 1

THIẾT KẾ VÀ TRIỂN KHAI MỘT BLOCKCHAIN TẦNG 1 TỐI GIẢN

MÔN HỌC: BLOCKCHAIN VÀ ỨNG DỤNG

Giảng viên hướng dẫn: Ngô Đình Hy

Thành viên nhóm:

STT	MSSV	Họ và tên
1	22120279	Phạm Tài Phúc
2	22120292	Nguyễn Hải Quân
3	22120129	Đặng Đức Huy
4	22120137	Nguyễn Minh Huy
5	22120276	Nguyễn Lê Anh Phúc

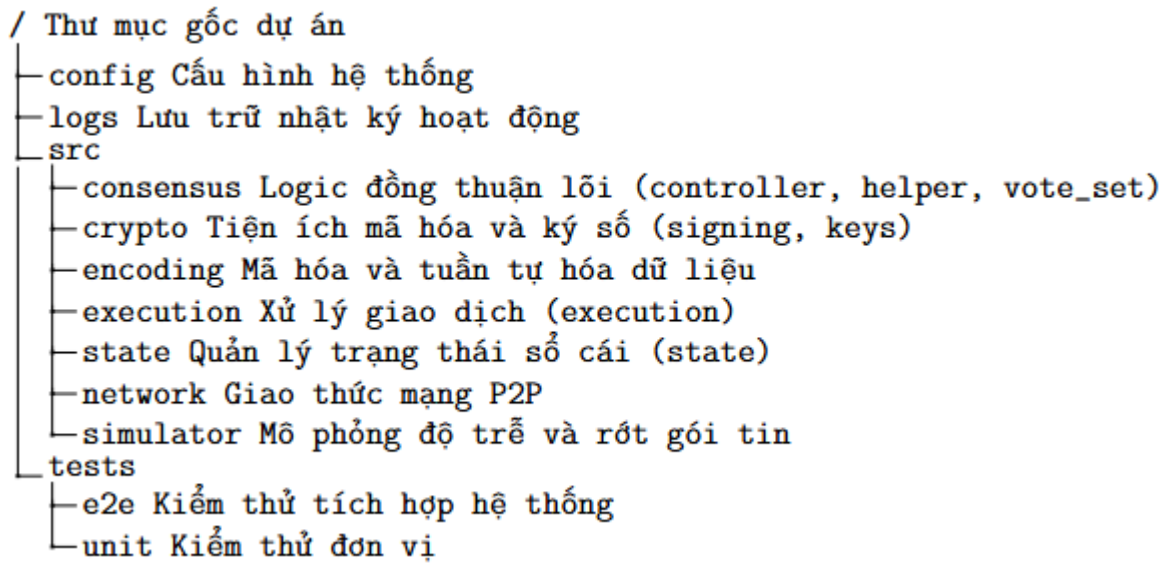
Thành phố Hồ Chí Minh, ngày 01 tháng 12 năm 2025

Mục lục

1. Phân công nhiệm vụ

STT	MSSV	Họ và tên	Nhiệm vụ	Đánh giá
1	22120279	Phạm Tài Phúc	Crypto Layer, Encoding Layer, State Machine, Block&BlockHeader, Ledger, Unit Tests cho các phần trên	100%
2	22120292	Nguyễn Hải Quân	Consensus Layer	100%
3	22120129	Đặng Đức Huy	Execution Layer	100%
4	22120137	Nguyễn Minh Huy	Network, Deteminism, Logging, Test cho các phần trên	100%
5	22120276	Nguyễn Lê Anh Phúc	Consensus Layer	100%

2. Cấu trúc dự án



3. Giới thiệu hệ thống

3.1. Mục tiêu hệ thống

Hệ thống được xây dựng nhằm mô phỏng một blockchain Layer 1 tối giản có khả năng đạt được tính cuối cùng (finality) xác định, ngay cả khi hoạt động trong một môi trường mạng không đáng tin cậy — có trễ, mất gói, trùng lặp và giới hạn băng thông. Thông qua mô phỏng này, chúng ta mục tiêu kiểm chứng bốn thuộc tính cốt lõi của blockchain hiện đại:

- **Dữ liệu xác thực (Authenticated Data)**: mọi giao dịch, khối và phiếu bầu đều được ký với domain separation để ngăn giả mạo và replay.
- **Lan truyền mạng (Network Propagation)**: mô phỏng truyền thông điệp không tin cậy nhưng vẫn đảm bảo hệ thống đạt một trạng thái thống nhất.
- **Đồng thuận chịu lỗi (Fault-Tolerant Consensus)**: triển khai giao thức hai pha kiểu Tendermint/HotStuff (Prevote – Precommit) để đạt finality với đa số validator.
- **Thực thi trạng thái xác định (Deterministic State Execution)**: các nút xử lý giao dịch theo cùng thứ tự để đảm bảo cùng một state root.

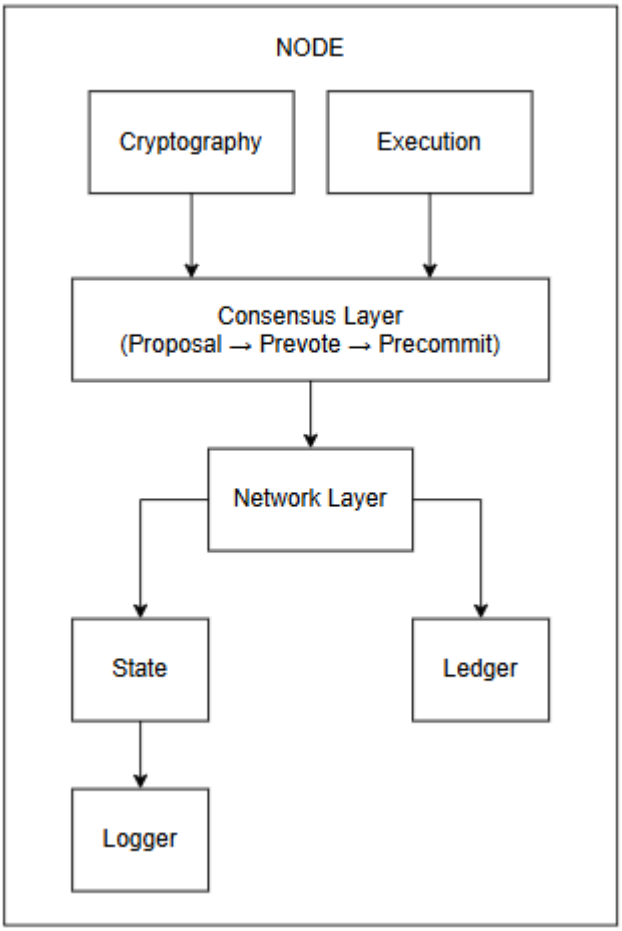
3.2. Tóm tắt các lựa chọn thiết kế chính

- Sử dụng một sơ đồ chữ ký nhất quán (Ed25519) cho toàn bộ thông điệp.
- Dùng SHA-256 làm hàm băm và định nghĩa mã hóa byte xác định cho state, giao dịch và header.
- Mô phỏng mạng với độ trễ ngẫu nhiên, giới hạn băng thông và khả năng drop/duplicate.
- Xây dựng lớp đồng thuận hai pha đảm bảo an toàn tuyệt đối và tính sống động khi độ trễ bị chặn trên.
- Tách hệ thống thành các module: Cryptography → Network → Consensus → Execution → State → Logger.

Mục tiêu cuối cùng của hệ thống là đảm bảo mỗi lần chạy cho cùng cấu hình đều tạo ra cùng log và cùng state hash, thể hiện tính xác định và đúng đắn của blockchain.

4. Thiết kế hệ thống

4.1. Sơ đồ hệ thống: Hệ thống gồm nhiều node , mỗi node chạy bộ ba lớp chính:



Nhiều nút kết nối qua Network Simulator tạo thành blockchain mô phỏng.

4.2. Các module chính và tương tác

4.2.1. Lớp mạng & Mô hình giao tiếp

a) Thiết kế

- Lớp mạng trong hệ thống được xây dựng với mục tiêu mô phỏng lại các đặc tính của một môi trường mạng không tin cậy nhưng vẫn có thể kiểm soát và tái lập được. Thay vì cho các node giao tiếp trực tiếp, toàn bộ message đều đi qua **NetworkSimulator**. Đối tượng này chịu trách nhiệm gán độ trễ, xác suất rơi gói, nhân đôi gói tin và áp dụng các giới hạn về băng thông, từ đó tạo ra các kịch bản mạng xấu để kiểm thử thuật toán đồng thuận.

- Bên cạnh đó, simulator sử dụng một seed ngẫu nhiên cố định cho bộ sinh số ngẫu nhiên, giúp mỗi lần chạy với cùng cấu hình sẽ sinh ra cùng một chuỗi sự kiện mạng. Điều này rất quan trọng cho việc phân tích và debug, vì nhóm có thể tái hiện chính xác một kịch bản lỗi chỉ bằng cách lặp lại cấu hình ban đầu.

b) Thành phần chính

- src/network/simulator.py: lớp NetworkSimulator + NetworkConfig
  - Delay, drop_rate, duplicate_rate; reorder qua hàng đợi ưu tiên.
  - Rate-limit per-sender/per-link và giới hạn bytes inflight; hàng đợi backpressure theo băng thông link.
  - Auto block/unblock khi vượt ngưỡng inflight (timeout tự mở).
  - Header-before-body: body bị chặn nếu chưa thấy header của receiver.
  - Log chi tiết mọi sự kiện (send/delay/deliver/drop/block/unblock/auto_block) kèm timestamp/height; RNG có seed để deterministic.
- src/network/__init__.py: import NetworkSimulator, NetworkConfig.
- src/simulator/node.py: node mẫu dùng simulator, lưu inbound, broadcast header/body.

4.2.2. Lớp mật mã & Định danh

Lớp mật mã trong hệ thống này có vai trò đảm bảo an toàn, tin cậy và tính xác thực (authentication) cho toàn bộ đối tượng: Transaction, BlockHeader và Vote. Tất cả được triển khai trên bộ thư viện Ed25519 với các thành phần chính

a) Cặp khóa Ed25519 Identity Layer

Trong src/crypto/keys.py: Hệ thống tạo ra cặp khóa Ed25519:

- Private key (SigningKey): Dùng để ký Transaction và BlockHeader.
- Public key (VerifyKey): Được phát tán cho toàn mạng và dùng để kiểm tra chữ ký

Điều này thiết lập nhận diện duy nhất cho mỗi node / validator trong hệ thống.

Ưu điểm của Ed25519:

- Bảo mật cao hơn ECDSA trong mô hình hiện đại.

- Hiệu năng cao.
- Kích thước khóa nhỏ (32 bytes public key, 64 bytes signature).
- Đủ tiêu chuẩn dùng cho blockchain (Solana, Libra/Diem đều dùng Ed25519).

b) Domain Sepearation (Chống Replay)

Cơ chế Domain Separation (Phân tách ngữ cảnh - Chống Replay): Để ngăn chặn tấn công phát lại (Replay Attack) giữa các mục đích khác nhau hoặc giữa các chuỗi (Chain) khác nhau, hệ thống áp dụng kỹ thuật Domain Separation. Trước khi ký, mọi dữ liệu đều được thêm tiền tố ngữ cảnh cụ thể:

- Transaction: TX:<chain_id>|json_payload
- Block Header: HEADER:<chain_id>|json_payload
- Vote: VOTE:<chain_id>|json_payload

Cơ chế này đảm bảo mọi chữ ký hợp lệ cho Giao dịch không thể bị cắt dán để giả mạo thành chữ ký cho Khối hoặc Phiếu bầu và một giao dịch hợp lệ trên chuỗi “TestNet-A” sẽ trở nên vô hiệu trên chuỗi”TestNet-B” do khác chain_id.

4.2.3. Lớp Đồng thuận

Tầng đồng thuận đóng vai trò trái tim của hệ thống, chịu trách nhiệm đảm bảo các node đạt được thỏa thuận về khối (block) tiếp theo ngay cả khi có node lỗi hoặc node xấu (Byzantine).

Cơ chế hoạt động và an toàn:

a) Quy trình 3 bước (3-Phase Process)

- Bước 1: Propose (Đề xuất)
  Proposer (được chọn theo Round-Robin) phát tán khối mới hoặc đề xuất lại khối đang bị khóa (nếu có) từ vòng trước.
- Bước 2: Prevote (Bỏ phiếu sơ bộ)
  Các node thẩm định đề xuất: nếu hợp lệ và không xung đột với trạng thái khóa hiện tại, gửi phiếu PREVOTE cho khối; ngược lại gửi phiếu NIL.
- Bước 3: Precommit (Cam kết)
  Khi thu thập đủ đa số (>2/3) phiếu PREVOTE cho một khối,

node thực hiện Lock (khóa) vào khối đó và phát tán phiếu PRECOMMIT để xác nhận đồng thuận.

b) Cơ chế An toàn (Safety: Proof-of-Lock)
   - Cơ chế An toàn (Proof-of-Lock): Để ngăn chặn phân nhánh (fork) và tấn công "Nothing-at-stake", **ConsensusController** áp dụng quy tắc khóa nghiêm ngặt: khi một node đã gửi phiếu PRECOMMIT cho khối B, nó buộc phải giữ trạng thái khóa vào khối đó. Ở các vòng tiếp theo, node chỉ được phép đề xuất hoặc bỏ phiếu lại cho B, trừ khi nhận được bằng chứng mở khóa hợp lệ từ đa số mạng.

c) Đảm bảo Sự sống (Liveness: Timeout)
   - Đảm bảo Sự sống (Liveness): Hệ thống thiết lập giới hạn thời gian (Timeout) cho từng bước đồng thuận. Khi hết giờ mà chưa đạt điều kiện chuyển tiếp, **ConsensusController** kích hoạt sự kiện on_timeout để bỏ phiếu NIL hoặc chuyển sang vòng (Round) mới, ngăn chặn hệ thống bị treo do lỗi mạng hoặc Proposer offline.

4.2.4. Lớp thực thi & Trạng thái

Lớp này chịu trách nhiệm chuyển đổi các giao dịch (Transactions) thành sự thay đổi cụ thể trên dữ liệu (State). Logic nàm ở hàm appy_transaction, tuân thủ 4 quy định:

1. **Kiểm tra tính xác thực (Authentication)**: Hệ thống xác minh chữ ký Ed25519 của giao dịch. Nếu dữ liệu bị sửa đổi hoặc chữ ký giả mạo, giao dịch bị từ chối ngay lập tức.
2. **Kiểm tra tuần tự (Sequence/Nonce Check)**: Mỗi tài khoản có bộ đếm nonce. Hệ thống chỉ chấp nhận giao dịch nếu tx.nonce == current_nonce + 1. Điều này ngăn chặn kẻ tấn công gửi lại (replay) các giao dịch cũ haowjc đảo lộn thứ tự thực thi.
3. **Kiểm tra quyền sở hữu (Ownership/Namespace Check)**: Để đơn giản hóa mô hình quyền, hệ thống áp dụng quy tắc tiền tố: Người dùng A chỉ được phép ghi dữ liệu vào csc key bắt đầu bằng A/. Nếu A cố tình ghi đè dữ liệu của B, giao dịch bị coi là không hợp lệ.
4. **Update & Commit**: Sau khi thỏa mãn mọi điều kiện, dữ liệu mới được ghi vào bộ nhớ Key-Value và nonce của người gửi được tăng lên. Cuối cùng, hàm compute_state_hash sẽ băm toàn bộ trạng thái để tạo ra state_root đưa vào Block Header.

4.2.5. Lớp Block & Ledger

- **Block Header**: Chứa các thông tin quan trọng để xác minh: Height, Parent Hash, State Hash, Proposer và Signature.
- **Ledger Safety**: ledger.py hoạt động như một log bất biến. Khi thêm block, nếu sẽ kiểm tra: *Hash(LastBlock) == NewBlock.ParentHash*. Nếu điều kiện không thỏa mãn, Block bị từ chối, đảm bảo chuỗi không bị phân nhánh trái phép.

5. Kiểm thử và kết quả

5.1. Mô tả các trường hợp kiểm thử

5.1.1. Unit Tests

a) Lớp mạng (Network)

- test_network_basic.py & _case.py: header phải đến trước body, topology giới hạn, block/unblock link, ghi log.
- test_network_backpressure.py & _case.py: backpressure theo bytes, auto-block/unblock, serialize throughput.
- test_network_drop_duplicate.py & _case.py: duplicate/drop detection.
- test_network_integration.py: 3 node A→B→C, forward header, body bị reject/accept đúng trình tự.

b) Lớp lõi (Core: Crypto - Encoding - State - Block - Ledger)

- test_crypto_basic.py: Kiểm tra ký và verify bằng Ed25519; sửa nội dung sau khi ký phải làm verify thất bại (chống giả mạo block/tx).
- test_encoding_basic.py: Kiểm tra canonical_json cho kết quả giống nhau dù dict bị shuffle key; kiểm tra encode_tx_for_signing phụ thuộc chain_id (domain separation, chống replay).
- test_state_basic.py:

  Kiểm tra apply_transaction với các trường hợp:

  - TX hợp lệ thì cập nhật data và nonces đúng.
  - Chữ ký sai thì bị reject.
  - Sai prefix key thì reject (ownership rule).
  - Replay nonce (gửi lại nonce cũ) thì reject.
  - Nonce bị lỗi (out-of-order) thì reject (chống replay/reorder).
- test_block_basic.py:

Kiểm tra:

- Chữ ký BlockHeader hợp thì thì verify thành công.
- Thay đổi state_hash sau khi ký thì verify thất bại.
- Sai height hoặc parent_hash thì block bị reject.
- Hash của cùng header tính 2 lần phải giống hệt (deterministic block hash).

- test_ledger_basic.py:

Kiểm tra:

- Ledger trống khởi tạo với height = 0, last_hash=0…0.
- Append 1-2 block liên tiếp với parent_hash đúng thì được chấp nhận, cập nhật height và last_hash.
- Append block với parent_hash sai thì reject, ledger không đổi (append-only + đúng chain).

5.1.2. End-to-end Tests

a) Determinism and Logging

- tests/e2e/determinism_check.py: chạy tx/state 2 lần, ghi logs/determinism_run1/2.log, so log + state hash.
- tests/e2e/determinism_consensus_network.py: chạy consensus+network 2 lần, so network_logs, consensus_logs, finalized_count.
- tests/e2e/run_determinism_suite.py: script tổng determinism, diff các cặp log (determinism, consensus smoke 4/8 nếu có).

5.1.3. Cách chạy

- Tất cả: python tests/run_all_tests.py
- Chỉ determinism: python tests/e2e/run_determinism_suite.py
- Coverage adversarial: python -m unittest tests.e2e.test_consensus_coverage
- Smoke 8 node (dùng config): python tests/e2e/consensus_network_smoke_8nodes.py
- Full sim 8 node: python tests/e2e/run_full_simulation_8nodes.py

5.2. Kết quả mong đợi và thực tế

- python tests/run_all_tests.py
  - Determinism

        - determinism_check: hash trạng thái lần 1/lần 2 giống hệt (44136f…aff8a), log trùng → PASS.
        - determinism_consensus_network: log mạng/log đồng thuận trùng, số block finalize 4/4 → PASS.
        - So sánh các cặp log (determinism, consensus smoke 4/8): tất cả "IDENTICAL" → Determinism suite PASS.
    - Phủ sóng đồng thuận: python -m unittest tests.e2e.test_consensus_coverage chạy 4 test → OK.
    - Smoke 4 node: consensus_network_smoke.py finalized 4/4, log trùng → PASS.
    - Smoke 8 node: consensus_network_smoke_8nodes.py finalized 8/8, log mạng/đồng thuận trùng → PASS.
    - Full pipeline 4 node: state hash mọi node = e0cc1b…d0dcb7f, ledger trùng, "All nodes same state hash: True" → PASS.
    - Full pipeline 8 node: state hash mọi node = e0cc1b…d0dcb7f, "All nodes same state hash: True" → PASS.
- Kết luận:
    - Log determinism: tất cả cặp log so sánh đều giống hệt.
    - State determinism: state hash trùng tuyệt đối giữa các lần chạy (4 và 8 node).
    - Smoke safety/liveness: finalized_count đạt 4 node và 8 node.

6. Tài liệu tham khảo

DarkcodeQuan/blockchain-consensus-lab: HCMUS K21 Blockchain Consensus Lab

https://tendermint.com/static/docs/tendermint.pdf

https://bitcoin.org/bitcoin.pdf

https://ethereum.github.io/yellowpaper/paper.pdf

https://ed25519.cr.yp.to/