Universidade Federal de Ouro Preto

Instituto de Ciências Exatas e Aplicadas

Departamento de Computação e Sistemas

Sistemas Distribuídos:

Lista II

Icaro Bicalho Quintão – 14.1.8083

João Monlevade

2018

1) O protocolo de requisição-resposta tem como propriedade mascarar a heterogeneidade dos sistemas pois ele configura apenas padrões para as requisições e para as respostas, dessa forma, eles podem ser usados em diversos tipos de redes.

2) a) Idempotente.

b) Não idempotente.

c) Idempotente.

Sim, pois temos a certeza de que ela não mudará seu valor caso seja executada mais de uma vez.

3) Se a sobrecarga da rede não for um problema dentro da rede que esse servidor executará, o uso de protocolo TCP na comunicação fará com que esse servidor não precise armazenar histórico nenhum de respostas, pois o protocolo já cobre filtro de duplicatas e estabelecimento de conexão para evitar perda de pacotes tanto da requisição quanto da resposta. Quando se usa UDP, servidores rodando sob protocolos RR devem manter um histórico da ultima resposta enviada a cada cliente, pois não se tem garantia que o cliente a receberá, a garantia vai vir quando o cliente fizer uma nova requisição, ai a resposta antiga pode ser apagada e substituída pela nova. Por outro lado, quando o servidor implementa um protocolo RRA, não é necessário manter um histórico de respostas, pois assim como o TCP, há um aviso de que a resposta chegou ao cliente de forma correta.

4) Porque tanto as mensagens de requisição quanto as de respostas costumam ser de tamanho pequeno, dessa forma, quando se usa TCP por exemplo, os 2 pares extras de mensagens para estabelecimento da conexão acabam por ser mais volumosas do que o devido par de comunicação, o que causa uma sobrecarga na rede e, consequentemente, uma diminuição do desempenho total do sistema.

5) A própria ideia da comunicação indireta já prega um desacoplamento temporal e espacial por definição. Quando tratamos de sistemas voláteis o uso dela passa a ser de certa forma padrão, pois dispositivos podem entrar e sair da rede com uma velocidade muito alta e, dessa forma, o uso de uma comunicação direta pode ser inviável, isso porque a comunicação indireta vai garantir que as mensagens serão enviadas, mesmo

que o remetente ou o destinatário não estejam conectados na rede naquele exato momento, dessa forma, a comunicação indireta se torna mais apropriada para esse tipo de sistema.

6) É possível sim a construção de um paradigma com essas características, o serviço de e-mail pode ser incluído nessa categoria, pois trata de um serviço no qual o remetente e o destinatário são acoplados (de uma forma ou de outra conhecem um ao outro), mas temporalmente são totalmente desacoplados pois um não depende do tempo de vida do outro, há algum serviço intermediário capaz de receber e encaminhar as mensagens no momento em que o receptor entra na rede.

7) Se uma comunicação é síncrona, as entidades participantes são acopladas temporalmente, porém, a comunicação ser assíncrona não faz com que o sistema seja totalmente desacoplado no tempo. O acoplamento (ou desacoplamento) temporal diz muito mais sobre o tempo de vida dos processos do que sobre como a comunicação se dará, é possível se ter um sistema que o cliente só consegue se conectar a rede enquanto um servidor está ligado (cliente e servidor são, portanto, acoplados no tempo) e, ainda sim, a comunicação entre eles ser assíncrona, talvez com o uso de uma entidade intermediária entre eles que vai lidar com a comunicação por exemplo.

8) Filas de mensagens distribuídas facilitam a comunicação dentro do sistema distribuído, evitando a sobrecarga da rede e permitindo uma comunicação mais confiável. Uma entidade do sistema fica responsável por gerenciar essa fila, todas as mensagens enviadas pela rede vão para essa fila e essa entidade reencaminha para o receptor correto. Isso permite uma transparência muito grande, uma vez que os processos comunicantes não precisam necessariamente conhecer uns aos outros, o que importa é que a entidade gerenciadora da fila saiba a origem e o destino de cada mensagem.

9)1) O cliente que acessa um serviço inclui um procedimento stub para cada procedimento da interface de serviço.

2) O stub se comporta como um procedimento local para o cliente, mas em vez de executar a chamada, ele empacota o identificador de procedimento e os argumentos em uma mensagem de requisição e a envia ao servidor.

3) O processo servidor contém um despachante junto a um procedimento stub de servidor e um procedimento de serviço para cada procedimento na interface de serviço.

4) o despachante seleciona um dos procedimentos stub de servidor de acordo com o identificador de procedimento presente na mensagem de requisição e chama os procedimentos de serviço correspondentes.

5) O stub de servidor empacota o resultado do procedimento e repassa para o despachante e este o envia de volta ao stub do cliente.

6) O stub do cliente desempacota o resultado e devolve a aplicação do cliente.

10) RMI permite o uso de orientação a objetos, permite também o uso de objetos distribuídos e tem suporte a chamadas de função com parâmetros enviados por referência, coisas que o RPC não suporta.

11) O repositório de implementações encapsula o código da maioria dos componentes distribuídos do sistema, dessa forma, a escalabilidade do sistema é muito simples, basta que os novos programas/aplicações/dispositivos acessem esse repositório para que tenham acesso a esses componentes. Na questão de tolerância a falhas, por outro lado, cria-se um único ponto do sistema que está sujeito a falhas, pois trata-se de um repositório centralizado. O que pode ser feito é a replicação desse repositório em diversas máquinas, evitando assim a presença de um único ponto de falha e possibilitando a transparência em casos extremos.

12) Fornece uma interface de serviço que permite a clientes a interação com servidores, trabalha com protocolos RR formatados em XML e transmitidos por HTTP. Suporta atividade conjunta e integração business-to-business, permite criação de software novo em cima de serviços já existentes, fornece um middleware para computação em grid e em nuvem. Se baseia na computação orientada a serviços, ou seja, um serviço é a unidade básica de sua operação e isso propicia um desenvolvimento rápido. Trata-se de um padrão adotado pela W3C capaz de atravessar firewalls, roteadores e servidores de proxy com uma abordagem simples e de fácil distribuição e interoperabilidade, além de se utilizar de padrões abertos e ser independente de uma plataforma.

13) A principal diferença é que o RMI trata-se de uma implementação muito específica para formas de comunicação ntre processos, enquanto CORBA é um padrão muito mais genérico e aplicável a diversas formas de comunicação e diversos tipos de sistemas, com uma grande diversidade, ainda, de linguagens de programação.

14) Sincronização dos relógios ajudam a resolver o problema da consistência dos dados em um sistema distribuído, por exemplo, em um sistema bancário, se há um depósito em uma conta e em seguida um saque, caso essas duas operações sejam feitas em dispositivos diferentes, é necessário que estes estejam sincronizados.

15) O NTP é uma rede hierárquica de servidores dentro da internet. Estes servidores se atualizam de 3 modos: Modo Multicast: um ou mais servidores periodicamente faz um multicast de seu tempo pros outros servidores que se atualizam. Modo Procedure Call: Servidores de níveis mais longe da raiz enviam uma requisição para seus servidores-pai pedindo o horário atual e se atualizam. Modo simétrico: servidores de um mesmo nível se comunicam par a par e se atualizam conforme seus tempos. Ao final desse processo todo, um cliente se comunica com algum servidor-folha na hierarquia e solicita o horário correto, recebendo sua resposta.

16) O Algoritmo de Cristian utiliza um servidor passivo para sincronizar os relógios do sistema, este servidor fornece sua hora local quando solicitado e os clientes calculam seus respectivos horários com base nessa hora e no RTT da mensagem na rede. Já o Algoritmo de Berkeley utiliza um servidor ativo que pergunta a todos os clientes na rede suas respectivas horas locais e, por meio de uma média desses valores, devolve aos clientes um valor de atraso ou adiantamento dos relógios locais para que haja a sincronia dentre todos os dispositivos. E por sua vez o NTP é baseado em UDP, se organiza numa estrutura de árvore onde cada nível (chamado stratus) é responsável por se atualizar com os níveis acima e manter a hora disponível para que os níveis abaixo possam se atualizar.

17) O método de Lamport especifica que, se um evento A ocorreu antes de um evento B (A->B), então o tempo de A é menor que o tempo de B (L(A)C), então por meio da propriedade da transitividade, é possível afirmar que A ocorreu antes de C. Dessa

forma, o método de Lamport vai ordenando o acontecimento dos eventos dentro do sistema distribuído.

18) Estado global é o conjunto de eventos que ocorreram até um momento do sistema. Por conta da dificuldade de sincronizar completamente os relógios de um Sistema Distribuído, aplica-se o conceito de corte, esse corte é uma marca no tempo que não precisa necessariamente ser a mesma para todos os processos envolvidos.

19) a) p1p3(m1); p1p2a(m2); p1p2b(m2); p2p1(m3); p3(m4); p3p1(m5);

b) p1p3a(1,0,0); p1p3b(1,0,1);

p1p2c(2,0,0); p1p2d(2,1,0);

p1p2e(3,0,0); p1p2f(3,3,0);

p2p1g(2,2,0); p2p1h(4,2,0);

p3i(0,0,2);

p3p1j(0,0,3); p3p1k(5,0,3);