Universidade Federal de Goiás – UFG
Instituto de Informática – INF
Sistemas de Informação
Sistemas Distribuídos

Aluno: Erik Pereira das Neves
Matricula: 202304398

Dê três exemplos específicos e contrastantes dos níveis de heterogeneidade cada vez maiores experimentados nos sistemas distribuídos atuais, conforme definido na Seção 2.2. página 39

Computação Móvel: Diferentemente das gerações anteriores de sistemas distribuídos, que eram compostos por computadores de mesa relativamente estáticos , os sistemas contemporâneos incorporam nós móveis, como notebooks e smartphones. Isso introduz heterogeneidade em termos de:

- **Localização e Conectividade**: Os nós podem mudar de local físico, o que exige suporte para descoberta de serviços e operação conjunta espontânea.
- **Capacidade do Dispositivo**: A capacidade de processamento, armazenamento e exibição varia drasticamente entre um smartphone e um servidor tradicional.

Computação Ubíqua: Esta tendência representa uma mudança de nós de computador distintos para arquiteturas onde a computação é incorporada em objetos do cotidiano e no ambiente. Um exemplo é uma "casa inteligente". A heterogeneidade aqui é vista na:

- **Forma e Função**: Os nós computacionais não são mais apenas computadores tradicionais, mas também eletrodomésticos, sensores e outros equipamentos incorporados.
- **Recursos de Hardware**: O sistema distribuído passa a abranger desde os menores dispositivos embarcados até elementos computacionais complexos.

Sistemas de Sistemas: Os sistemas distribuídos modernos podem ser tão complexos que são descritos como "sistemas de sistemas". Um exemplo contrastante é um sistema de gerenciamento ambiental para previsão de enchentes. Este único sistema agrega múltiplos subsistemas radicalmente diferentes, demonstrando um nível extremo de heterogeneidade:

- **Redes de sensores**: Implantadas para monitorar parâmetros ambientais.
- **Clusters computacionais**: Utilizados para executar simulações complexas de previsão.
- **Sistemas de alerta**: Que interagem com as partes interessadas por meio de telefones celulares.

Quais problemas você antevê no acoplamento direto entre entidades que se comunicam, que está implícito nas estratégias de invocação remota? Consequentemente, quais vantagens você prevê a partir de um nível de desacoplamento, conforme o oferecido pelo não acoplamento espacial e temporal?

Problemas do Acoplamento Direto em Estratégias de Invocação Remota

O acoplamento direto, implícito em paradigmas como a Invocação de Método Remoto (RMI) e a Chamada de Procedimento Remoto (RPC), cria uma dependência rígida entre as entidades que se comunicam. Os principais problemas que surgem dessa abordagem são:

Dependência de Existência Simultânea: No modelo de invocação remota, tanto o remetente quanto o destinatário devem existir e estar em execução ao mesmo tempo para que a comunicação ocorra. Se o processo servidor estiver temporariamente indisponível (por falha, reinicialização ou manutenção), qualquer invocação direta do cliente falhará, tornando o sistema menos resiliente.

Conhecimento Explícito do Destinatário: O remetente precisa direcionar explicitamente a invocação para um destinatário específico, conhecendo sua identidade e, muitas vezes, sua localização na rede. Isso torna o sistema inflexível. Se a localização do servidor mudar ou se um novo servidor for instanciado para balanceamento de carga, os clientes precisam ser atualizados, o que dificulta a manutenção e a evolução do sistema.

Fluxo de Comunicação Rígido: A comunicação é estritamente bilateral, entre um remetente e um destinatário. Isso torna complexo e ineficiente implementar padrões de comunicação mais elaborados, como a disseminação de informações para múltiplos consumidores (um-para-muitos), pois exigiria que o cliente gerenciasse múltiplas invocações diretas.

Vantagens do Desacoplamento (Espacial e Temporal)

As estratégias de comunicação indireta, como sistemas publicar-assinar ou filas de mensagens, introduzem um intermediário que possibilita um alto grau de desacoplamento, oferecendo vantagens significativas.

Vantagens do Desacoplamento Espacial:

Definição: Os remetentes não precisam saber para quem estão enviando as mensagens.

Vantagens:

- **Flexibilidade e Escalabilidade**: Produtores de informação podem simplesmente enviar eventos ou mensagens para um tópico ou fila, sem se preocupar com a quantidade ou a localização dos consumidores. Novos consumidores podem ser adicionados ou removidos dinamicamente sem qualquer alteração nos produtores, o que facilita a escalabilidade e a evolução do sistema.

- **Anonimato**: Os componentes podem interagir sem conhecerem a identidade uns dos outros, o que simplifica o desenvolvimento e permite a substituição de componentes com maior facilidade.

Vantagens do Desacoplamento Temporal:

Definição: Remetentes e destinatários não precisam existir ou estar ativos ao mesmo tempo.

Vantagens:

- **Resiliência e Tolerância a Falhas**: Um produtor pode enviar uma mensagem mesmo que o consumidor esteja offline ou indisponível. O intermediário armazena a mensagem até que o consumidor esteja pronto para recebê-la. Isso é crucial para sistemas distribuídos com componentes que podem ter conectividade intermitente (como dispositivos móveis) ou que operam em fusos horários diferentes.
- **Operação Assíncrona**: Permite que os produtores de mensagens não fiquem bloqueados esperando por uma resposta, podendo continuar seu processamento imediatamente após o envio. Isso melhora a eficiência e a capacidade de resposta do sistema.

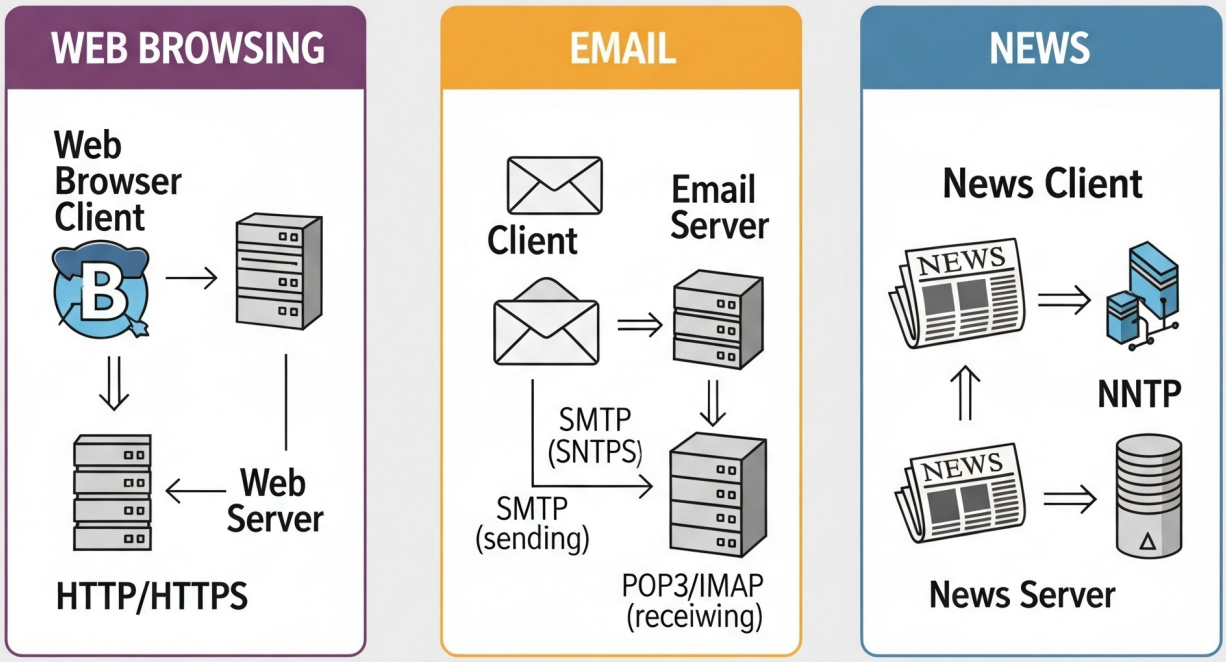
Descreva e ilustre a arquitetura cliente-servidor de um ou mais aplicativos de Internet importantes (por exemplo, Web, correio eletrônico ou news).

Arquitetura Cliente-Servidor

Nesta arquitetura, os processos assumem um de dois papéis: cliente ou servidor.

- **Processo Cliente**: É responsável por interagir com o usuário e iniciar a comunicação. Ele envia uma requisição (chamada de "invocação") para um processo servidor a fim de acessar um recurso compartilhado que o servidor gerencia. Um navegador Web é um exemplo clássico de um processo cliente.
- **Processo Servidor**: Este processo, localizado em um computador hospedeiro (possivelmente distinto do cliente), gerencia recursos compartilhados, como arquivos, páginas Web ou bancos de dados. Ele aguarda passivamente por requisições dos clientes, executa a operação solicitada e envia o resultado de volta. Um servidor Web que hospeda um site é um exemplo de processo servidor.

Uma característica importante é que um servidor pode, por sua vez, atuar como cliente de outros servidores. Por exemplo, um servidor Web é frequentemente um cliente de um servidor de arquivos local (para obter as páginas Web) e também um cliente do serviço DNS (para mapear nomes de domínio a endereços de rede).



Um mecanismo de busca é um servidor Web que responde aos pedidos do cliente para pesquisar em seus índices armazenados e (concomitantemente) executa várias tarefas de Web crawling para construir e atualizar esses índices. Quais são os requisitos de sincronização entre essas atividades concomitantes?

De acordo com o documento fornecido, um mecanismo de busca opera com duas tarefas principais que ocorrem concomitantemente: responder às consultas dos usuários e executar

web crawlers para atualizar seus índices.

Os requisitos de sincronização entre essas atividades são mínimos, pois as tarefas são consideradas "totalmente independentes". Especificamente:

- **Independência das Tarefas**: A tarefa do servidor de responder às consultas dos clientes e a tarefa do *web crawler* de fazer pedidos a outros servidores são independentes.
- **Baixa Necessidade de Sincronização**: Devido a essa independência, há pouca necessidade de sincronizá-las, o que permite que sejam executadas concomitantemente.
- **Execução Concorrente**: Um mecanismo de busca típico utiliza várias *threads* concorrentes, onde algumas atendem aos clientes enquanto outras executam os *web crawlers*.

Frequentemente, os computadores usados nos sistemas peer-to-peer são computadores desktop dos escritórios ou das casas dos usuários. Quais são as implicações disso na disponibilidade e na segurança dos objetos de dados compartilhados que eles contêm e até que ponto qualquer vulnerabilidade pode ser superada por meio da replicação?

Com base no documento fornecido, o uso de computadores desktop de escritórios e residências em sistemas peer-to-peer (P2P) tem as seguintes implicações:

Disponibilidade:

- A principal implicação para a disponibilidade é a desconexão de computadores individuais, o que é descrito como algo que "inevitavelmente, acontece" nessas redes. Como esses computadores não são servidores dedicados, eles podem ser desligados ou perder a conexão com a rede a qualquer momento, tornando os objetos de dados que armazenam indisponíveis.

Como a Replicação Supera a Vulnerabilidade de Disponibilidade:

- Para superar a baixa disponibilidade, a arquitetura P2P replica os objetos de dados em vários computadores.
- Essa estratégia fornece "poder de recuperação", garantindo que, se um computador for desconectado, os dados que ele continha ainda possam ser acessados a partir de outras cópias (réplicas) em outras máquinas na rede.
- Além de melhorar a tolerância a falhas, a replicação também ajuda a distribuir a carga de armazenamento, processamento e comunicação por muitos computadores e conexões de rede.

Considere um servidor simples que executa pedidos do cliente sem acessar outros servidores. Explique por que geralmente não é possível estabelecer um limite para o tempo gasto por tal servidor para responder ao pedido de um cliente. O que precisaria ser feito para tornar o servidor capaz de executar pedidos dentro de um tempo limitado? Essa é uma opção prática?

Com base no documento fornecido, a análise para um servidor simples que não acessa outros servidores é a seguinte:

Por que não é possível estabelecer um limite de tempo?

Geralmente não é possível estabelecer um limite para o tempo de resposta de um servidor porque os sistemas distribuídos de propósito geral são, na prática, **assíncronos**. Isso se deve a vários fatores que tornam a velocidade de execução imprevisível:

- **Velocidade de Execução do Processo**: Em um sistema assíncrono, não há garantias sobre as velocidades de execução dos processos; uma etapa pode levar um tempo arbitrariamente longo.
- **Carga do Servidor**: Não há um limite intrínseco para a carga do servidor, o que afeta diretamente o tempo de processamento de um pedido.
- **Recursos Compartilhados**: Os processos em sistemas reais precisam compartilhar recursos como tempo de processamento, canais de comunicação e acesso à rede. Se vários processos de características desconhecidas compartilham um processador, o desempenho de qualquer um deles não pode ser garantido.
- **Atrasos no Sistema Operacional**: O tempo de processamento gasto pelos serviços do sistema operacional nos processos de envio e recepção varia de acordo com a carga momentânea do computador.

O que seria necessário para executar pedidos em tempo limitado?

Para que o servidor fosse capaz de executar pedidos dentro de um tempo limitado, ele precisaria operar como um **sistema distribuído síncrono**. Isso exigiria:

- **Limites de Tempo Definidos**: Seria necessário que o tempo para executar cada etapa do processo do servidor tivesse limites inferior e superior conhecidos.
- **Alocação de Recursos**: Para garantir que as restrições temporais sejam respeitadas, seria preciso alocar os recursos necessários, como tempo de processamento e capacidade de rede, de forma dedicada.

- **Sistemas Especializados**: A implementação poderia exigir sistemas operacionais de tempo real, que são projetados especificamente para garantir o cumprimento de prazos.

Essa é uma opção prática?

Não, para a maioria dos sistemas de propósito geral, essa não é uma opção prática. O documento aponta as seguintes dificuldades:

- É muito difícil chegar a valores realistas para os limites de tempo e dar garantias de que serão cumpridos.
- A maioria dos sistemas operacionais de propósito geral, como o UNIX, não foi projetada para satisfazer restrições de tempo real.
- Os sistemas operacionais de tempo real, que poderiam oferecer tais garantias, são mais complexos em seu projeto e podem exigir hardware redundante, o que aumenta o custo.
- A necessidade de compartilhamento de recursos nos sistemas atuais torna a abordagem síncrona impraticável para aplicações comuns.

Considere dois processos, X e Y, que utilizam o serviço de comunicação B do Exercício 2.14 para se comunicar entre si. Suponha que X seja um cliente e que Y seja um servidor e que uma invocação consiste em uma mensagem de requisição de X para Y, seguida de Y executando a requisição, seguida de uma mensagem de resposta de Y para X. Descreva as classes de falha que podem ser exibidas por uma invocação

Com base no serviço de comunicação B e nos modelos de falha descritos no documento, uma invocação do cliente X para o servidor Y pode exibir as seguintes classes de falha:

O serviço de comunicação B é caracterizado por mensagens que "podem ser perdidas, retardadas... mas sempre chegam com o conteúdo correto". Isso significa que o canal de comunicação em si exibe falhas por omissão e de desempenho, mas não falhas arbitrárias (de conteúdo).

As falhas na invocação podem ocorrer em três pontos: na transmissão da requisição, na execução pelo servidor e na transmissão da resposta. As classes de falha são:

1. **Falhas por Omissão**
    - **Perda da mensagem de requisição**: A mensagem de X para Y pode ser perdida pelo canal de comunicação. Nesse caso, o servidor Y nunca recebe o pedido.
    - **Falha do servidor (colapso)**: O servidor Y pode entrar em colapso (parar de funcionar) após receber a requisição, mas antes de enviar a resposta.

- **Perda da mensagem de resposta**: Após Y executar a requisição, a mensagem de resposta enviada para X pode ser perdida pelo canal de comunicação.

2. **Falhas Arbitrárias (Bizantinas)**
    - **Falha do processo servidor**: Embora o canal não corrompa mensagens, o processo do servidor Y pode apresentar uma falha arbitrária. Por exemplo, ele pode executar a operação incorreta, retornar um valor errado em resposta a uma invocação, ou omitir passos do processamento. O cliente X receberia uma resposta válida em sua forma, mas incorreta em seu conteúdo.
3. **Falhas de Desempenho (ou de Temporização em um contexto assíncrono)**
    - **Atraso na comunicação**: A mensagem de requisição ou a mensagem de resposta podem ser significativamente retardadas pelo canal de comunicação.
    - **Servidor lento**: O processo do servidor Y pode demorar muito para processar o pedido. Como o sistema é assíncrono, uma resposta lenta não é uma falha de temporização (pois não há limites de tempo definidos), mas pode ser considerada uma falha de desempenho.

Descreva as possíveis ocorrências de cada um dos principais tipos de ameaça à segurança (ameaças aos processos, ameaças aos canais de comunicação, negação de serviço) que poderiam ocorrer na Internet.

1. Ameaças aos Processos

Essas ameaças ocorrem porque um processo que recebe uma mensagem não consegue determinar com certeza a identidade do remetente.

- **Ameaça a Servidores**: Um servidor na Internet, como um servidor de e-mail, pode receber uma requisição de um atacante que falsificou o endereço de origem. Por exemplo, um invasor poderia enviar uma solicitação para ler os e-mails de uma caixa de correio específica, e o servidor não teria como saber se o remetente é o usuário autorizado ou um impostor. Sem um reconhecimento garantido da identidade, o servidor não sabe se deve executar a operação ou rejeitá-la.
- **Ameaça a Clientes**: Um processo cliente, como um navegador web, pode ser vítima de *spoofing*, onde um invasor se passa por um servidor legítimo. Por exemplo, ao solicitar o conteúdo de sua caixa de e-mail, um cliente poderia receber uma mensagem falsa, enviada por um atacante, que não estava originalmente na sua caixa de correio. O cliente não teria como identificar se a resposta veio do servidor desejado ou de um invasor.

2. Ameaças aos Canais de Comunicação

Um invasor pode copiar, alterar ou injetar mensagens enquanto elas trafegam pela rede da Internet. Isso representa uma ameaça à privacidade e à integridade das informações.

- **Violação de Privacidade e Integridade**: Um atacante pode interceptar comunicações na rede. Por exemplo, uma mensagem de e-mail contendo informações confidenciais poderia ser copiada e lida por outra pessoa (ameaça à privacidade). Além disso, o conteúdo da mensagem poderia ser alterado para dizer algo completamente diferente (ameaça à integridade).
- **Ataque de Reprodução (*Replay Attack*)**: Um invasor pode salvar cópias de mensagens transmitidas e reenviá-las posteriormente. Um exemplo prático na Internet seria um atacante que captura uma mensagem de invocação solicitando uma transferência de dinheiro de uma conta bancária e a reenvia várias vezes para repetir a transação indevidamente.

3. Negação de Serviço (Denial of Service - DoS)

Esta é uma forma de ataque onde o objetivo é interferir nas atividades dos usuários autorizados, tornando um serviço indisponível.

- **Sobrecarga de Recursos**: O atacante pode sobrecarregar os recursos de um servidor ou da rede. Isso pode ser feito enviando um número massivo de invocações sem sentido a um serviço online, como um site de e-commerce, ou inundando a rede com um fluxo incessante de mensagens.
- **Impedimento de Acesso Legítimo**: O resultado dessa sobrecarga é que o servidor fica lento ou para de responder completamente, impedindo que usuários válidos consigam acessar o serviço. Por exemplo, um ataque de DoS poderia saturar o computador que controla um serviço de home banking com tantos pedidos inválidos que os clientes legítimos seriam impedidos de realizar suas transações.