

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Redes de Computadores

TRABALHO PRÁTICO 3
Servidor para múltiplos clientes

Guilherme Saulo Alves

28 de Junho de 2015

Belo Horizonte – MG

1. Introdução

1.1. Objetivo

O desafio deste trabalho prático foi implementar um sistema de envio de mensagens utilizando o modelo cliente-servidor em que o servidor atenda a múltiplas requisições de clientes utilizando a biblioteca socket do Linux. Além disso, o desenvolvimento do trabalho propôs encontrar um contra-exemplo para a Conjectura de Beal por meio de uma troca de mensagens usando protocolo TCP.

1.2. Conjectura de Beal

- Se $A^x + B^y = C^z$, onde A, B, C, x, y e z são inteiros positivos e $x, y, z \geq 3$, então A, B e C têm um fator primo comum - o que significa que a, b e c são divisíveis por um mesmo número primo.

Uma outra forma equivalente de enunciar a Conjectura de Beal seria:

- A equação $A^x + B^y = C^z$ não tem solução para inteiros positivos com $x, y, z \geq 3$ e $\text{mdc}(A, B, C) = 1$.

1.2.1. Contra-Exemplo

Para encontrar um contra-exemplo para a conjectura é necessário encontrar uma solução em inteiros da equação $A^x + B^y = C^z$ com todos os expoentes maiores que dois, sem existir um fator primo comum à A, B e C , ou seja, $\text{mdc}(A, B, C) = 1$. Por exemplo: A equação $2^5 + 7^2 = 3^4$ não é um contra-exemplo, mesmo sendo $\text{mdc}(2, 7, 3) = 1$, haja vista que os expoentes de A, B e C não são todos maiores que 2.

1.3. Arquitetura Servidor

Um servidor concorrente lida com vários clientes ao mesmo tempo. A técnica usada para gerar um servidor de múltiplas requisições foi por meio da função *fork()*. A função cria um processo filho para cada cliente. Quando uma conexão é estabelecida, o servidor chama *fork()* e o processo filho atende o cliente. Enquanto isso, o processo principal espera por outra ligação acontecer. A interação entre cliente e servidor é apresentado na figura a seguir:

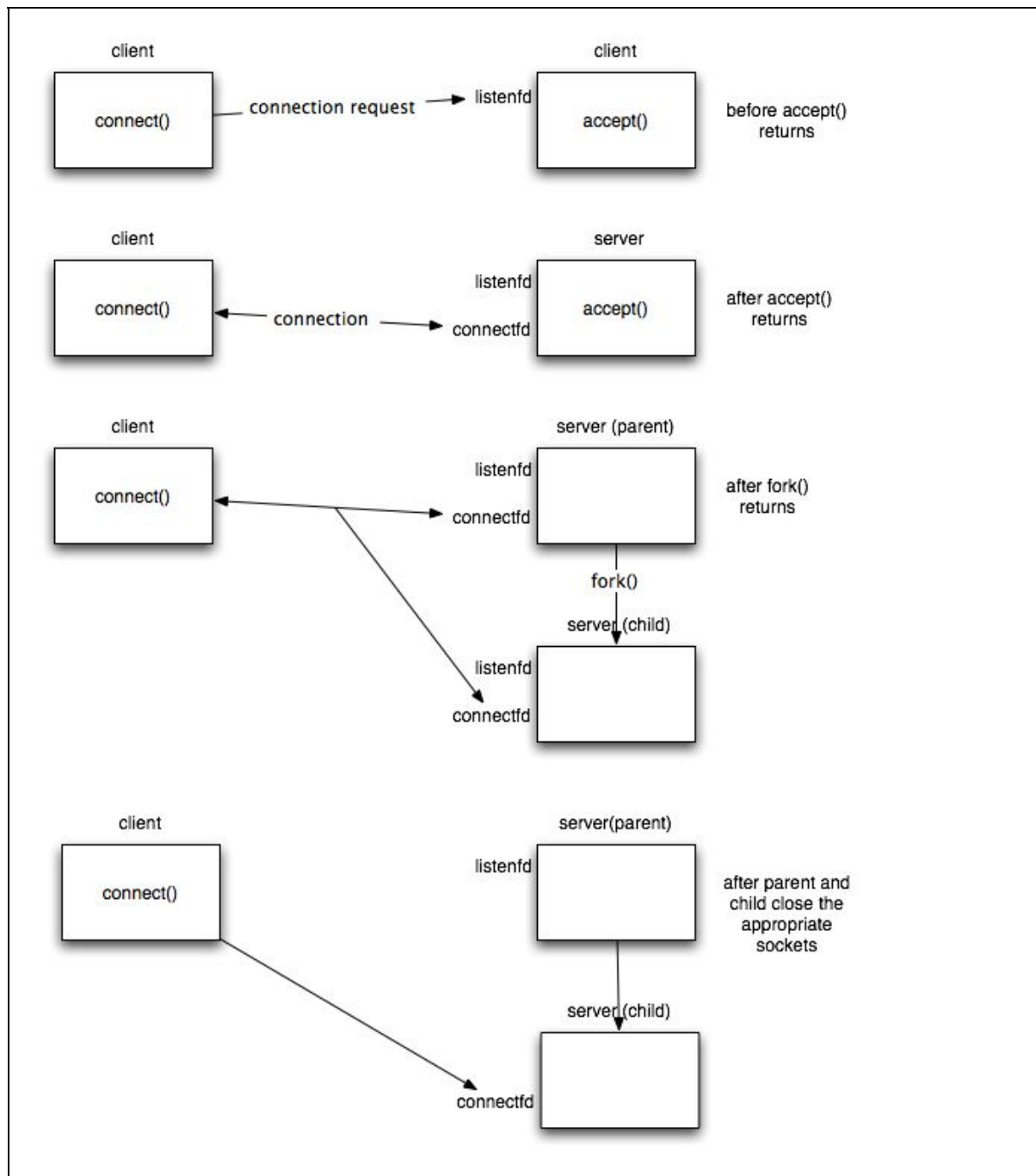


Figura 1 - Exemplo de interação entre um cliente e um servidor de múltiplas requisições.

2. Implementação

2.1. Funcionamento

O servidor é iniciado executando o comando `./servidor <porta_servidor>` no terminal. Em seguida é realizada o processamento dos argumento da linha de comando. Caso a porta seja 80, o servidor responde com uma página HTML

com o status atual do servidor. Caso contrário, ele obtém informações de endereço do servidor, cria um socket pro servidor e aguarda conexão do cliente.

Um cliente é iniciado executando o comando `./cliente <host_do_servidor> <porta_servidor>` no terminal. Em seguida é verificado os argumentos e criado um socket pro cliente. Logo em seguida o cliente estabelece a conexão com o servidor e envia uma mensagem para contendo um identificador do cliente. Para identificação do cliente, foi usado o numero do pid do processo do cliente na maquina.

O servidor aceita a conexão do cliente, cria um processo filho e confirma o recebimento do id do cliente respondendo com quatro valores inteiros *bmin*, *bmax*, *pmin*, *pmax* onde representam respectivamente os valores mínimos e máximos para a base A,B,C e expoentes x,y,z da equação da Conjetura de Beal.

O cliente irá responder o servidor confirmando os valores recebidos com uma mensagem *ACK*. Em seguida o cliente irá procurar um contra exemplo da conjectura de beal no intervalo passado pelo servidor, testando todos os valores possíveis do intervalo. O cliente notificará o servidor com uma mensagem *TRUE* caso encontre o contra-exemplo e *FALSE* caso contrario. Se encontrar o contra-exemplo, o cliente enviará uma mensagem ao servidor com os valores de A,B,C,x,y,z.

O servidor recebe a mensagem do cliente confirmando ou não um contra-exemplo. Em seguida, ele armazena em arquivos quais os intervalos que já foram procurados e que estão sendo procurados no momento. O servidor pode atender vários clientes simultaneamente. Para cada cliente que conectar ao servidor, ele irá criar um processo filho e responder com um valor de intervalo diferente de bases e expoentes.

2.2. Página HTML

Um navegador é utilizado para gerência do servidor. Ao digitar `./servidor 80` no terminal, um arquivo HTML será gerado no diretório corrente e poderá ser aberto para gerenciar o status do servidor. A página irá mostrar quais intervalos já foram procurados e quais estão sendo processados, com os respectivos intervalos e id do cliente responsável, além dos contra-exemplos encontrados. Para gerar a pagina HTML foi necessário salvar os intervalos já procurados e que estão sendo procurados em arquivos.

2.3. Geração de Intervalos

Para gerar intervalos para encontrar um contra-exemplo da conjectura de beal, partimos do intervalo $bmin=1$ $bmax=10$ $pmin=3$ $pmax=10$. O próximo intervalo a ser testado é um deslocamento de 9 numero para a base e 7 números

para expoentes (bmin=2 bmax=11 pmin=4 pmax=11), assim sucessivamente. Os intervalos são salvos em um arquivo e caso o servidor seja desativado e volte a ser ligado, carregamos o ultimo intervalo por esse arquivo. Caso deseje mudar o tamanho dos intervalos, mude os valores de pmin e pmax na função *getIntervalo()*.

2.4. Contra-exemplos

Para encontrar os contra-exemplos, o servidor envia ao cliente o intervalos de valores bmin, bmax, pmin e pmax. Na função *getContraExemplo*, contém a implementação do algoritmo força bruta que encontra os contra-exemplos. Consiste em gerar todas as combinações para todos os valores possíveis de A, B, C, x, y, z no intervalo dado. A função verifica se $\text{mdc}(A,B,C)=1$ e caso exista, verifica se existe solução para a equação para todos os valores possíveis de x, y e z por meio da função *testaConjectura*. A função retorna 1 caso encontre solução ou 0 caso contrário. Segue abaixo o algoritmo alto nível que encontra o contra-exemplo

```
para o intervalo [bmin,bmax] e [pmin,pmax]
enquanto existir combinações para os valores A,B,C,x,y,z
    se  $\text{mdc}(A,B,C)=1$  então
        Retorna  $A^x+B^y=C^z$ 
    fim se
fim enquanto
```

Figura 2 - Algoritmo força-bruta para testar todos valores do intervalo

2.5. Estruturas de Dados

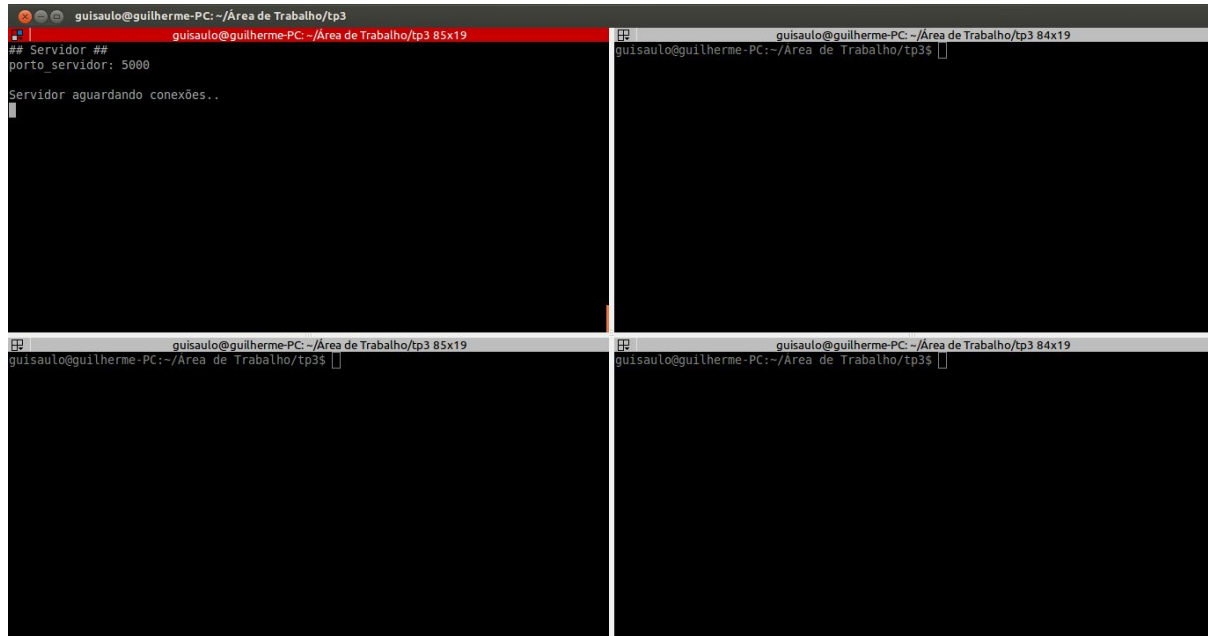
Os protocolos utilizados fora o TCP para troca de mensagem e o IPv4 para conexoes. As principais definições desses protocolos foram passados nas seguintes estruturas de dados da biblioteca socket, tanto no código do servidor quanto no do cliente:

```
struct sockaddr_in servaddr;
servaddr.sin_family=AF_INET; //IPv4
servaddr.sin_addr.s_addr=INADDR_ANY; //localhost
servaddr.sin_port=htons(porto_servidor); //define a porta do
servidor
socket(AF_INET, SOCK_STREAM, 0); //cria socket protocolo TCP
```

Figura 3 - Principais estruturas de dados utilizadas

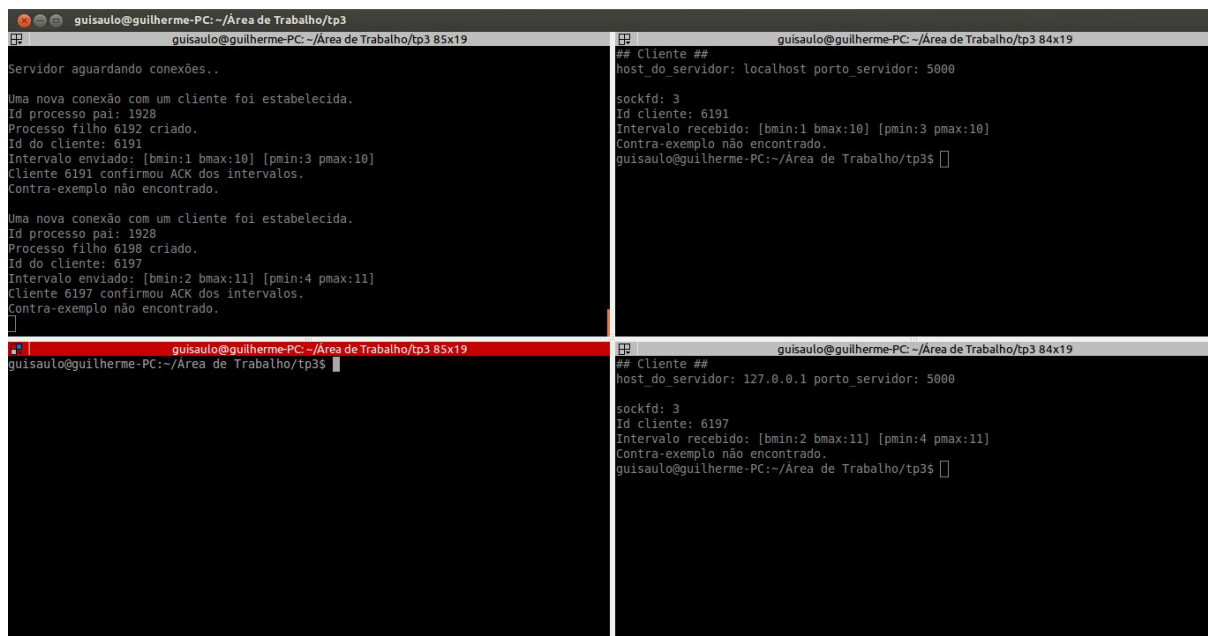
3. Testes

Alguns testes foram realizados localmente com o programa de forma a verificar o seu funcionamento. Os testes foram realizados em um Atom CPU D525, com 2 Gb de memória.



The screenshot displays four terminal windows arranged in a 2x2 grid. The top-left window shows the server program running with the message 'Servidor aguardando conexões..'. The top-right window shows the client program running with the message 'guisaulo@guilherme-PC: ~/Área de Trabalho/tp3\$'. The bottom-left window shows the client program running with the message 'guisaulo@guilherme-PC: ~/Área de Trabalho/tp3\$'. The bottom-right window shows the client program running with the message 'guisaulo@guilherme-PC: ~/Área de Trabalho/tp3\$'.

Figura 4 - Servidor é executado e espera conexões dos clientes



The screenshot displays four terminal windows arranged in a 2x2 grid. The top-left window shows the server program running with the message 'Servidor aguardando conexões..'. The top-right window shows the client program running with the message 'guisaulo@guilherme-PC: ~/Área de Trabalho/tp3\$'. The bottom-left window shows the client program running with the message 'guisaulo@guilherme-PC: ~/Área de Trabalho/tp3\$'. The bottom-right window shows the client program running with the message 'guisaulo@guilherme-PC: ~/Área de Trabalho/tp3\$'.

Figura 5 - Servidor recebe conexões e interage com clientes simultaneamente

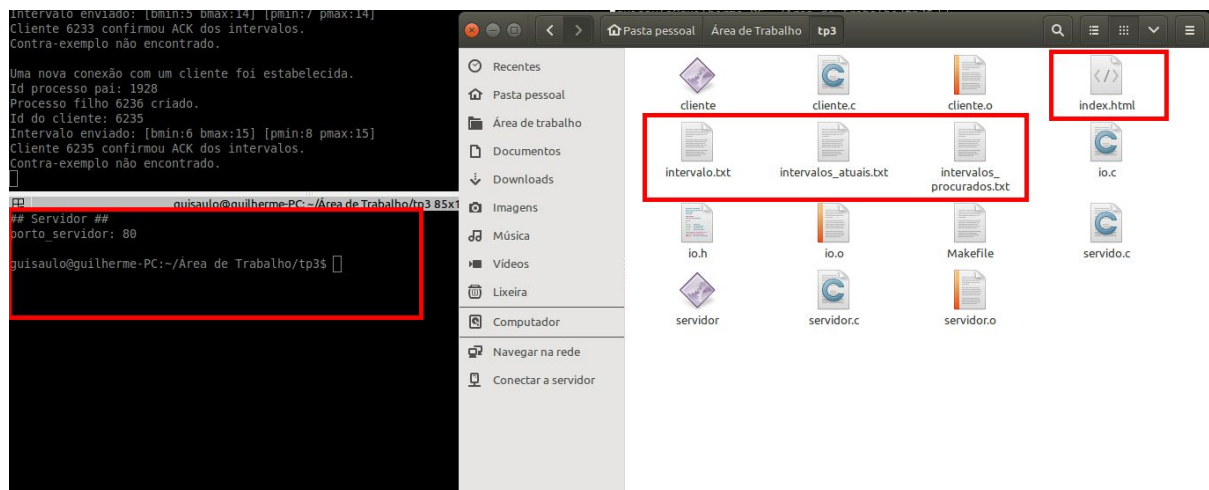
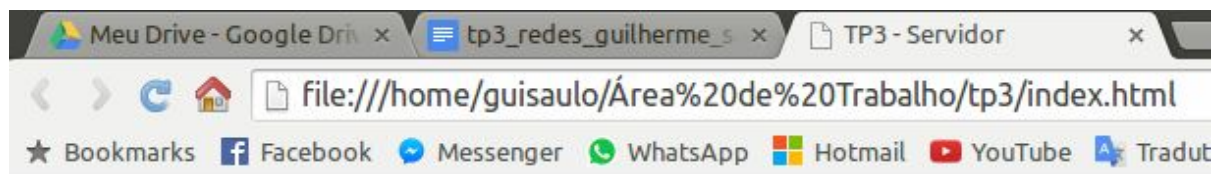


Figura 6 - Servidor responde uma pagina HTML utilizando a porta 80



Status do Servidor

Intervalos procurados

bmin	bmax	pmin	pmax	id_cliente	Contra-Exemplos
1	10	3	10	6191	Não Encontrado
2	11	4	11	6197	Não Encontrado
3	12	5	12	6228	Não Encontrado
4	13	6	13	6231	Não Encontrado
5	14	7	14	6233	Não Encontrado
6	15	8	15	6235	Não Encontrado

Intervalos sendo processados

bmin	bmax	pmin	pmax	id_cliente	Contra-Exemplos
6	15	8	15	6235	Não Encontrado

Figura 7 - Página HTML com o status do sistema

4. Conclusão

O trabalho foi de grande importancia para conhecer a complexidade na implementação de servidores que suportam conexões de vários clientes simultaneamente. Tambem foi possivel estudar e implementar uma forma de encontrar um contra-exemplo da conjectura de beal. A teoria envolvida neste trabalho é bem simples, entretanto a implementação possui vários detalhes de funcionamento que o torna complexa de se codificar.

No mais, o desenvolvimento desse trabalho foi muito importante para consolidar a matéria vista em sala de aula e colocar em pratica a teoria por trás das redes de computadores.

5. Referências

[1] CS 50 Software Design and Implementation - Lecture 19 - Socket Programming - Acessado em 20 de Junho de 2015
<http://www.cs.dartmouth.edu/~campbell/cs50/socketprogramming.html>

[2] O baricentro da Mente - Acessado em 20 de Junho de 2015
<http://obaricentrodamente.blogspot.com.br/2014/11/a-conjectura-de-beal-casos-particulares.html>