Unit-5

👷 Blockchain Application Development Overview

This unit shifts focus from theory to *practical blockchain engineering*, introducing frameworks and tools used to build decentralized applications (dApps) in both enterprise and public settings.

🧱 Hyperledger Fabric

Architecture

Hyperledger Fabric employs a **modular, permissioned architecture** with distinct node roles[16]:

- **Peers**: Execute transactions (endorsers) and maintain ledgers (committers).
- **Orderers**: Package transactions into blocks and sequence them.
- **Execute-Order-Commit Model**:
  1. **Execute**: Endorsing peers simulate transactions.
  2. **Order**: Orderers sequence transactions into blocks.
  3. **Commit**: Committers validate results and update ledgers.
     This separation enables parallelism, scalability, and flexible consensus mechanisms[15].

Identities and Policies

- **Membership Service Providers (MSPs)**: Define identity management rules using X.509 certificates issued by Fabric CA. MSPs authenticate participants and enforce organizational policies[32].
- **Attribute-Based Policies**: Combine user attributes (e.g., roles, clearance) with logical operators (`AND`/`OR`) for granular access control[2].
- **Hierarchy**: Supports organization-level MSPs for entity-specific rules and consortium-level MSPs for cross-organization governance[3].

Membership and Access Control

- **Permissioned Network**: Only vetted entities participate, verified via cryptographic identities[13].
- **Revocation**: Certificates can be invalidated for compromised identities.
- **Policy Enforcement**: Chaincode invocations require endorsement from peers specified by policies[35].

Channels

Private sub-ledgers enabling confidential transactions between specific organizations:

- **Creation**: Members configure channel policies, then join peers using a genesis block[4].
- **Isolation**: Each channel maintains separate transactions, ledgers, and access controls[14].

- **Use Case**: Competing businesses transact bilaterally without exposing data to non-participants16.

Transaction Validation

A multi-stage process ensuring integrity5:

1. **Endorsement**: Clients collect signatures from peers per chaincode policy.
2. **Ordering**: Orderers sequence endorsed transactions into blocks.
3. **Validation**: Committers verify:
    - Signature authenticity
    - Policy compliance
    - Read/write set consistency
      Invalid transactions are flagged but retained for audit5.

Smart Contracts

Hyperledger Fabric (Chaincode)

- **Languages**: Go, Java, Node.js16.
- **Execution**: Runs in Docker containers isolated from peers.
- **Development Flow**:

  ```
  bash





  `# Write chaincode → Package → Install → Approve → Commitpeer lifecycle chaincode package mycc.tar.gzpeer lifecycle chaincode install mycc.tar.gzpeer lifecycle chaincode approveformyorg -C mychannel --package-id mycc:1.0peer lifecycle chaincode commit -C mychannel`
  ```

- **State Management**: Uses key-value stores (LevelDB/CouchDB)6.

Ethereum

- **Language**: Solidity.
- **Compilation**: Converted to EVM bytecode via `solc`.
- **Deployment**: On-chain via transactions7.
- **Limitations**: Public networks lack Fabric’s privacy features7.

Comparative Overview

Platform	Use Case	Consensus	Privacy
Ripple	Cross-border payments	Federated Byzantine Agreement	Limited (public ledger)
Corda	Financial agreements	Notary-based	Transaction-level (flow frameworks)
Fabric	Enterprise B2B	Pluggable (Raft, BFT)	Channels/MSPs[13]

Hyperledger Fabric's modular design, channel-based privacy, and policy-driven access control make it ideal for complex enterprise applications requiring auditability and multi-party trust[136]. Its separation of execution, ordering, and commitment phases optimizes performance while maintaining Byzantine fault tolerance[5].

🌊 Overview of Ripple and Corda

- **Ripple**: Focused on *real-time cross-border payments*. Uses a **consensus protocol** (not PoW or PoS). Banks use it to settle international payments quickly and efficiently.
- **Corda**: Designed for **financial institutions**. Offers a permissioned blockchain that **doesn't broadcast transactions globally**—only to those involved. Supports smart contracts in **Kotlin** or **Java**.