1) Recursos de hardware:

- CPU: servidor de computação (executa aplicativos com uso intensivo de processador para clientes), servidor de objeto remoto (executa métodos em nome de clientes), programa worm (compartilha a capacidade de CPU da máquina desktop com o usuário local).

- memória: servidor de cache (mantém as páginas da web acessadas recentemente em sua RAM, para acesso mais rápido por outros computadores locais), memoria compartilhada

- disco: servidor de arquivos, servidor de disco virtual

- impressora: as impressoras em rede aceitam trabalhos de impressão de muitos computadores. Gerenciá-los com um sistema de filas.

- capacidade da rede: a transmissão de pacotes permite que muitos canais de comunicação simultâneos (fluxos de dados) sejam transmitidos nos mesmos circuitos.

Dados / software:

- página da web: os servidores da web permitem que vários clientes compartilhem o conteúdo da página somente leitura (geralmente armazenado em um arquivo, mas às vezes gerado instantaneamente).

- arquivo: os servidores de arquivos permitem que vários clientes compartilhem arquivos de leitura e gravação.

- banco de dados: os bancos de dados destinam-se a registrar o estado definitivo de alguns conjuntos de dados relacionados.

- conteúdo do grupo de notícias: o sistema netnews disponibiliza para os clientes na Internet cópias somente leitura dos itens de notícias postados recentemente. Uma cópia do conteúdo do grupo de notícias é mantida em cada netnewsserver que é uma réplica aproximada daqueles em outros servidores. Cada servidor disponibiliza seus dados para vários clientes.

- stream de vídeo / áudio: os servidores podem armazenar vídeos inteiros em disco e entregá-los na velocidade de reprodução para vários clientes simultaneamente.

2)Nao tem a questao de replicação para outros servidores, consertar problemas no servidor fica mais facil, garantia de maior desempenho, total controle para suas configurações. Os problemas é em questao da escalabilidade de localidade e administrativa porque em questao de localidade que se um cliente(jogador) estiver jogando o jogo e estiver muito longe, ele pode ter problemas de comunicaçao com o servidor, consequentemente, latencia, travando. Em questao administrativo, seria em questao de recursos e construçao de um servidor enorme em um so lugar, podendo necessitar de um lote grande e de muito dinheiro para construçao, sem falar que com a perda do servidor ninguem consegue jogar

3) A novidade da computação em nuvem que você não necessita mais instalar programas ou recursos pela internet, você pode acessar tudo de forma remoto, via online, de qualquer lugar do mundo com qualquer aparelho, um dos maiores exemplos é a memoria, você não necessita mais

comprar HD ou mais memoria, você pode colocar todos os arquivos e dados de forma remoto que estara salvo la

4) HTML é uma linguagem relativamente simples de analisar e renderizar, mas confunde a apresentação com os dados subjacentes que estão sendo apresentados.

Os URLs são localizadores de recursos eficientes, mas não são suficientemente ricos como links de recursos. Por exemplo, eles podem apontar para um recurso que foi realocado ou destruído; sua granularidade (um recurso inteiro) é granular demais para muitos propósitos.

HTTP é um protocolo simples que pode ser implementado com uma pegada pequena e que pode ser usado em muitos tipos de transferência de conteúdo e outros tipos de serviço. Seu detalhamento (mensagens HTML tendem a conter muitas strings) o torna ineficiente para passar pequenas quantidades de dados.

HTTP e URLs são aceitáveis como base para computação cliente-servidor, exceto que: não há verificação de tipagem forte e há a ineficiência que mencionamos nas desvantagens deles.

5) Como os computadores estão conectados a uma Internet, podemos supor que os protocolos da Internet lidam com diferenças entre as redes. Mas os computadores podem ter hardware diferente - portanto, temos que lidar com diferenças de representação de itens de dados em mensagens de solicitação e resposta de clientes a objetos. Um padrão comum será definido para cada tipo de item de dados que deve ser transmitido entre o objeto e seus clientes. Os computadores podem rodar diferentes sistemas operacionais, portanto, precisamos lidar com diferentes operações para enviar e receber mensagens ou para expressar invocações. Assim, no nível Java/C++, uma operação comum seria usada, a qual será traduzida para a operação específica de acordo com o sistema operacional em que é executada. Temos duas linguagens de programação diferentes C++ e Java, elas usam representações diferentes para estruturas de dados, como strings, arrays, registros. Um padrão comum será definido para cada tipo de estrutura de dados que deve ser transmitida entre o objeto e seus clientes e uma forma de tradução entre essa estrutura de dados e cada uma das linguagens. Eles precisarão concordar com os padrões comuns mencionados acima e documentá-los.

6) Para torna-lo aberto, ele precisa seguir alguns padroes como:
- Sistemas deveriam concordar com interfaces bem definidas
- Sistemas deveriam suportar portabilidade de aplicaçoes
- Sistemas deveriam interoperar com facilidade
- Sistemas deveriam ser facilmente extensíveis

Já quando tratamos de heterogeneidade, alem dessas caracteristicas anteriores podemos colocar também:
- Foco na segurança da aplicação
- Escalabilidade tem que ser considerada
- Desempenho
- Comunicaçao entre componentes
- Tolerancia a falhas

Entao basicamente a abertura e suporte a heterogeneidade não são a mesma coisa, enquanto a abertura envolve mais a conexao, formato de dados, representaçao de dados, a heterogeneidade foca essas caracteristicas mais o desempenho por tras

7) Encriptação - Uso de criptografia para troca de mensagens e armazenamento de informações sensíveis

Autenticação - Para acesso a um recurso
Autorização - Para executar ações em um recurso
Auditoria - Registro de atividades realizadas em logs

8) As falhas são:
1- O cliente não consegue localizar o servidor
2- A mensagem de requisição do cliente para o servidor se perde
3- O servidor cai após receber uma requisição
4- A mensagem de resposta do servidor para o cliente se perde
5- O cliente cai após enviar uma requisição

Solução:
1- Criar uma exceção caso isso aconteça
2- O cliente tem um temporizador, se o temporizador expirar antes do recebimento da resposta de reconhecimento, a mensagem é reenviada
3- O cliente tem um temporizador esperando pela resposta, caso expire ele reenvia uma requisição
4- O sistema deve informar a falha ao cliente
5- Esperar pelo retorno do cliente

9) Os problemas são:
- Atraso ou latencia no recebimento das respostas porque são muitas requisições para um servidor único
- Problemas de conexão entre um cliente e o servidor
- Desgaste de CPU e memoria
- Gargalo

Solução:
- Criando mais servidores dedicados a esse serviço
- Aumentar a eficiência do servidor
- Melhore a conectividade da rede

10) Seria porque ele além de estar replicando para todas as maquinas, estamos enviando as requisições dos clientes para todos os servidores

11) As URLs garantem transparência de localização porque mesmo se referindo ao endereço de rede no qual se encontra algum recurso informático, elas em nenhum momento indicam a localização de um recurso na URL, basicamente os nomes lógicos não indicam a localização de um recurso.