Concurrent and Distributed Systems - 2019–2020

Supervision 4: Transparency, RPC, distributed objects, leadership, and replication

Q1 Transparency

Distributed system middleware provides a number of types of *transparency* to make it easier to program for and use distributed systems.

(a) Explain the difference between location and relocation transparency. How might *pure names* help to implement these?

(b) Give an example of a distributed service that might provide persistence transparency.

(c) Explain how could the lack of failure transparency makes it significantly more difficult to write distributed applications.

(d) Explain how performance transparency might (i) mask network delays and (ii) cope with a busy object server.

Q2 RPC and retry semantics

(a) Write an XDR interface for a remote bank service that supports creating accounts, getting the balance of an account, withdrawing, and depositing.

(b) RPC requests and/or replies might get lost by the network if an unreliable transport protocol, such as UDP, is used. Moreover, the remote machine hosting the service may crash before it is able to send a reply. Both of these scenarios would require the client to retry sending the RPC request. Briefly describe how one might achieve each of the following *retry semantics* in these scenarios:

- All-or-nothing
- At-most-once
- At-least-once

(c) For each of the remote bank RPC operations, which retry semantic would be required for correct execution?

Q3 Distributed objects

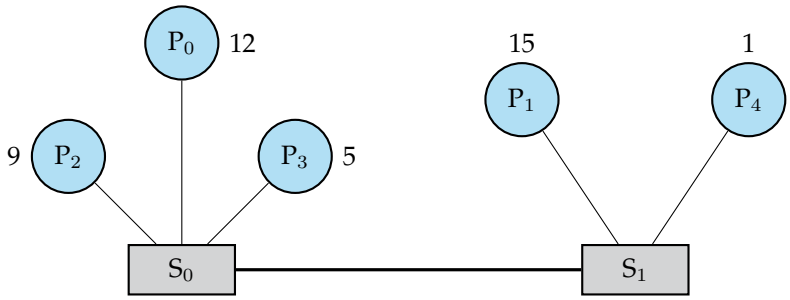
(a) Provide pseudocode for a Java RMI version of the remote bank service in question 2 that comprises Account and Bank objects.

(b) Extend the Bank class with a remote method that takes two Account instances and atomically transfers money from one to the other. What synchronisation will you require in the Bank class to ensure atomicity? How can distributed deadlock be avoided?

(c) Explain the difference between RMI *remote objects* and RMI *serialisable objects*. When might each be used?

Q4 Leadership election

(a) In the example system below, explain how the *Bully Algorithm* would be used to elect a leader:



(b) Suppose a new process P_5 entered the system with ID 20 and was connected to switch S_1; how would this effect leadership election?

(c) What might happen in the original system in part (a), if the link between switches S_0 and S_1 went down?

Q5 Replication

(a) Give two reasons why replication might be used in a distributed system.

(b) A major challenge when keeping multiple replicas of data is ensuring the replicas remain *consistent*. Briefly describe the following types of consistency, giving one advantage and one disadvantage for each:

- Strong consistency
- Weak consistency
- FIFO consistency
- Eventual consistency

(c) *Quorums* allow strong consistency to be maintained while avoiding the need for all servers to participate in all transactions. *Sloppy quorums* decay those requirements to improve failure tolerance at the cost of consistency. In a 5-server system, what true quorum parameters offer greatest read availability and throughput – at a cost to writes? In the same system, what sloppy quorum parameters offer greatest write availability and throughput – at cost to consistency?

(d) The Domain Name System (DNS) comprises a very large distributed database that is spread out geographically across many servers. Research the DNS protocol: how does it use replication, and why? What failure tolerance and consistency semantics does DNS offer?