MAC0352 - Redes de Computadores e Sistemas Distribuídos - 2s2023
EP1
Entrega até 8:00 de 25/9/2023
(INDIVIDUAL)

Prof. Daniel Macêdo Batista

1 Problema

Neste EP você deverá implementar a interpretação e o processamento de algumas mensagens da camada de aplicação de um servidor AMQP na versão 0.9.1 do protocolo. O código referente às camadas inferiores não deve ser escrito, pois o código de um servidor de eco está disponibilizado no e-Disciplinas e deve ser usado como base (Apenas os trechos referentes à camada de aplicação nesse código devem ser modificados para transformá-lo em um servidor AMQP. Claro que novas variáveis, funções ou *headers* podem ser incluídos livremente caso necessário).

Seu servidor não precisa ser um `RabbitMQ` [1]! **Ele só precisa aceitar conexões e desconexões de clientes, receber e enviar mensagens em uma fila específica sem se preocupar com falhas e sem se preocupar com autenticação ou criptografia**.

Para entender mais sobre isso, consulte a especificação do protocolo no site oficial em `https://www.amqp.org/specification/0-9-1/amqp-org-download` e a explicação do modelo do protocolo na página do RabbitMQ em `https://www.rabbitmq.com/tutorials/amqp-concepts.html`. A correção do servidor será feita utilizando os programas `amqp-publish`, `amqp-consume` e `amqp-declare-queue`, três utilitários do protocolo AMQP disponibilizados no pacote debian `amqp-tools`[2]. É altamente recomendável que você entenda como a comunicação entre cliente e servidor AMQP funciona instalando o `RabbitMQ` na sua máquina e realizando comunicações com o `amqp-publish` e o `amqp-consume` enquanto o `Wireshark` é executado[3]. Observar os pacotes no `Wireshark` vai ajudar muito a entender o protocolo. Talvez mais do que ler a especificação (Em certos casos, ver a troca de mensagens no `Wireshark` dispensa a leitura da especificação de um protocolo ao implementar algum programa que interprete as mensagens desse protocolo).

Você verá que, por padrão, quando mais de um assinante está conectado no servidor em uma mesma fila, as mensagens enviadas para aquela fila são alternadas entre os assinantes em um esquema Round Robin. Esse comportamento deve ser mantido na sua implementação.

[1] `https://www.rabbitmq.com/`

[2] `https://packages.debian.org/bookworm/amqp-tools`

[3] Dica 1: antes é necessário criar uma fila com o `amqp-declare-queue`

2 Requisitos

2.1 Comportamento do servidor AMQP

O servidor, ou *broker*, como é mais conhecido um servidor de um sistema *publish-subscribe*, deve se comportar de forma similar a como o servidor `RabbitMQ` se comporta em quatro situações:

- Declaração da fila;
- Conexão de vários clientes simultaneamente (cada cliente simultâneo pode publicar ou requisitar mensagens da mesma fila ou de filas distintas);
- Inscrição de cliente em uma fila e consequente envio das mensagens desta fila para o cliente respeitando o esquema de Round Robin caso mais de um cliente esteja conectado na mesma fila;
- Publicação de mensagem em uma fila;
- Desconexão de cliente.

As mensagens e as filas devem respeitar os limites de tamanho do protocolo e devem possuir apenas caracteres ASCII[4].

Para saber como o `RabbitMQ` se comporta em todos os casos, instale o servidor em alguma máquina e faça os testes rodando o `Wireshark` em paralelo.

Caso você não tenha GNU/Linux na sua máquina ou se você não quer instalar o servidor `RabbitMQ` com medo de esquecer de desinstalá-lo e ele ficar aceitando conexões na sua máquina, crie uma máquina virtual na sua própria máquina física, instale alguma distribuição de GNU/Linux, instale o `RabbitMQ` e capture os pacotes com o `Wireshark`. Informações sobre a utilização de máquinas virtuais podem ser encontradas em `https://www.virtualbox.org/`, `https://xenproject.org/` ou `https://www.vmware.com/`.

Seu EP **não** precisa se preocupar com falhas. Por exemplo, se um cliente que estiver assinando uma fila for interrompido por qualquer motivo e nesse meio tempo alguma mensagem for enviada para aquela fila, quando ele reconectar ele não precisa receber as mensagens que foram enviadas enquanto ele esteve desconectado.

2.2 Linguagem

O servidor deve ser escrito em C. Certifique-se de que ele funciona no GNU/Linux pois ele será compilado e avaliado apenas neste sistema operacional.

O código `redes-servidor-exemplo-ep1.c` disponível no e-Disciplinas deve ser usado como base. Ele é um servidor de eco. Leia os comentários no início do código para entender como fazer para executá-lo. Toda a parte de gerência da conexão no código pode ser ignorada. Basta focar no trecho onde devem ser feitas as mudanças para o EP, que está identificado no código.

[4] `man ascii` em uma máquina rodando GNU/Linux vai mostrar quais são esses 128 caracteres

2.3 Slides

Você deverá entregar, além dos códigos, um .pdf com slides que você usaria caso você fosse apresentar o seu trabalho. Os slides deverão descrever, em alto nível, a sua implementação destacando os pontos principais de cada comando do AMQP que foi implementado. Decisões de projeto que você julgar que merecem ser apresentadas também podem ser incluídas. Além das informações sobre a implementação, os slides também devem apresentar gráficos de análise de desempenho comparando o desempenho do servidor, em termos de uso de rede e de CPU, em três cenários: (i) apenas com o servidor, sem nenhum cliente conectado; (ii) com o servidor e com dez clientes publicando e recebendo mensagens simultaneamente e (iii) com o servidor e cem clientes publicando e recebendo mensagens simultaneamente. Nos slides inclua informações sobre o ambiente computacional e de rede que você usou para realizar os experimentos e como você avaliou a carga na rede de modo a garantir que tudo que você mediu foi de fato decorrente da comunicação do seu código e não de outros programas que estavam usando a rede ao mesmo tempo. Note que você não precisa ter 100 computadores reais para ir testando o EP e para fazer os experimentos. Vários desses clientes podem estar rodando no mesmo computador, que pode ser na verdade uma máquina virtual rodando no `VirtualBox`, `Xen` ou `VMWare`.

Esses slides não serão apresentados mas considere que eles seriam apresentados em um tempo máximo de 10 minutos. Portanto, antes de enviar seu .pdf, faça um ensaio da apresentação para garantir que a quantidade de conteúdo cabe dentro de 10 minutos de apresentação.

Entregas sem o .pdf da apresentação não serão corrigidas e receberão nota ZERO.

3 Entrega

Você deverá entregar um arquivo .tar.gz contendo os seguintes itens:

- fonte;
- Makefile (ou similar);
- arquivo LEIAME;
- .pdf dos slides.

O desempacotamento do arquivo .tar.gz deve produzir um diretório contendo os itens. O nome do diretório deve ser ep1-seu_nome. Por exemplo: ep1-joao_dos_santos.

A entrega do .tar.gz deve ser feita no e-Disciplinas.

O EP deve ser feito individualmente.

Obs.1: Serão descontados 2,0 pontos de EPs com arquivos que não estejam nomeados como solicitado ou que não criem o diretório com o nome correto após serem descompactados. Confirme que o seu .tar.gz está correto, descompactando ele no shell (não confie em interfaces gráficas na hora de testar seu .tar.gz pois alguns gerenciadores de arquivos criam o diretório automaticamente mesmo quando esse diretório não existe. Se você nunca usou o comando tar, leia a manpage e "brinque" um pouco com ele para entender o funcionamento).

Obs.2: A depender da qualidade do conteúdo que for entregue, o EP pode ser considerado como não entregue, implicando em MF=0,0. Isso acontecerá por exemplo se for enviado um .tar.gz corrompido, ou códigos fonte vazios.

Obs.3: O prazo de entrega expira às 8:00:00 do dia 25/9/2023.

4 Avaliação

60% da nota será dada pela implementação, 10% pelo LEIAME e 30% pelos slides. Os critérios detalhados da correção serão disponibilizados apenas quando as notas forem liberadas.

Obs.: a correção não será feita com conexões no localhost (127.0.0.1) mas sim com os processo rodando em máquinas distintas em uma rede local. Certifique-se de que seu programa funciona corretamente mesmo em cenários onde as conexões não venha do localhost.