1. Physical and Logical Clocks
   1. Clock synchronization problems (4 – 6)
   2. Cristian Algorithm (7)
      1. Client initiated (8)
      2. Client requests to sync and gets response
      3. Assumptions & problems included
         1. Zero delay (10)
         2. Symmetric delays (14)
         3. While message passing, client clock is accurate
   3. Caveats of advancing clocks -> Time forwarding (18)
   4. Berkley Algorithm (19)
      1. Server initiated (20)
      2. No dedicated time server
      3. Server polls to all clients & do some math (23)
   5. Network Time Protocol (NTP) (24)
      1. Hierarchy of servers in a strata
      2. Sync w each other in a vertical/horizontal manner
   6. Logical Clocks (32)
      1. Is like a common clock that all servers share
      2. Update with the MAX value
      3. Total global ordering (41)
      4. Causality sucks (45)
   7. Causality violation (50)
      1. Some actions needs to be done earlier than the rest
      2. Can be detected by vector, but not logical
   8. Vector Clocks (53)
      1. Each processor/server has a clock (0, 0, …)
      2. For server i, update V[i] to take the MAX
      3. Concurrency & ordering (63)
      4. Causality violation portions (67)
2. Coordinator Election (Leader Election)
   1. Bully Algorithm (5)
      1. Elect with highest ID
      2. Different scenarios (18)
      3. Best case -> N-1 finds out, only O(n) (25)
      4. Worst case scenario -> Start with node 1, O(n^2) (25)
      5. Issues (29)
   2. Ring Election (30)
      1. Everyone is aware of entire ring structure
      2. Node with highest ID is the leader again
      3. Goes in a circle, if full cycle is detected, then start electing
      4. Best case -> 1 node detects -> O(n) in a ring (46)
      5. Worst case -> all starts simultaneously -> O(n^2)
   3. Invitation Algorithm (50)
      1. Protocol to form larger groups due to network partitions
      2. Partition comes from when communication between 2 groups are congested/cannot pass through, creating a virtual partition
      3. Looking for other partition coordinators and inviting to merge (60)
         1. Communicate to all participants to join the other group
      4. Advantages: Sub-groups can work concurrently (66)
      5. PROBLEM: No global consistency, but allows for more concurrency (67)
   4. Node Failure Issues (68)
3. Distributed Mutual Exclusion (DME) -> Mutex Lock in distributed systems
   1. Properties: Safety, Liveness, Fairness (6)
   2. Centralized lock server (11)
      1. Disadvantages (24)
   3. Ring-based algorithm (25)
      1. Unfair one just passes token and executes as soon it has the tokens
         1. Problem of not being sequential or fair
         2. Complexity analysis (31)
         3. Issues (41)
      2. Fair based one passes token in a round using time T (42)
         1. Wait for token to return with time T
         2. Actions for receiving tokens (47)
         3. Complexity analysis (56)
         4. Issues (69)
   4. Lamport Shared Priority Queue (70)
      1. Request to enter queue with time T
      2. Do actions & wait for ALL replies to enter CS
      3. Don’t send reply if have earlier own request/when to send reply? (98)
      4. Multiple Requests (82)
      5. 3(n-1) messages per request & exit
   5. Ricart & Agrawala (improved Lamport) (102)
      1. Only add requests that comes after its own
      2. Reduces data required & RELEASE message needed
      3. Complexity analysis (117)
      4. Caveats for R&A and Lamport (119)
   6. Voting Protocol (120)
      1. Vote & wait for majority to arrive (only can vote for 1)
      2. Has a local queue for voting order (next request that comes in) (135)
      3. Fix deadlock with rescinding of vote (141)
      4. Advantages (146)
   7. Maekawa Voting Protocol (improved voting) (147)
      1. Create a voting set
      2. Voting sets MUST overlap for EVERY machine
      3. Must get ALL votes
         1. Reason is cause only vote for 1, so there will be no safety issues
4. Consistency (x.y = y-th read at x-th machine)
   1. Naïve DSM (6)
   2. Strict Consistency (15)
      1. Follow global wall clock time
      2. Hard to implement due to global clock
      3. Exercise (21)
   3. Sequential Consistency (Ivy) (23)
      1. Order is specified and followed
         1. FIFO queue & requirements (30)
         2. Global ordering of ALL operations, no concurrency
         3. MUST follow order by processors, no going backwards
         4. Is a strict form of causal consistency
      2. Exercise (34)
   4. Causal Consistency (39)
      1. Looser-phrased sequential consistency (no global/total order)
      2. Can be loosely linked based on resource
         1. Order must still be according to processors order though
         2. But, if different processor, can be loosely linked
      3. Exercise (43)
      4. Differences between SEQ & CAUSE (44, 49)
      5. Has advantages over sequential (50)
   5. Total Store Order (TSO) (52)
      1. Somewhat follows global wall clock
      2. If read says a certain value, subsequent reads must follow that as well until a write is imposed and the next read value changes
      3. Exercises (55)
      4. Advantages (69)
   6. Eventual Consistency (71)
      1. Resolution of conflicts later, many benefits to this
      2. Comparison with Sequential (74)
   7. Ivy (87)
5. Fault Tolerance & Paxos (NOT BFT)
   1. Recoverability (Atomicity) (3)
   2. Replicated State Machine (4)
      1. Rules (4)
      2. Failure scenarios (who request/sends to which server + crashes) (8)
   3. FLP Theorem (17)
      1. Agreement
      2. Termination
      3. Integrity
   4. REGULAR Paxos (13)
      1. Overview (20)
      2. Roles (22)
      3. Naïve solution (23)
      4. Multiple acceptors (25)
      5. Challenges (28) *(Solutions are all the way below)*
         1. Race condition
         2. Non-termination
         3. Network partition
         4. Faults
         5. Faults + non-termination
      6. Paxos State & symbols (34)
      7. Protocols (35)
      8. Prepare stage (36)
         1. Rej if proposal < seen
         2. Reply if proposal > seen
      9. Accept stage (42)
         1. Sends after getting majority <prepare-ok>
         2. Choosing value (44, 45, 58, 59)
         3. Choose the highest frequency for accepted number & send <accept>
         4. Reject & reply works like prepare (52)
      10. Decide stage (54)
          1. Receive > 50% <accept-ok>
          2. This a broadcasting system without any replies
      11. Paxos Properties (56)
   5. Solutions to challenges earlier (60)
6. Byzantine Fault Tolerance (BFT) Part 1
   1. RSM recap (3)
   2. Types of faults (5)
   3. Byzantine Problem (9)
      1. Ideal condition (10)
      2. Proof that it’s impossible below 3f +1 (17)
   4. Paxos limitations with BFT (21)
   5. Byzantine failure cases (23)
      1. Withholding (23)
      2. Malicious message (25)
   6. Protocol & Basic idea (26)
      1. State of replica (28)
      2. Pre-prepare stage (30)
         1. Backup replicas sends out the prepare stage, not the main replica
      3. Prepare stage (36)
      4. Commit stage (42)
         1. Backup replicas received 2f + 1 prepare then commit & broadcast
         2. Everyone will reply to the client AFTER own execution
            1. Client waits for f + 1 replies
      5. Complexity = (n – 1) \* (n – 1) + (n – 1) = (n – 1) ^2
7. BFT Part 2
   1. View Change (2)
   2. Problem scenarios
      1. Not sending expected seq# (4)
      2. Multiple different requests (10)
      3. Message withholding (16)
         1. Client times out and broadcast request to everyone (18)
         2. Replicas will request from the primary for seq# (19)
   3. View change example & sequence diagram (11)
   4. Handling view change (13)
   5. Checkpointing (21)
      1. Saves memory & message size
   6. Correctness & liveness considerations (23)
   7. Other exercises
   8. Limitations (34)
8. Distributed File Systems (DFS) Part 1
   1. Designs of DFS (6)
      1. Operations (7)
   2. Naïve design – Centralized (8)
      1. Advantages (12)
      2. Disadvantages (13)
   3. Caching in DFS (15)
   4. Sun Network File System (NFS) (17)
      1. Uses caching
      2. Frequent synchronization & flushes changes on close (19)
      3. Stateless design to request (23)
      4. Deterministic operations (26)
      5. Consistency issues (29)
   5. CMU Andrew File System (AFS) (37)
      1. Close-to-open consistency (40)
      2. Cache invalidation callbacks (41)
      3. Advantages & disadvantages (43)
   6. NFS & AFS summary (43)
   7. Google File System (GFS) (44)
      1. Chunks & advantages (47)
      2. Overview (53)
      3. How it works (59)
      4. Synchronization (63)
      5. Communication types (64)
      6. Updates & consistency (69)
      7. Chunk selection & redundancy (72)
9. DFS Part 2
   1. GFS
      1. Selection of primary replica for chunks (2)
      2. Writing of data (4)
         1. Data is replicated in a chain (14)
         2. Responds to client (15)
      3. File spanning multiple chunks may face issues (15)
      4. Record append (17)
         1. Case 1 of not enough space (19)
         2. Case 2 of enough space (21)
      5. Record append issues (24)
      6. Dealing with issues (27)