1. **Compare os seguintes conceitos mostrando vantagens e desvantagens:**
2. **Broadcasting e servidor de nomes.**

Transparência de localização, o usuário não sabe sobre a localização de um recurso. Na arquitetura cliente servidor essa transparência é implementada sobre duas formas: broadcast e servidor de nomes.

Na abordagem de Broadcasting um processo que deseje se comunicar com outro envia sua mensagem para todos os nós na rede, ou seja, não há necessidade de saber o endereço específico do receptor. A vantagem está que a implementação de transparência de localização é feita, além disso não apresenta um único ponto de falha. A desvantagem está no fato de que só atinge nós em uma rede local.

Na abordagem do servidor de nomes, os nós da rede se “cadastram” em uma máquina chamada servidor de nomes e para acessar outras máquinas os nós “perguntam” ao servidor de nomes o endereço da máquina de destino. A desvantagem está no fato de que o servidor de nomes representa um ponto de falha, se ele cair a transparência cai. A vantagem é que funciona para redes locais e remotas.

1. **Servidor de arquivos**, um servidor que guarda arquivos para os clientes interessados. Ele apenas repassa o arquivo e não os manipula. Desvantagem é que todo o gerenciamento fica para o cliente, além isso o arquivo deve ser carregado completamente para a máquina cliente para ser manipulado e o mesmo arquivo não pode ser manipulado por múltiplos clientes. Vantagem é que foi historicamente a primeira implementação de um serviço desse tipo.

SGBD, um banco de dados que permite consultas. As consultas “CRUD” são implementadas e podem ser utilizadas pelos clientes. Vantagem, mais de um ciente pode utilizar o banco, desvantagem muitas consultas em um ambiente de rede ainda não tem um bom desempenho, além disso quando a complexidade do banco é grande demais fica difícil de gerenciar.

1. **Explique, detalhadamente, como os mecanismos de Servidor de nomes e RPC fornecem as transparências necessárias para a construção de um sistema distribuído mínimo.**

Sabendo primeiramente que para construir um sistema distribuído mínimo é necessário haver as transparências de localização e acesso.

A arquitetura C/S utiliza o servidor de nomes e o endereçamento para implementar naturalmente a transparência de localização. O servidor de nomes concentra os endereços necessários para dois pontos se comunicarem, assim se um processo A precisa se comunicar com o processo B ele faz apenas uma consulta ao servidor de nomes.

Para implementar a transparência de acesso a arquitetura C/S faz uso do RPC, ou Remote Procedure Call. Uma técnica que, por meio da implementação de métodos stubs, empacotam chamadas remotas em chamadas locais. Assim, um cliente que chama um método de um stub faz essa chamada local, o stub trata da comunicação com o servidor remoto, o stub no lado do servidor apresenta a requisição ao servidor como se fosse local e este responde como se fosse local, a resposta é tratada pelo RPC passando pela rede e o cliente recebe a resposta como se viesse local. Assim, todo o processo é parece ser local, embora seja remoto, implementando assim a transparência de acesso.

1. **Fale sobre a evolução dos sistemas distribuídos sob a perspectiva dos paradigmas de programação estruturado e orientado a objetos.**

Quando olhamos para o desenvolvimento, primeiramente dos sistemas computacionais vemos que desde a época dos mainframes o paradigma estruturado era predominante. Um dos primeiros modos de conseguir algo parecido com distribuição, o Servidor de Arquivos, utilizava esse paradigma como a premissa. A evolução do servidor d arquivos para SGBD e depois para um SGBD distribuído sempre balanceava a pergunta, onde colocar dados e onde colocar processos. No caso do paradigma estruturado dados vinham separados de processos, o servidor de arquivos por exemplo o cliente processava os dados que estavam disponíveis no servidor, no SGBD o cliente acessava os dados disponíveis no banco de dados. Essa separação sempre existente entre dados e processo sofreu uma modificação com o advento do paradigma orientado a objeto. Nesse paradigma, a unidade de distribuição (objeto) é uma instancia de uma classe, essa classe utilizando o conceito de encapsulamento traz em si dados (atributos) e processos(métodos) em um mesmo lugar. Vale ressaltar que objetos são unidades de memória e em muitos casos precisam ser persistidos, levantando novamente o questionamento de onde colocar os dados. Além disso, sob a luz da orientação a objeto o conceito de herança em um ambiente distribuído pode ser prejudicial visto que certos dados (variáveis de classe) são compartilhados or todas as instancias de uma classe, inclusive aquelas que herdaram de uma outra classe e a modificação desses dados precisaria ser feitas em todas essas instâncias. Consuma-se dizer que herança não combina com distribuição, um modo que foi encontrado para resolver isso foi fazer a distribuição não de objetos, mas de ramos de herança (os componentes) garantido que essas classes não teriam problema com a distribuição.

1. **Qual a relação da taxonomia de Flynn com o conceito de sistemas distribuídos?**

A taxonomia de Flynn versa sobre dois tipos de sistemas. Um sistema onde vários processadores compartilham um mesmo banco de memória que por conta dessa dependência de um certo recurso são intitulados sistemas fortemente acoplados. O outro tipo de sistema são aqueles onde várias CPUS estão trabalhando em conjunto e cada CPU tem sua unidade de memória, esses são os sistemas fracamente acoplados. O conceito de sistemas distribuídos fala sobre um conjunto de computadores autônomos e independes que se conectam e parecem para o usuário como uma única máquina. Observando a taxonomia de Flynn podemos ver que o melhor candidato para isso seria um sistema fracamente acoplado que possui várias unidades independentes e autônomas e que por estarem na rede podem ser mascaradas para se parecerem com apenas 1 máquina. Vale a pena ressaltar também que na abordagem fracamente acopladas se uma máquina parar de funcionar o sistema não cai, mas na abordagem fortemente acoplada se o recurso dependente (no caso a memória) parar de funcionar todo o sistema será prejudicado.