1. Alguns aspectos do sistema distribuído devem ser transparentes para o utilizador. Apresente as diversas formas de transparência desejáveis num sistema distribuído.

Uma das preocupações em Sistema Distribuído é a transparência que mantem alguns aspectos da distribuição invisíveis para o utilizador. As diversas formas de transparência desejáveis num Sistema Distribuído são o acesso que permite o acesso a recursos locais e remotos com operações idênticas, a localização que permite a utilização de recursos sem o conhecimento da sua localização exacta, a concorrência que permite a execução simultânea de vários processos com recursos partilhados sem que haja interferências entre eles, a replicação que utiliza múltiplas instâncias de recursos para aumentar a fiabilidade e a performance, mas sem que o utilizador ou as aplicações cliente tenham conhecimento das réplicas, as falhas que permite o tratamento de falhas, para que o utilizador e as aplicações completem a sua tarefa, independentemente da ocorrência de um problema de hardware ou software, a mobilidade que permite a mobilidade de recursos e clientes dentro de um sistema sem afectar as operações de utilizadores e programas, a performance que permite a reconfiguração do sistema para aumentar o desempenho à medida que as cargas variam e a escala que permite a expansão das componentes do sistema sem alterar a estrutura do mesmo.

2. Distinga Modelo Cliente-Servidor de Modelo Peer Processes.

No modelo Cliente-Servidor, os clientes invocam um servidor mas o servidor também pode assumir o papel de cliente para ligar-se a outro servidor enquanto no modelo Peer Processes é baseado em peer processes (processos que cooperam e comunicam de forma simétrica para realizar uma tarefa) em que os processos desempenham papéis idênticos, sem uma separação prévia entre clientes e servidores. Pontualmente, alguns destes processos podem comportar-se como clientes ou como servidores.

3. Descreva o comportamento das operações send e receive (na transmissão de uma mensagem), relativamente ao bloqueio, em cenários de comunicação síncrona e assíncrona.

Na comunicação síncrona os processos emissor e receptor sincronizam-se a cada mensagem, onde o send e o receive são operações bloqueantes. O emissor fica parado no send até que o receive seja efectuado, e ao efectuar um receive, o receptor fica bloqueado até a mensagem chegar enquanto que na comunicação assíncrona não há sincronização, ou seja, o send não é bloqueante visto que o emissor prossegue assim que a mensagem passa ao buffer local de saída e o receive pode ser bloqueante ou não, visto que o processo prossegue com um buffer que será preenchido em background, havendo uma notificação quando isso acontecer.

4. Explique a diferença entre sistemas síncronos e assíncronos.

Nos sistemas síncronos existem limites para o tempo de execução de cada passo de um processo, para o tempo até à recepção de uma mensagem enviada e para o clock drift rate em cada máquina enquanto nos sistemas assíncronos não há limite definido ou garantias para a velocidade de execução de um processo, para o tempo de transmissão de uma mensagem, esta pode ter atraso (delay) nem para o clock drift rate, visto que a taxa deste é arbitrária.

5. Descreva o processo designado por Marshalling e justifique a sua utilização.

O processo de Marshalling consiste na tradução das estruturas e tipos primitivos para uma representação externa de dados (formato usado para a representação de estruturas e tipos primitivos) adequada para a sua transmissão.

O processo de Marshalling é utilizado para que uma estrutura de dados possa ser usada em RPC ou RMI, ou seja, para que possa ser representado de modo flattened e os tipos primitivos num formato acordado.

7. Indique uma diferença entre Marshalling em CORBA e o usado em Java RMI.

No Marshalling em CORBA assume-se que o emissor e receptor conhecem os tipos de cada elemento da mensagem, por isso o tipo não é passado (apenas o valor) enquanto que em Java

RMI, a aplicação que recebe a mensagem pode não conhecer o tipo de dados. Então, a representação serializada inclui informação sobre a classe do objecto

8. Descreva as vantagens fornecidas pela camada Middleware e indique algumas abstracções que disponibiliza ao programador de sistemas distribuídos.
As vantagens fornecidas pela camada Middleware são:
- Transparência face à localização;
- Independência dos protocolos de comunicação;
- Independência do Hardware;
- Independência dos Sistemas Operativos;
- Utilização de diferentes linguagens de programação.
Algumas abstracções que estão disponíveis para o programador são a abstracção RPC e a abstracção RMI. A RPC descreve os procedimentos disponíveis e respectivos argumentos e não se podem passar pointers como argumentos e a RMI tem métodos de um objecto disponíveis para Invocação Remota e podem passar-se referências para objectos remotos.

9. Existem três tipos de medidas de tolerância a falhas desejáveis para a primitiva doOperation, no protocolo Request-Reply.
a) Enumere e explique em que consistem essas medidas.
As medidas de tolerância da primitiva doOperation no protocolo Request-Reply são o reenvio do pedido que reenvia a mensagem com o pedido para o servidor até a resposta chegar ou se detectar que o servidor está com problemas, a filtragem de duplicados que decide se o duplicado deve ser processado para reenvio ou ignorado e a retransmissão de resultados que através de um histórico de resultados evita uma nova execução da operação.

b) Indique quais são usadas pelas semânticas de invocação Maybe, At-least-once e At-most-once.
A Maybe não reenvia o pedido, a filtragem de duplicados e a retransmissão de resultados não são aplicáveis.
O At-least-once reenvia o pedido, não filtra os duplicados e executa de novo a retransmissão de resultados.
O At-most-once reenvia o pedido, filtra os duplicados e retransmite do histórico os resultados.

c) Das semânticas de invocação referidas, indique qual a mais apropriada quando estão envolvidas operações não idempotentes.
Quando estão envolvidas operações idempotentes, a semântica de invocação mais apropriada é o Maybe.

10. Imagine um sistema baseado em diferentes linguagens de programação e plataformas.
a) Indique um mecanismo para invocação remota de métodos adequado a este cenário.
Um mecanismo para invocação remota adequado a este cenário é o CORBA. Que permite que objectos distribuídos e implementados em diferentes linguagens de programação possam comunicar.

b) Ainda neste cenário, seria necessária uma Interface Definition Language? Porquê?
Neste cenário é necessário uma IDL pois requer uma IDL que forneça a notação para as interfaces que poderão ser usadas pelas diferentes aplicações.

11. Em Java RMI, que operações devem ser obrigatoriamente efectuadas pela aplicação Servidor, para que um cliente possa invocar métodos remotamente sobre o objecto remoto que presta um serviço?
As operações que devem ser obrigatoriamente efectuadas pela aplicação Servidor é ter as classes dispatcher e skeleton, ter as classes com implementação para todos os objectos remotos que podem ser a Servant ou Impl, tem-se de criar pelo menos um objecto remoto e inicializá-lo,

registar o objecto no binder e para evitar demoras, cada invocação remota é tratada numa nova thread.

12. Descreva o papel de Proxy, Dispatcher e Skeleton na abstracção RMI.

O papel do Proxy torna a invocação remota transparente para o cliente, faz o marshalling de argumentos e o unmarshalling do resultado da invocação, é unico para cada objecto remoto que um processo referencie e implementa os métodos da interface remota do objecto, mas cada método faz marshall da referência do objecto, methodid, e argumentos, aguardando a resposta para o unmarshall.

O Dispatcher é unico para cada classe de objecto remoto, no servidor. Recebe a mensagem e pelo methodId selecciona o método apropriado no Skeleton. O Skeleton é um por cada classe que representa um objecto remoto, no servidor e implementa os métodos na interface remota, mas efectuando unmarshall a argumentos no pedido, invocando o método no objecto remoto (localmente) e devolvendo o marshall do resultado e eventual excepção na resposta ao proxy.

13. Distinga os modos de sincronização interna e externa de relogios num sistema.

Relogios sincronizados de modo interno não estão necessariamente sincronizados de modo externo, ou seja, se cada nó de um sistema está sincronizado de modo externo (com a mesma fonte) com limite D, então esse sistema está internamente sincronizado com um limite 2D.

14. Descreva sucintamente o algoritmo de sincronização de Cristian.

O algoritmo de sincronização de Cristian sincroniza se o tempo para a troca de mensagens cliente-servidor é suficientemente pequeno quando comparado com a precisão desejada e o servidor de tempo é UTC S.

O processo p envia pedido mr e recebe um tempo t em mt, p regista o tempo de viagem de mr e mt onde T = Tmr + Tmt. Daqui estima-se o tempo em p com t + T/2.

15. Descreva sucintamente o algoritmo de sincronização de Berkeley.

O algoritmo de Berkeley é para sincronização interna de um grupo de computadores, onde uma máquina é escolhida para coordenar (master). O master pede periodicamente uma leitura aos restantes (slaves) e estima a hora em cada slave, pela observação do tempo de viagem das mensagens e pelo valor recebido (como em Cristian) e faz a média de todos os valores (incluindo o seu próprio tempo). O master em vez de enviar o tempo actualizado aos slaves (o que estaria sujeito ao tempo de envio variável), o master envia a cada um o valor exacto que deve usar para ajustar o seu relógio. Se o master falhar, outro será escolhido para assumir a sua função.

16. A distribuição dos objectos requer cuidados no Garbage Collection, especialmente do lado do servidor.

a) Explique resumidamente o algoritmo de GC no Oject Model Distribuído.

A cada servidor mantém uma lista com o conjunto de processos com referências para os seus objectos e quando um cliente cria um Proxy para um objecto, é adicionado ao conjunto de processos com referências para aquele objecto. Quando o Garbage Collection do cliente detecta que o Proxy do objecto já não é necessário, este envia uma mensagem ao servidor e elimina o proxy, onde o servidor remove o processo da lista e quando a lista estiver vazia, o Garbage Collection do servidor recupera o espaço do objecto, excepto se existirem referências locais.

b) Identifique eventuais vulnerabilidades do algortimo face a falhas no cliente.

Eventuais vulnerabilidades do algoritmo são que o servidor (com objectos remotos) atribui um intervalo de tempo ao cliente e a contagem é válida até que o tempo expire ou o que cliente remova a referência do objecto.

17.O protocolo NTP permite três modos para sincronização de relógios em servidores.

a) Descreva esses métodos de sincronização.

Os métodos de sincronização sao o multicast que usa-se em redes locais de alta velocidade, onde um ou mais servidores enviam periodicamente o tempo num broadcast e os servidores noutras máquinas acertam o relógio assumindo um pequeno delay. O multicast é de baixa precisão. Outro método é o procedure-call onde um servidor aceita pedidos de outros computadores, aos quais responde com a informação horária que tem. Este é utilizado quando se pretende maior precisão que no modo multicast, ou simplesmente não é possível multicast. O terceiro é o modo simétrico que serve para sincronizações entre servidores que fornecem a informação em redes locais e em níveis mais altos da NTP subnet, onde se pretende a máxima precisão. Aqui um par de servidores trabalha de modo simétrico, troca mensagens com informação horária e o tempo das mensagens também é considerado.

b) Qual o método onde se consegue a precisão máxima?

O método onde se consegue a precisão máxima é no modo simétrico.