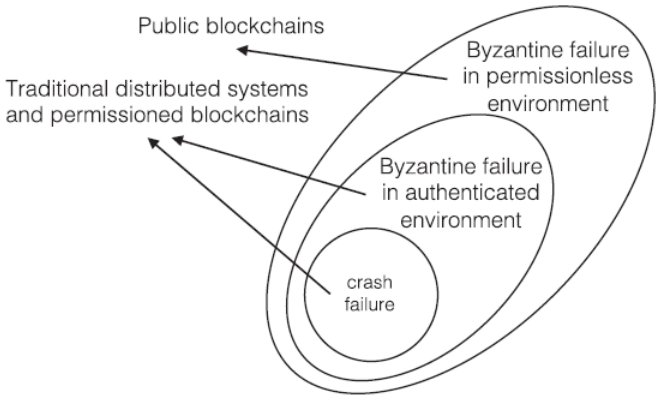
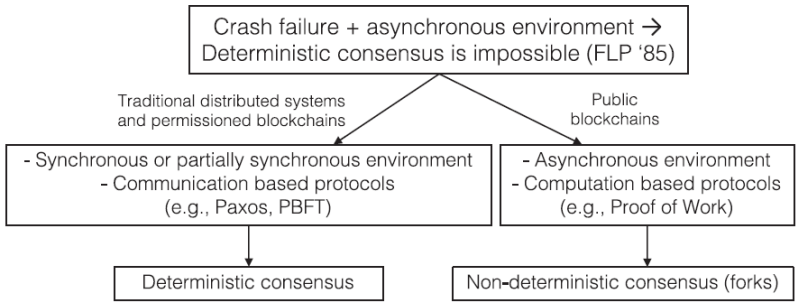
1. **[CO2: 5 Marks] Discuss impossibility theorem and how it is related to distributed consensus. Present the working of Paxos and RAFT consensus algorithms.**

[2.5 Marks] FLP 85 (Fischer, Lynch, and Patterson) says that if there is a single point of failure then there is no deterministic algorithm to solve consensus problem (this is also called in impossibility theorem). Failures causing due to FLP 85 and its related to the distributed consensus is represented as follows





[2.5 Marks] Working of PAXOs and RAFT consensus algorithms

Paxos

- **Paxos is a consensus algorithm that is used to achieve consensus among nodes in a distributed system.**
- **It was first introduced by Leslie Lamport in 1989 and has since become a widely used consensus algorithm in distributed systems.**
- **Paxos is designed to work in a network where there may be failures of nodes and communication links.**
- **The algorithm consists of three types of nodes: proposers, acceptors, and learners.**
- **Proposers propose new values for the consensus, acceptors accept or reject the proposed values, and learners learn the final value that is agreed upon by the acceptors.**
- **Paxos uses a two-phase approach to reach consensus: first, a prepare phase in which proposers send prepare requests to the acceptors, and second, an accept phase in which proposers send accept requests to the acceptors.**

- **In order for a proposed value to be accepted, a majority of acceptors must accept the proposed value.**
- **Paxos is designed to be highly fault-tolerant, able to continue functioning even in the presence of node failures.**
- **It has been used in a variety of distributed systems, including Google's Chubby lock service and the Apache ZooKeeper distributed configuration service.**
- **Despite its robustness and wide usage, Paxos is considered complex and can be difficult to understand and implement.**

RAFT

- **RAFT (Replicated state machine approach) is a consensus algorithm designed for distributed systems.**
- **The main objective of RAFT is to ensure that the same set of commands are executed on all servers in the cluster, in the same order.**
- **RAFT divides time into terms, and each term has a leader that is responsible for receiving client requests and replicating them to the other servers in the cluster.**
- **The leader is elected through a voting process, where the servers in the cluster vote for the candidate that has the most up-to-date log of commands.**
- **In case of a leader failure, a new leader is elected through the voting process**
- **RAFT ensures that the logs are consistent across all servers, by using a log replication mechanism, where the leader sends the log entries to the followers and waits for their confirmation**
- **RAFT is considered a more understandable and easy to implement algorithm compared to Paxos.**
- **RAFT can handle network partitions and leader failures.**
- **RAFT is used in various distributed systems like etcd, CockroachDB, and Consul.**
- **RAFT algorithm is a good choice for distributed systems that have a small number of servers and low write loads.**

2. **[CO3: 5 Marks] Present at least 6 commonly used instructions in Bitcoin scripts. Develop a Bitcoin script and show step by step checking timestamp of two transactions. Script should return TRUE if the timestamps are equal else halt the execution.**

[2.5 Marks] Bitcoin Script is a low-level programming language used in the Bitcoin network to encode the conditions under which a transaction can be spent. Bitcoin Script allows for a wide range of programmatic conditions and instructions to be defined when a transaction is created. Some of the commonly used instructions in Bitcoin Script are:

- **OP_DUP: Duplicates the top item on the stack**
- **OP_HASH160: Computes the SHA-256 hash of the top item on the stack and then the RIPEMD-160 hash**
- **OP_EQUAL: Returns 1 if the inputs are exactly equal, 0 otherwise**
- **OP_EQUALVERIFY: Same as OP_EQUAL, but runs OP_VERIFY afterward**
- **OP_CHECKSIG: Verifies a signature against a public key and pushes the result (True or False) onto the stack**
- **OP_VERIFY: Marks transaction as invalid if top stack value is not True**
- **OP_ADD: Pops two items off the stack, adds them together, and pushes the result back onto the stack.**

- **OP_HASH160: Computes the SHA-256 hash of the top item on the stack and then the RIPEMD-160 hash.**
- **OP_RETURN: can be used to store up to 80 bytes of arbitrary data on the Bitcoin blockchain. It marks a transaction output as provably unspendable, making it an efficient way to burn BTC.**
- **OP_CHECKMULTISIG: Commonly used in Pay To Script Hash (P2SH) transactions. OP_CHECKMULTISIG looks at 3 public keys and 2 signatures in the stack and compares them one-by-one. Funds become spendable only when the order of the signatures matches the order in which the public keys were provided.**

Other op codes https://wiki.bitcoinsv.io/index.php/Opcodes_used_in_Bitcoin_Script

[2.5 Marks] Here is a sample Bitcoin script that checks the timestamps of two transactions:

OP_DUP
OP_HASH160 <Transaction 1 Hash>
OP_EQUAL
OP_VERIFY
OP_DUP
OP_HASH160 <Transaction 2 Hash>
OP_EQUAL
OP_VERIFY
OP_TIMESTAMP <Transaction 1 Timestamp>
OP_DUP
OP_TIMESTAMP <Transaction 2 Timestamp>
OP_EQUAL

In the script, the first two lines hash the transaction IDs and then checks if they are equal to the specified transaction IDs. Then, it compares the timestamps of both transactions and returns 1 if they are equal, 0 otherwise.

3. **[CO3: 2+2+1 = 5 Marks] Let consider the current difficulty solving hash puzzle with 48 leading bits as zero in the target for mining a block in Bitcoin blockchain is 559550, using this information compute the following**
    a) **Next expected difficulty if the time to mine next 2016 blocks is 5040 minutes.**
    b) **Expected hash rate to solve the possible with the difficulty obtained in (a)**
    c) **Mean time to find block**

Mining Difficulty

```
next_difficulty= previous_difficulty *
                 (2 weeks)/(time to mine last 2016 blocks)
```

In the above formula, 2 weeks to be considered as 2 weeks in milliseconds during calculation.

$$\frac{559550 * 2\ weeks\ in\ milliseconds}{5040} = 559550 * (4) = 2238200$$

Next difficulty after 2 weeks is 2238200

Expected Hash rate

- Hash is a random number between 0 and 2^{256}-1
- If the leading 48-bits are zero, the offset for the difficulty is 0xffff * 2^{208}
- The offset for difficulty D is (0xffff * 2^{208}) /D
- The expected number of hash rate we need to calculate to find block with difficulty D is (D * 2^{256}) / (0xffff * 2^{208})

  Expected hash rate = $\frac{(2238200 * 2^{48})}{(65535)} = 34.15 * 10^{15}$ Hashes

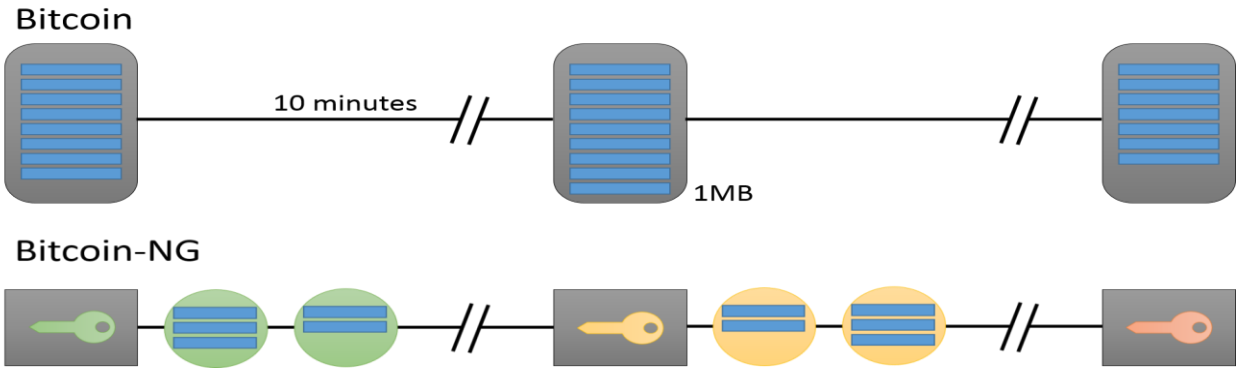
- Mean time to find a block =
  (10 minutes) / fraction of hash rate

  $$Mean\ time\ to\ find\ block = \frac{10 * 60}{34.15 * 10^{15}}\ sec$$

NOTE: Representing the formulas in the answer is MUST. Only half marks will be awarded if the answer is attempted without formula.

4. Answer the following
   a. [CO3: 3 Marks] Functioning of Bitcoin-NG protocol

- Proposed by Eyal, I. Gencer, A. E Sirer E.G & Van Rensesse, "Bitcoin-NG: A Scalable Blockchain Protocol", in NSDI 2016.
- Developed on top of PoW to ensure scalability.
- Issues with PoW
  - Transaction scalability
  - Issues with Forks – Prevents consensus finality, and makes the system unfair –> a miner with poor connectivity is always in disadvantageous position.
- Bitcoin-NG – A scalable PoW protocol
  - Bitcoin think of winning miner as leader
  - Bitcoin-NG decouple Bitcoin's blockchain operation in to two
    - 1) Leader election: Use PoW to randomly select a leader
    - 2) Transaction serialization: The leader serialize the transaction until the new leader is elected.



Bitcoin-NG Keyblock

- Key blocks are used to choose a leader
- A key block contains
  - The reference to the previous block

  - The current Unix Time
  - A coinbase transaction to pay of the reward
  - A target hash value
  - A nonce field
- For key block to be valid, the cryptographic hash of its header must be smaller than the target value
- The key block also contains a public key (name, and key block) used in subsequent microblocks
- Key blocks are generated based on regular Bitcoin mining procedure – (Finding out nonce)
- Key blocks are generated infrequently – the intervals between two key blocks is exponentially distributed.

Bitcoin-NG Microblock

- Once a node generates a key block, it becomes the leader
- As a leader, the node is allowed to generate micro blocks
  - Micro blocks are generated at a set rate smaller than a predefined maximum
  - The rate is much higher than the key block generation.
- A Micro block contains
  - Ledger entries
  - Header
    - Reference to the previous block
    - The current Unix time
    - A cryptographic hash of the ledger entries
    - A cryptographic signature of the header
- Signature uses private key corresponding to the key block public key

b. CO3: 2 Marks] Distributed databases vs. public vs. permission blockchain.

Property	Distributed Database	Public Blockchain	Perm. Blockchain
Append-Only (Immutable)	No	Yes	Yes
Integrity via Cryptography	Varies	Yes	Yes
Turing-Complete Logic	Triggers	Smart Contracts	Smart Contracts
Privacy Easily Achieved	Yes	No	Yes, restricted to members
Easily Auditable	No	Yes	Yes
Requires Incentive or Penalties	No	Yes	No incentive, optional penalties for misbehaving
Requires Consensus	Yes	Yes	Yes
Consensus Protocol	Communication-based	Computation-based	Varies
Explicit Leader in Consensus	Yes	No	Yes
Permanent Consensus	Yes - Once reached	No	Yes—Once reached
Anyone Can Join	No	Yes	No
Must Tolerate Sybil Attacks	No	Yes	No
Must Trust Participant Nodes	Yes	No	Yes
Byzantine Fault Tolerance	Authenticated env.	Open env.	Authenticated env.
Unbounded Forks	No	Yes	No
Scalability and Throughput	Most	Least	Varies
Cost for Participation	No	Yes	No