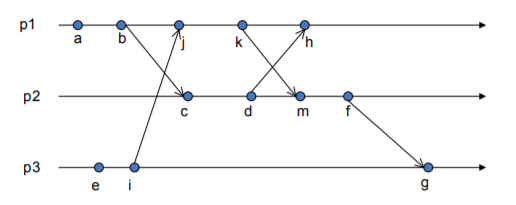
**1. [12 points (2+5+5)] This problem is about Events and Clocks. Consider the figure below:**

****

**a) Enumerate all pairs of concurrent events:**

a//e, a//i, b//e, b//i, c//eij, d//eijk, f//h, g//h, h//f, m//h

**b) Indicate vector clocks for each event. Assume that all logical clocks start initially with zero.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | P1 | P2 | P3 |
| A | 1 | 0 | 0 |
| B | 2 | 0 | 0 |
| C | 2 | 1 | 0 |
| D | 2 | 2 | 0 |
| E | 0 | 0 | 1 |
| F | 4 | 4 | 2 |
| G | 4 | 4 | 3 |
| H | 5 | 2 | 2 |
| I | 0 | 0 | 2 |
| J | 3 | 0 | 2 |
| K | 4 | 0 | 2 |
| M | 4 | 3 | 2 |

**c) Indicate matrix clocks for events i, d, k. Assume that all logical clocks start initially with zero.**

Index i:

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 2 |

Index d:

|  |  |  |
| --- | --- | --- |
| 2 | 0 | 0 |
| 2 | 2 | 0 |
| 0 | 0 | 0 |

Index k:

|  |  |  |
| --- | --- | --- |
| 4 | 0 | 2 |
| 0 | 0 | 0 |
| 0 | 0 | 2 |

**2. [10 points] This problem is about Christian’s and Berkeley algorithms. If the real-time clocks in a group of workstations can drift 15 seconds max per day, how frequently should the clocks be synchronized for each approach to keep them within 1 second of each other using a) Christian’s Algorithm and b) The Berkeley Algorithm?**

Both combined can max drift up to 30 seconds. Worst case scenario, computer goes 15secs fast and server goes 15secs slower

Using Christians: should be able to try do it every 24hours/15seconds for both the server and the workstations, which is going to be about half that amount: 2880 seconds. Berkeley’s should be about the same, we just ignore the response time.

**MP, 2 servers A,B.**

**A sent it at 16:34:13.430**

**B receives message at 16:34:23.480**

**B sends reply at 16:34:25.7**

**A receives reply at 16:34:25:725**

**3. [10 points] This problem is about Mutual Exclusion. Consider a group of distributed processes, P1, P2, P3, and P4 that share an object. They use the Maekawa algorithm for management of mutual exclusion. P1 is currently in the critical section (CS) and there is no other node in the wanted state. Now consider requests from P1, P4, P2 and P3 (in that order) to enter the same CS. Note: These requests are also received in this order (P4, P2, P3). Choose your own voting sets. Clearly state them. Show the state (as required by the algorithm, i.e. held, wanted, etc.) and queue entries at each processor.** P4, P2, P3

Sets State Queue

V1: {P1, P2, P3} Held {P4, P3

V2: {P2, P3, P4} Release/Voted {P4, P3

V3: {P3, P4, P1} Wanted {P2,

V4: {P4, P1, P2} Voted=true/Wanted {P3

**4. This problem is about Programming Assignment 1, the Chat Room Service. (a) When a client terminates a chat session, the server needs to free resources it used for this client. In socket programming, how does the server know if the client has closed the TCP connection?**

This is detected in the write function. Basically, the TCP connection will eventually detect a ton of timeouts from the client and declare that it has been disconnected by returning a -1.

**(b) When we design the server side code for handling TCP master sockets, two common approaches are a) multi-threaded and b) single-thread with select(). Comparing these two approaches, name one advantage and one disadvantage of using approach a). Assuming the server we are designing may have hundreds of simultaneous connections, which approach would you choose? Why?**

This is more related to the general advantages and disadvantages of threading. Multithreaded code can handle numerous requests at once, but will have to worry about thread overhead. It wasn’t necessary with our homework since we only had a few clients talking one at a time, but with hundreds of users, we need to implement multithreading to respond to requests quickly.

**5. This problem is about Programming Assignment 2, the Improved Chat Room Service (with gRPC) (a) What is the advantage of using RPC instead of socket programming?**

Sockets are locked into being read/write only, which is not good if we want distributed computing to act like centralized computing. It’s more scalable, plus easier for programmers since we don’t have to worry about socket ports/byte ordering etc.

**(b) In order to achieve real time chatting, consider these two approaches: a) push, i.e. whenever the client sends a message to the server, the server immediately forwards this message to all his/her followers; b) pull, i.e. when the client sends a message to the server, the server only stores it in a buffer, while the client’s followers need to periodically (e.g. every 1 second) pull new messages from the server. Consider a system where its users seldom send chats, which approach would you choose? Why?**

Comparing the two different methods, the first one is more responsive, but will need more resources in order to push messages to followers consistently. In a chatroom that is not so frequently used, using the second method will be better as it will strain our server less and just cause the participants don’t need consistent, immediate demand.

**6. Choose the statements below that are TRUE:**

**(A) In a Synchronous Distributed System each transmitted message is received within bounded time.** True

**(B) In an Asynchronous Distributed System there is no bound on clock drift rates.** True

**(C) Omission failures in a distributed system include process omission failures.** True

**(D) In a Byzantine failure model, nonexistent messages can not be delivered.** False, it nonexistent messages can be delivered

**(E) One false assumption about a distributed system is that the network topology changes.** False, the topology can change within a distributed system. This is a true assumption

**(F) In the OSI reference model, RPC is on Layers 4 and 5.** False, in layers 5 and 6: Session and presentation

**(G) Logical clocks keep track of event ordering.** True among related (casual) events

**(H) Christian’s algorithm for time synchronization assumes that network delays are asymmetric.** False delays are symmetric forwards and back

**(I) Berkeley’s algorithm for time synchronization obtains an average for all participating computers.** False, will filter out some outliers

**(J) The 1st stratum in NTP servers is directly connected to accurate time sources.** True

**(K) Messages in NTP bear timestamps of recent events.** True

**(L) Logical clocks encode causality relationships.** True

**(M) Two events a and b are concurrent if a does not happen before b, and b happens before a.** False, neither must not happen before each other

**(N) If a total order is needed for logical clocks, we can tie breaks using ids.** True

**(O) Logical clocks are total ordered.** True

**(P) Happens Before relationship is total ordered.** False, partially ordered

**(Q) An implementation of vector clocks requires vectors of length n, equal to the number of processors.** True

**(R) Matrix clock tells me what I know about what other people know.** True

**(S) The Centralized Algorithm for Mutual Exclusion satisfies ME1 and ME3, but not ME2.** False, ME1, ME2 but not ME3.

**(T) The Token Ring Algorithm for Mutual Exclusion satisfies ME1, and ME2, but not ME3.** True

**(U) Ricart and Agrawala Algorithm assumes that all processors have distinct numeric IDs and maintain logical clocks.** True

**(V) In Maekawa Algorithm, processes need only obtain permission to enter from subsets of their peers.** True

**(W) Deadlocks are possible in Maekawa Algorithm.** True

**(X) In the Bully Algorithm for Leader Election, a process sends and election message to all processes with higher IDs.** True

**(Y) In the Chang and Roberts Algorithm, if a process sends a message, it marks its state as a participant.** True

**(Z) In Paxos, Acceptors are processes that remember the state of the protocol.** True

**(AA) In Phase 1a of Paxos, a Proposer (leader) creates a proposal number N where N is greater than any previous proposal number used by this proposer.** True

**(AB) In Phase 2a of Paxos, if proposer receives enough promises it set a value v to the proposal, where v can be max(previous values).** False

**(AC) Zookeper is a highly available, scalable, multi-dimensional, distributed system.** False

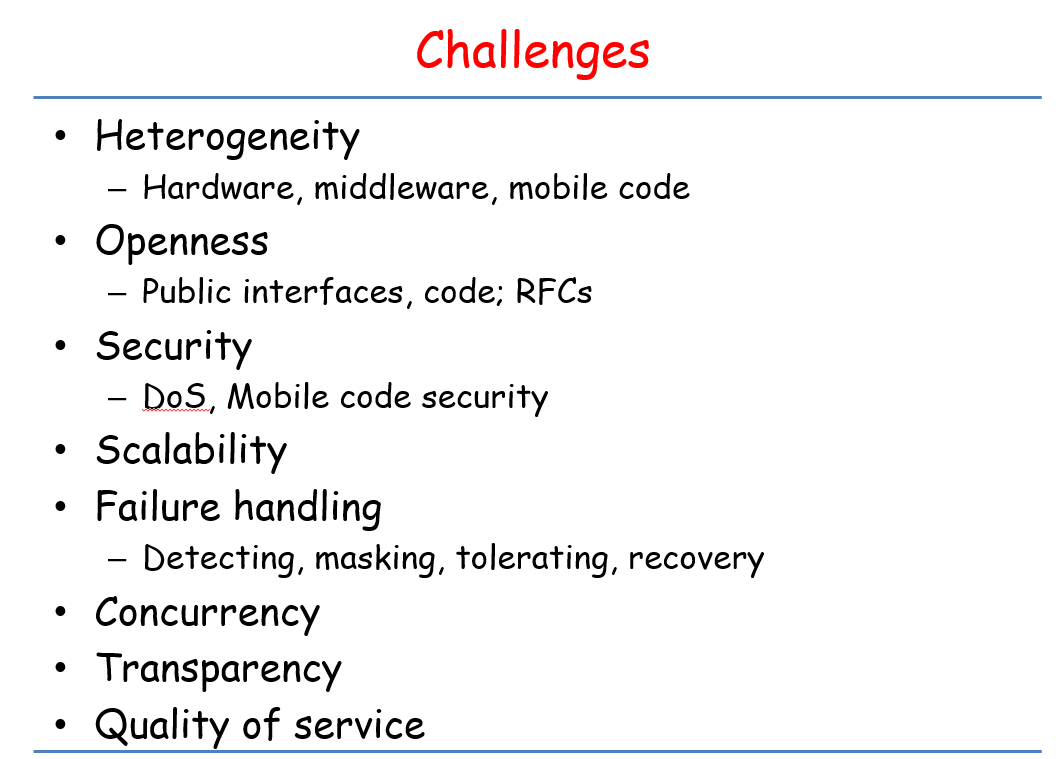
**(AD) Zookeper provides consensus, group membership, leader election and naming.** True

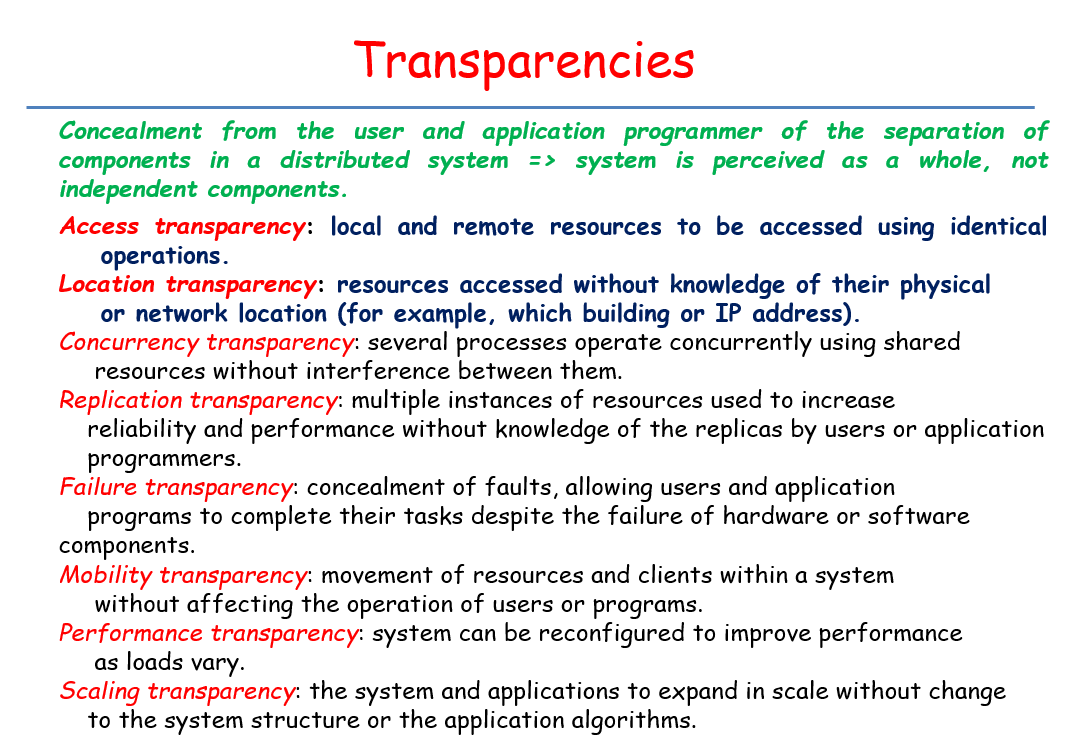
**(AE) Zookeper provides a file API without partial reads/writes.** True

**(AF) The hierarchical namespace in Zookeper contains znodes.** True

**(AG) A znode in Zookeper has data and no children.** False, has data and children

Lecture 1: Slides 16-17





Lecture 2: Mostly focus on the homework assignments

Lecture 3: 25 – 36

Lecture 4: Skip 44-70. 71-72 are game

Lecture 5: everything

Lecture 6: everything

Lecture 7: everything

Lecture 8: everything