**Day 1: Progress Report – Blockchain Learning**

Completed 4 lectures covering the following core topics:

* Centralized, decentralized, and distributed systems
* Definition and structure of blockchain
* Hash functions, digests, and Merkle trees
* Miners and smart contracts
* Block structure and transaction data
* Block hashes and their significance
* Permissioned vs. permissionless blockchains
* Challenge–response in consensus mechanisms
* Provenance tracking in data
* IPFS and the distributed web

**Day 2: Progress Report – Blockchain Learning**

**Lectures Covered:** Lecture 5 to Lecture 9 (5 lectures)

**Focus:** Cryptographic foundations and Bitcoin fundamentals

**Core Topics Learned:**

**🔐 Cryptographic Foundations**

* Cryptographic Hash Functions  
  • Properties: Collision Resistance, Pre-image Resistance, Avalanche Effect, Puzzle-Friendliness  
  • Introduction to the SHA-256 algorithm
* Hash Pointers and Hashchains  
  • Concept of hash pointers as secure references to previous data  
  • Use of hashchains in ensuring data integrity

**🌲 Merkle Trees**

* Structure and purpose in efficient data verification
* Use in blockchain for verifying large transaction sets efficiently

**✍️ Digital Signatures & Public Key Cryptography**

* Purpose of digital signatures in verifying authenticity
* Public and Private Key Mechanism  
  • Public Key (used for encryption)  
  • Private Key (used for decryption)
* RSA Algorithm: Basics of public key encryption
* Digital Signing Workflow  
  • Signing with private key, verification with public key  
  • Efficiency via signing the hash (digest) instead of full message

**💰 Cryptocurrency Concepts**

* Introduction to Cryptocurrency secured by hashchains and digital signatures
* **Bitcoin Basics**  
  • Coin creation and supply  
  • Sending payments  
  • Double spending problem and blockchain’s solution  
  • Anonymity in Bitcoin

**📜 Bitcoin Script**

* Introduction to Bitcoin’s scripting language (similar to Forth)  
  • Stack-based, uses postfix notation  
  • Common instructions and examples of real-world Bitcoin scripts

**🌐 Bitcoin P2P Network**

* Joining the Bitcoin peer-to-peer network
* Transaction broadcast (flooding mechanism)
* Mining and block generation  
  • Mining puzzle and Proof-of-Work  
  • Block flooding and propagation  
  • Consensus via longest valid chain  
  • Handling latency in block propagation

**Day 3 Progress Report – Blockchain Learning**

Lecture covered: 10 to 13 (4 lectures)

1. Consensus
2. why Consensus
3. example
4. why it can be difficult in certain scenarios
5. Distributed consensus
   1. Faults in distributed systems
      1. Crash fault
      2. Network/portioned fault
      3. Byzantine fault
   2. Properties
      1. Termination
      2. Validity
      3. Integrity
      4. Agreement
6. Synchronous vs Asynchronous system
   1. Asynchronous system
      1. FLP85
   2. Synchronous system
      1. Paxos
      2. Raft
      3. Byzantine fault tolerance (BFT)
7. Correctness of distributed consensus protocol
   1. Safety
   2. Liveliness
8. Consensus in an open system
   1. Traditional distributed consensus protocol
      1. Message passing
      2. Shared memory
9. Consensus in a bitcoin network
10. Consensus algorithm in bitcoin
    1. PoW
       1. Cryptographic hash as PoW
       2. Hashcash PoW
11. Bitcoin Proof of Work
    1. Solving the double spending problem
12. Sybil attack
13. DOS (denial of services) attack
14. Breaking bitcoin PoW
    1. The monopoly problem
    2. PoW power consumption
15. Handling monopoly and power consumption – proof of stake(PoS)
16. PoS (Proof of stake)
    1. Variant of stake
17. PoB (proof of burn)
18. Difference between PoW, PoS and PoB
19. PoET (Proof of elapsed time) {intel}
    1. PoET over trusted environments
20. Mining bitcoins
    1. The life of a miner
    2. Mining difficulty
    3. Setting the difficulty
    4. Hash rate vs difficulty
    5. Mining hardware
    6. Mining pool methods
       1. Pay per share (PPS)
       2. Proportional
       3. Pay per last N share (PPLNS)
       4. Pros and cons

**Day 4 Progress Report – Blockchain Learning**

Lectures: 14 to 17

1. Permissioned model
   1. Use cases
      1. Provenance tracking of assets
2. Smart contracts
   1. Design limitations
      1. Sequential execution
      2. Non-deterministic execution
      3. Execution on all nodes
3. Do we really need to execute contracts at each node?
   1. What if the node that executes the contracts is faulty?
   2. Use state machine replication
4. State machine replication
5. State machine
6. Smart contract state machine – crowd-funding
7. Distributed state machine replication
   1. Place copies of the state machine on multiple independent server
   2. Receive client requests, as an input to the state machine
   3. Propagate the inputs to all the servers
   4. Execute the inputs based on the order decided, individually at each server
   5. Sync the state machines across the servers, to avoid any failure
   6. If output state is produced, inform the clients about the output
8. Permissioned blockchain and state machine replication
9. Why distributed consensus
10. Faults in distributed consensus
    1. Crash fault
    2. Network or partitioned faults
    3. Byzantine faults
       1. Malicious behaviour in nodes
       2. Hardware fault
       3. Software error
11. Consensus for three processes
12. Requirement of a consensus algorithm
    1. Termination
    2. Agreement
    3. Integrity
13. Different consensus algorithms
    1. Crash or network faults
       1. PAXOS
       2. RAFT
    2. Byzantine faults (including crash or network failures)
       1. Byzantine fault tolerance (BFT)
       2. Practical byzantine fault tolerance (PBFT)
14. PAXOS
    1. Types of nodes
       1. Proposer
       2. Acceptor
       3. Learner
    2. Making a proposal:
       1. proposer process
       2. Acceptor’s decision making
       3. Acceptor’s message
    3. Accepting a value:
       1. Proposer’s decision making
       2. Accept message
       3. Notifying learner
    4. Single proposer: No rejection
15. Handling failure:
    1. Proposer failure
    2. Dueling proposers
16. RAFT consensus
17. RAFT
18. Electing the leader:
    1. voting request
    2. Follow node’s decision making
    3. Majority voting
19. Multiple leader candidate:
    1. Current leader failure
    2. Simultaneous request vote
20. Commiting entry log
21. Handling failure
22. Byzantine general problem
23. Practical byzantine fault tolerant
    1. Practical byzantine fault tolerant model
       1. Assumptions (asynchronous network, faulty nodes)
       2. 3f+1 nodes req. to tol. f byzantine faults
    2. Practical byzantine fault tolerant algorithm
       1. Three phase protocol
          * Pre-prepare phase
          * Prepare phase (2f) (excluding its own)
          * Commit phase (2f+1) (including its own)
       2. View change protocol
          * Handle primary failure
          * Ensures progress despite faulty/malicious leader/primary
    3. Correctness
       1. Safety
       2. Liveness
24. Consensus in permissioned model

**Day 5 Progress Report – Blockchain Learning**

Lecture covered (18 -24)

1. Blockchain defined
   1. Problem; difficult to track asset transfers in a business network
   2. Solution; shared, replicated, permissioned ledger
      1. Consensus, provenance, immutability and finality
2. Key concepts and benefits of blockchain for business
   1. Shared ledger
   2. Smart contracts
   3. Security
   4. Consensus
   5. Reduced time
   6. Removes cost
   7. Reduces risk
   8. Enables new business model (IoT integration into supply chain)
3. Degree of centralization
   1. From 100% centralized to 100% decentralized in a gradient manner
4. Permissionless vs permissioned blockchain
   1. Access
   2. Scale
   3. Consensus
   4. Identity
   5. Asset
5. The Linux foundation: Hyperledger project
6. Hyperledger fabric: distributed ledger platform
7. Hyperledger composer: accelerating time to value
8. Actors in a blockchain solution
   1. Regulator
   2. B2B transaction
   3. Membership services
   4. Traditional data resources
   5. Traditional processing platforms
   6. Blockchain network operator
   7. Blockchain developer
   8. Blockchain architect
9. Component in a blockchain solution
   1. Ledger
   2. Smart contract
   3. Peer network
   4. Membership
   5. Events
   6. System management
   7. Wallet
   8. System integration
10. Ledger component
    1. Blockchain
       1. A linked list of blocks (hashchain)
       2. Each block describes a set of transactions
       3. Immutable – blocks can’t be tempered
    2. World state
       1. Stores the most recent state of smart contracts/output of transations
       2. Stored in a traditional database (e.g. key value store)
       3. Data elements can be added, modified, deleted, all recorded as transactions on blockchain
11. Block detail (simplified)
    1. The first block known as genesis block
    2. Block contents (explained)
12. Ledger example: A change of ownership transaction
13. How applications interact with ledger
14. Blockchain events
15. Integrating with existing systems – possibilities
16. Byzantine general problem (was supposed to be cover above but dues to series in playlist, got misplaced)
    1. Three generals problem
       1. Lieutenant faulty
       2. Commander faulty
    2. Four generals problem
       1. Lieutenant faulty
       2. Commander faulty
    3. Byzantine generals model
    4. Lamport-shostak-pease algorithm
17. Hyperledger fabric V1 architecture
    1. External CA, membership services, fabric CA
    2. Client application / SDK (HFC)
    3. Ordering service
    4. Peer
       1. Endorser
       2. Committer
       3. Ledger
       4. Chaincode
       5. Events
    5. Admin etc
    6. Nodes and roles
       1. Committing peer
       2. Endorsing peer
       3. Ordering node
    7. Transaction Flow
       1. Endorse > order > validate
    8. Transaction flow in 7 steps
       1. Propose transaction
       2. Execute proposed transaction
       3. Propose response
       4. Order transaction
       5. Deliver transaction
       6. Validate transaction
       7. Notify transaction
    9. Key benefit of transaction flow
18. Ordering services
    1. Configuration options
       1. SOLO
          1. Single node for development
       2. Kafka: crash fault tolerant consensus
          1. 3 nodes minimum
          2. Odd numbers of nodes recommended
19. Channels
    1. Single channel network
    2. Multi-channel network
20. Fabric peer
21. Client application
22. Fabric certificate authority
23. Organisations
    1. Each organisation defines:
       1. Membership services provider (MSP) for identities
       2. Admin (s)
       3. Users
       4. Peers
       5. Orderers (optional)
    2. A network can include many organisations representing a consortium
    3. Each organisation has an ID
24. Consortium network
    1. An example consortium network of 3 organisations
       1. Org 1 and 3 run peers
       2. Org 2 provides the ordering services only
25. MSP overview
26. Transport Layer Security (TLS)
27. User Identities
28. Admin identities
29. Peer and orderer identities
30. Channel MSP information
31. New user registration and enrolment
32. Transaction signing
33. Hyperledger fabric network setup
    1. Configure and start ordering service
    2. Configure and start peer node
    3. Install chaincode
    4. Create channels
    5. Join channels
    6. Instantiate chaincode in channel
34. Endorsement policies
    1. Endorsement system chaincode
    2. Validation system chaincode
    3. Endorsement policy syntax
    4. Endorsement policy example