UNIVERSIDADE CATÓLICA DE PELOTAS
ENGENHARIA DE COMPUTAÇÃO
SISTEMAS DISTRIBUIDOS

Gabriel Harter Zoppo

Protocolos de requisição-resposta

Pelotas, 24 de junho de 2022

Protocolos de requisição-resposta:

No caso normal, a comunicação por requisição- -resposta é síncrona, pois o processo cliente é bloqueado até que a resposta do servidor chegue. Ela também pode ser confiável, pois a resposta do servidor é efetivamente um confirmação para o cliente. A comunicação por requisição-resposta assíncrona é uma alternativa útil em situações em que os clientes podem recuperar as respostas posteriormente.

O método doOperation é usado pelos clientes para invocar operações remotas. Seus argumentos especificam o servidor remoto e a operação a ser invocada, junto às informações adicionais (argumentos) exigidas pela operação. Seu resultado é um vetor de bytes contendo a resposta. Presume-se que o cliente que chama doOperation empacota os argumentos em um vetor de bytes e desempacota os resultados do vetor de bytes retornado.

O getRequest é usado por um processo servidor para obter requisições de serviço. Quando o servidor tiver invocado a operação especificada, ele usa sendReply para enviar a mensagem de resposta ao cliente. Quando a mensagem de resposta é recebida pelo cliente, a operação doOperation original é desbloqueada, e a execução do programa cliente continua

Pergunta 1:

Que tipo de falhas ocorrem em protocolos de requisição resposta?

- Falhas por omissão.
- Falta de garantias de entrega das mensagens na ordem do envio
- Falha de processos

Tempos limite (timeouts):

Existem várias opções para o que doOperation deva fazer após esgotar um tempo limite (timeout). A opção mais simples é retornar imediatamente de doOperation, com uma indicação para o cliente de que doOperation falhou. Essa não é a estratégia mais comum – o tempo limite pode ter esgotado devido à perda da mensagem de requisição ou de resposta e, neste último caso, a operação foi executada. Para levar em conta a possibilidade de mensagens perdidas, doOperation envia a mensagem de requisição repetidamente, até receber uma resposta ou estar razoavelmente seguro de que o atraso se deve à falta de resposta do servidor e não à perda de mensagens. Finalmente, quando doOperation retornar, indicará isso para o cliente por meio de uma exceção, dizendo que nenhum resultado foi recebido.

Descarte de mensagens de requisição duplicadas:

Para evitar mensagens de requisição duplicadas, o protocolo é projetado de forma a reconhecer mensagens sucessivas (do mesmo cliente) com o mesmo identificador de requisição e eliminar as duplicatas. Se o servidor ainda não enviou a resposta, não precisa executar nenhuma ação especial – ele transmitirá a resposta quando tiver terminado de executar a operação.

Mensagens de resposta perdidas:

Uma operação idempotente é aquela que pode ser efetuada repetidamente com o mesmo efeito, como se tivesse sido executada exatamente uma vez. Um servidor cujas operações são todas idempotentes não precisa adotar medidas especiais para evitar suas execuções mais de uma vez

Histórico:

Para os servidores que exigem retransmissão das respostas sem executar novamente as operações, pode-se usar um histórico. Seu objetivo é permitir que o servidor retransmita as mensagens de resposta quando os processos clientes as solicitarem. Um problema associado ao uso de um histórico é seu consumo de memória. Um histórico pode se tornar muito grande, a menos que o servidor possa identificar quando não há mais necessidade de retransmissão das mensagens.

O histórico precisa conter apenas a última mensagem de resposta enviada a cada cliente. Entretanto, o volume de mensagens de resposta no histórico de um servidor pode ser um problema quando ele tiver um grande número de clientes. Isso é combinado com o fato de que, quando um processo cliente termina, ele não confirma a última resposta recebida – portanto, as mensagens no histórico normalmente são descartadas após determinado período de tempo.

Estilos de protocolos de troca:

Pergunta 2:

Quais são os três protocolos, que produzem diferentes comportamentos na presença de falhas de comunicação e são usados para implementar vários tipos de comportamento de requisição?

- o protocolo request (R);
- o protocolo request-reply (RR);
- o protocolo request-reply-acknowledge reply (RRA).

O protocolo R pode ser usado quando não existe nenhum valor a ser retornado do método remoto e o cliente não exige confirmação de que a operação foi executada. O cliente pode prosseguir imediatamente após a mensagem de requisição ser enviada, pois não há necessidade de esperar por uma mensagem de resposta. Esse protocolo é implementado sobre datagramas UDP e, portanto, sofre das mesmas falhas de comunicação.

O protocolo RR é útil para a maioria das trocas cliente-servidor, pois é baseado no protocolo de requisição-resposta. Não são exigidas mensagens de confirmação especiais, pois uma mensagem de resposta (reply) do servidor é considerada como confirmação do recebimento da mensagem de requisição (request) do cliente.

O protocolo RRA é baseado na troca de três mensagens: requisição, resposta e confirmação. A chegada de um requestId em uma mensagem de confirmação será interpretada como a acusação do recebimento de todas as mensagens de resposta com valores de requestId menores; portanto, a perda de uma mensagem de confirmação não é muito prejudicial ao sistema.

Uso de TCP para implementar o protocolo de requisição-resposta:

A implementação de protocolos de requisição-resposta com TCP, permitindo a transmissão de argumentos e resultados de qualquer tamanho. Se o protocolo TCP for usado, isso garantirá que as mensagens de requisição e de resposta sejam entregues de modo confiável. O protocolo TCP simplifica a implementação de protocolos de requisição-resposta. A sobrecarga em razão das mensagens de confirmação TCP é reduzida quando uma mensagem de resposta é gerada logo após a mensagem de requisição. Suas operações são projetadas para serem idempotentes, tornando desnecessário manter um histórico.

HTTP: um exemplo de protocolo de requisição-resposta:

Terceira pergunta: Como funcionam a negociação de conteúdo e a Autenticação?

Negociação de conteúdo: As requisições dos clientes podem incluir informações sobre qual representação de dados elas podem aceitar (por exemplo, linguagem ou tipo de mídia), permitindo que o servidor escolha a representação mais apropriada para o usuário.

Autenticação: credenciais e desafios (challenges) são usados para suportar autenticação com senha. Na primeira tentativa de acessar uma área protegida com senha, a resposta do servidor contém um desafio aplicável ao recurso.

Uma conexão persistente pode ser encerrada a qualquer momento, tanto pelo cliente como pelo servidor, pelo envio de uma indicação para o outro participante. Os servidores encerrarão uma conexão persistente quando ela estiver ociosa por determinado período de tempo. As requisições e respostas são empacotadas nas mensagens como strings de texto ASCII, mas os recursos podem ser representados como sequências de bytes e podem ser compactados. A utilização de texto na representação externa de dados simplificou o uso de HTTP pelos programadores de aplicativos que trabalham diretamente com o protocolo. Neste contexto, uma representação textual não aumenta muito o comprimento das mensagens.

Métodos HTTP:

Quarta pergunta:
Quais são os métodos HTTP?

GET: Solicita o recurso cujo URL é dado como argumento. Se o URL se referir a dados, o servidor Web responderá retornando os dados identificados por esse URL. Se o URL se referir a um programa, então o servidor Web executará o programa e retornará sua saída para o cliente.

HEAD: esta requisição é idêntica a GET, mas não retorna nenhum dado. Entretanto, retorna todas as informações sobre os dados, como a hora da última modificação, seu tipo ou seu tamanho.

POST: especifica o URL de um recurso (por exemplo, um programa) que pode tratar dos dados fornecidos no corpo do pedido. O processamento executado nos dados depende da função do programa especificado no URL. Esse método é feito para lidar com:

- O fornecimento de um bloco de dados para um processo de manipulação de dados, como um servlet – por exemplo, enviando um formulário Web para comprar algo em um site;
- O envio de uma mensagem para uma lista de distribuição ou da atualização de detalhes de membros da lista;
- A ampliação de um banco de dados com uma operação append.

PUT: solicita que os dados fornecidos na requisição sejam armazenados no URL informado, como uma modificação de um recurso já existente ou como um novo recurso.

DELETE: o servidor exclui o recurso identificado pelo URL fornecido. Nem sempre os servidores permitem essa operação; nesse caso, a resposta indicará a falha.

OPTIONS: o servidor fornece ao cliente uma lista de métodos que podem ser aplicados no URL dado (por exemplo, GET, HEAD, PUT) e seus requisitos especiais.

TRACE: o servidor envia de volta a mensagem de requisição. Usado para propósitos de diagnóstico.