


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA

Relatório da Disciplina de Sistemas Distribuídos

Aluno: Phelipe Wener
Matrícula: 12/013289

Professor:
Fernando W. Cruz

Brasília, DF
27 de Março de 2017



1. Objetivo

O objetivo desse experimento é que o aluno compreenda as características inerentes à construção de aplicações distribuídas, incluindo passagem de parâmetros, envolvendo módulos cliente e servidor conectados por sockets UDP e TCP.

2. Ambiente e configuração

Todos os experimentos abaixo foram executados numa distribuição linux chamada kubuntu, que difere fundamentalmente em termos de interface com o ubuntu. Portanto, deve ser possível executar os mesmos passos em qualquer versão do Ubuntu 14.04, bem como em um Debian 7.

Já na compilação do código, foi utilizado gcc na versão 4.8.4, com a flag `-Wall` ativa, tal que seja possível corrigir todos warnings acusados pelo compilador.

3. Utilizando protocolo UDP

Os códigos `udpMathClient.c`(Cliente) e `UdpMathServer.c`(Servidor) se comunicam afim de retornar o resultado de uma operação matemática simples (soma, subtração, multiplicação e divisão). Os passos executados pelo código se resumem da seguinte forma:

- Servidor fica esperando uma requisição em uma porta x
- Cliente manda uma operação simples para o endereço do servidor com a porta x
- Servidor recebe a operação realizando o cálculo
- Servidor responde ao cliente o resultado
- Cliente imprime o resultado

No terminal, as seguintes telas foram obtidas:

- Servidor ligado

```
$ ./ums 192.168.0.23 9090
./ums: esperando por dados no IP: 192.168.0.23, porta UDP numero: 9090
```

- Cliente enviando operação e recebendo resultado

```
$ ./umc 192.168.0.23 9090 5 + 7
Enviando parametro 1: 5
Enviando parametro 2: +
Enviando parametro 3: 7
Result is 12.000000
```

3.1 Problemas encontrados

Inicialmente houve uma dificuldade em saber como se tratar os dados para calcular o resultado, entretanto foi uma discursão mais relacionada a forma como poderia ser feito. Ainda em sala, o problema foi resolvido mantendo os operandos e o operador em um array de strings(ponteiros de

char) com três posições.

3.2 Limitações de código

O código do cliente udp aceita cada argumento como entrada para a operação matemática, portanto é necessário colocar cada o cálculo que se quer no seguinte formato:

```
./umc <ip_do_servidor> <porta> operando operador operando
```

Sem os espaços o servidor vai entender tudo como um operando.

Além disso, uma vez executado de forma certa, o servidor realiza a operação, devolve o resultado e finaliza, o cliente recebe o resultado e também é finalizado, portanto a cada execução é possível calcular uma única operação.

3.3 Resultados

O servidor UDP respondeu a múltiplos clientes de forma integrada, tal que se cada cliente mandasse um operando ou operador, ao final ele mandava o resultado para o último cliente. O que significa que a cada nova conexão o servidor implementado consegue continuar o seu algoritmo de cálculo como se estivesse comunicando com mesmo cliente. Dessa forma o servidor não muda o comportamento de acordo com quem está mandando o pacote.

O protocolo UDP supriu as necessidades da aplicação proposta sem nenhuma deficiência.

3.4 Conclusões

Criar uma ligação cliente-servidor utilizando o protocolo UDP tem única e exclusivamente a dificuldade técnica em se entender cada função utilizada. Para tal, no sistema operacional utilizado, basta que se execute no terminal o comando `man nome_funcao` para saber o suficiente para criar uma aplicação. Dentre as pesquisas realizadas, estão `recvfrom`, `bind` e `send_to`, importantes funções para comunicação entre sockets.

O que torna uma conexão com protocolo UDP é basicamente o parâmetro `SOCK_DGRAM`, o que será mais explicado no experimento TCP, principalmente em conclusões.

4. Utilizando protocolo TCP

Os códigos `tcpMathClient.c`(Cliente) e `tcpMathServer.c`(Servidor) se comunicam afim de retornar o resultado de uma operação matemática simples (soma, subtração, multiplicação e divisão). Os passos executados são os mesmos explicados no UDP, no tópico 3

O experimento com TCP é muito similar ao UDP em termos de conexão de sockets. Porém o programa implementado para esse experimento tem algumas particularidades.

No terminal, as seguintes telas foram obtidas ao se executar os 2 programas:

- Servidor recebendo mensagens

```
$ ./tms 192.168.0.23 9090
Servidor ouvindo no IP 192.168.0.23, na porta 9090 ...

Client 192.168.0.23: 48842 conectado.
[192.168.0.23:48842] => 1 + 1

[192.168.0.23:48842] => 2 * 2

[192.168.0.23:48842] => 4 - 4

[192.168.0.23:48842] => 6 / 2
```

- Cliente enviando operação e recebendo resultado

```
$ ./tmc 192.168.0.23 9090
> 1 + 1
Result is 2.000000
> 2 * 2
Result is 4.000000
> 4 - 4
Result is 0.000000
> 6 / 2
Result is 3.000000
>
```

3.1 Problemas encontrados

Um dos maiores problemas ocorreu ao apagar a variável que recebia o retorno de `recv` no servidor, pois uma vez ligado, quando o cliente mandava uma mensagem, por algum motivo, o `while` rodava sucessivamente imprimindo o último `print` do loop com entrada vazia. Ao declarar uma variável para guardar esse retorno, o `recv` se comportava de maneira esperada, aguardando o envio de mensagem. Provavelmente isso era algum problema de buffer.

3.2 Limitações de código

Para entrar com uma operação basta estabelecer conexão com o servidor e digitar uma operação em qualquer formato parecido com:

```
operando operador operando
```

Dessa forma o servidor devolve o resultado ao cliente.

3.3 Resultados

Como foi utilizado `fork` para criação de processos filhos, o servidor permite multiplas conexões,

porém diferente do primeiro, o resultado não pode ser calculado de forma integrada. Ainda que seja possível fazer isso através de variáveis compartilhadas. Para a aplicação TCP que se apresentava, o protocolo TCP foi suficiente e não apresentou nenhuma falha.

3.4 Conclusões

Bem como o UDP, o TCP foi suficiente para a comunicação sugerida pelo roteiro. Nos dois protocolos foi possível receber uma mensagem de um cliente por meio de um servidor e responder com o mesmo para o cliente, ambos utilizando sockets. A grande diferença entre os dois nos experimentos é que o TCP conseguia estabelecer conexão, enquanto o UDP sempre sobrescrevia a conexão, isso pode ser atestado mudando as seguintes linhas de código do UDP:

```
sd = socket(AF_INET, SOCK_DGRAM, 0);  
sd = socket(AF_INET, SOCK_STREAM, 0);
```

Ao fazer isso o servidor UDP se comporta de forma a abrir e fechar múltiplas conexões, a arquitetura usada para a aplicação não permitia a ele estabelecer uma conexão única.