Questions

Explain why middleware based distributed systems are built above network operating systems, not distributed operating systems

Each node in a distributed operating system (DOS) share the same operating system; however, in a network operating system, nodes are heterogeneous (run different operating systems). Middleware provides a form of abstracting the personal operating system from the different nodes, this would only be necessary in a system with heterogeneous operating systems.

Describe the differences between network operating systems, distributed operating systems and middleware based distributed systems.

A network operating system is a loosely-coupled operating system for heterogeneous multicomputers. A distributed operating system is a tightly coupled operating system for multiprocessors and homogenous multicomputers. Middleware based distribution systems are network operating systems with an additional layer implementing general purpose services.

A process in a distributed system runs on one node and accesses data from another node. After some time, for load balancing purposes, this process relocates to a different node. What kind of transparencies should be provided for this process in a distributed system?

Access (if the new node has a different data representation, it doesn't matter), location (it should not matter where the new node is), migration (user should not worry on whether or not it will move), and relocation (user does not need to know that it has changed location).

Definitions of the above answers: Access: Hide differences in data representation and how a resource is accessed Location: Hide where a resource is located Migration: Hide that a resource may move to another location Relocation: Hide that a resource may be moved to another location while in use Extra definitions that were possible answers: Replication: Hide that a resource has been replicated Concurrency: Hide that a resource may be shared by several competitive users Failure: Hide the failure and recovery of a resource

Why is transparency an important property that a distributed systems designer should achieve?

Same as abstraction for developers, it allows the mechanisms hidden to change without requiring redesigning every time on the client side.

Give one good reason why it may be desirable for a system NOT to provide complete transparency.

Attempting to maintain transparency in the event of a server failure can result in a slowdown to the system as a whole. Therefore, attempting to maintain transparency in systems where performance is critical is not a good idea.

Is a distributed algorithm more fault tolerant than a centralized algorithm? Provide an example

Yes. A distributed algorithm does not fail if a single machine fails, and it does not assume there is a global clock. Also, no machine has complete information about the system state, and machines make decisions based only on their local information.

An example of a centralized algorithm failing where a distributed one would not would be an international banking network that sends all banking requests to a single machine, and distributes the results of each transaction to a collection of banks around the world. The failure of a network link to the central server, or indeed any failure of the central server, would result in the distributed system failing altogether.

Is the name “www.uq.edu.au/index.html” location independent? Explain what a “location independent” name is.

Yes. Location independence allows the underlying address to change without requiring the name to change.

In a structure overlay network messages are routed according to the topology of the overlay. What is an important disadvantage of this approach?

The overlay is logical, and may not reflect the optimal physical layout (i.e. message transfer time) for routing messages.

Routing tables in current application level messaging systems are configured manually. Describe a protocol which would provide automatic configuration of the routing tables?

An idea is to use a decentralised routing algorithm by having each node within the network discover the network topology, and calculate its own best routes to other nodes in the network.

What are the main differences between Remote Procedure Calls and Remote Method Invocations?

With RPC you just make calls to a remote server requesting to execute a function; withRMI,
you can have references to remote objects and invoke their methods. Also, in RMI you'reable
to pass object references through the system, including both local and remote object references.

Give two examples of forward recovery and two examples of backward recovery.

Forward recovery is the process of constructing lost data using the available data. Examples include checksums and Redundant Array of Independent Disks (RAID). Backwards recovery is the process of reverting the system state to the correct state, and continue processing new data. Backwards recovery typically results in the pre-failure operation being repeated. Examples include the retransmission of a packet, or a failed database update.

Is the recovery of faults in RAID (Redundant Array of Independent Disks) considered to be backward recovery or forward recovery? Explain why.

Forward recovery, because it is using data that it is still available to recreate the lost data. In RAID specifically, it uses another disk.