Trabalho 2

Autor: Renato Britto Araujo 180027239

Este projeto é uma demonstração de sistema distribuido usando sockets em C. A meta é gerar uma lista de valores pseudo-aleatórios e utilizar um servidor remoto para iterar sobre esta lista a extrair os máximos e mínimos entre os valores. Um cliente é o resposável por gerar a lista, e um ou mais servidores é resposável por calcular mínimos e máximos.

Foi utilizado a comunicação TCP porque existe um custo de manutenção associado com UDP no caso de perda de mensagens. Com TCP, é mais fácil ter a certeza de que a mensagem recebida é confiável e não está corrompida, fazendo o código menor e mais limpo. Não foram encontradas nenhuma justificativa interessante o suficiente para usar UDP levando em conta o custo associado dessa manutenção. Além disto, a resposta final no cliente apenas pode estar correta se e apenas se todos os números foram analisados por um servidor, ou seja, a perda de pacotes não é tolerável.

Não existe distinção entre os dois sistemas (multiplos servidores ou não), o mesmo cliente é capaz de lidar com os dois. O uso de múltiplos servidores funciona por um sistema de dividir e conquistar, ou seja, a lista de floats é segmentada em partes iguais para cada um dos servidores disponíveis (arbitrário) e o resultado é calculado localmente e então unificado no resultado final.

Como o único membro já tinha conhecimento tanto de threads quanto de sockets, nada de novo foi aprendido.

A nota auto-atribuída para este projeto é 9: todos os requisitos cumpridos com sucesso. Os únicos elementos restantes seriam adicionar tratamento de erros de comunicação, que não foi feito, e realizar a mesma implementação em UDP (o que não é um requisito, porém seria de valor para o conhecimento). Em questões positivas, é possível usar um número arbitrário de servidores sem que estes precisem ser constantemente criados e derrubados, sem interrupções na comunicação. O sistema é bem simples, está bem escrito, tem logs apropriados e está bem comentado.

A seguir, uma tabela que demonstra o número de servidores para o tempo de resposta do processo cliente. O tempo foi extraido usando o comando ‘time’ em um Ubuntu 20.04.

Numero de servidores	**Tempo de resposta (ms)**
1	11
2	7
4	5
6	4
8	4
10	4

Processo de comunicação

Cliente

1. Cria array
2. Segmenta array para o numero de servidores disponíveis (arbitrários)
3. Para cada segmento
    1. Enquanto houverem números a serem enviados
        1. Envia no pacote tamanho do buffer temporário
        2. Envia no mesmo pacote floats do buffer (4 bytes cada)
        3. Espera acknowledge
    2. Envia mensagem 'RES' como pedido de resposta
    3. Recebe resposta desta thread
4. Unifica respostas recebidas usando dividir e conquistar
7. Print do resultado

Servidor

1. Liga o servidor
2. Para cada uma das conexões
    2. Enquanto houverem números a serem enviados
        1. Tamanho do buffer temporário
        2. Floats do buffer (4 bytes)
        3. Envia acknowledge
    3. Ao receber RES, calcula resultado sobre buffer final
    4. Envia ao cliente a resposta (min/max, 9 bytes)
    5. Encerra conexão

Como usar

Compile

```
./cmp.sh
```

Levante um número arbitrário de servidores

```
./server 0.0.0.0 <porta>
```

Por questão de clareza, faça cada servidor em um terminal separado

Levante uma instância de cliente

```
./client 0.0.0.0 <porta servidor 1> <porta servidor 2> ...
```

Por questão de clareza, faça o cliente em um terminal separado
Note que é necessário que pelo menos um servidor seja utilizado