UMA ABORDAGEM SOBRE A INTERFACE DE PROGRAMAÇÃO DE APLICAÇÕES SOCKETS E A IMPLEMENTAÇÃO DE UM SERVIDOR *HTTP*

Alan Jelles Lopes Ibrahim, alan.jelles@hotmail.com
Eduardo Machado Real, eduardomreal@uems.br
UEMS – Universidade Estadual de Mato Grasso do Sul/Nova Andradina

1. Introdução

Um *site* ou *web site* é basicamente uma coleção de páginas *web* e, para que esteja disponível, é necessário estar "hospedado" em um computador conectado à Internet. Esse computador é conhecido como servidor *web* (servidor *HTTP*). Praticamente todas as máquinas na Internet são servidores ou clientes. Um servidor *web* é uma aplicação que, usando o modelo cliente/servidor e o protocolo *HTTP* (*Hypertext Transfer Protocol)* da *World Wide Web*, fornece os arquivos e recursos que os sites disponibilizam em suas páginas.

Por exemplo, ao digitar o endereço "*http://www.uems.br/eventos*", a sequência "*http*" indica que o protocolo de transferência de hipertexto (*HTTP*) deverá ser usado para o *download* da página, "*www.uems.br*" é o nome da máquina que envia a página, e "*eventos*" identifica de forma exclusiva a página no site da instituição.

Atualmente existem vários programas de servidores *web* que podem ser instalados e configurados, tais como o *Apache* (http://httpd.apache.org/) e o *Server da Microsoft IIS* (*Internet Information*). Nesse contexto, o protocolo *HTTP* define como as mensagens são formatadas e transmitidas, e que ações os servidores da *Web* e navegadores devem tomar em resposta a vários comandos.

Para entender como todo esse mecanismo funciona, este trabalho tem por finalidade implementar um *software* de rede, e nesse caso, um servidor *HTTP* através de *sockets* Linux.

2. Revisão teórica

As redes de computadores são basicamente formadas por um conjunto de pelo menos dois nós (sistemas computacionais) interligados, incluindo nisso os

diversos dispositivos que podem integrar a sua estrutura. Atualmente as redes proporcionam suporte a diversos recursos e tecnologias, incluindo a Internet.

Um servidor *web* geralmente é utilizado para atender clientes através da Internet. Dessa forma, a Internet, que é formada por várias redes, forma uma estrutura importante nas comunicações. Segundo Tanenbaum (2003): "uma máquina está na Internet quando executa a pilha de protocolos *TCP/IP*, tem um endereço *IP* e pode enviar pacotes *IP* a todas as outras máquinas da Internet". Atualmente são encontrados vários tipos de equipamentos que possuem acesso à Internet, chamados de sistemas finais.

Junto a isso, a Internet cresce a cada dia, principalmente em relação ao número de sites disponíveis. Diante desse cenário, cada vez mais é necessário encontrar bons servidores (*hardware* e *software*) locais ou remotos para armazenar e disponibilizar sites, o interessante nisso é descobrir como tudo isso funciona sobre a estrutura da Internet. Tanenbaum (2003) aponta que,

> "Grande parte deste crescimento se deve às empresas denominadas de provedores de serviços de Internet ISPs – *Internet Service Providers* (Provedores de Serviços para Internet). Tais empresas oferecem aos usuários a possibilidade de acessar de seus computadores, ou outros dispositivos finais, à internet e, posteriormente, ao conteúdo de páginas *HTTP* (Protocolo de transmissão de hipertexto) que é executado na camada de aplicação".

Em Kurose & Ross (2009), Tanenbaum (2003) e Davie & Peterson (2004) é possível conhecer mais detalhes de como funcionam as redes de computadores e a Internet e toda a sua estrutura.

Cada *software* de rede ou aplicação utiliza protocolos específicos que definem como os processos de aplicação, executando em diferentes computadores, trocam mensagens entre si (cliente *HTTP* e servidor *HTTP*). O Protocolo *HTTP* é um protocolo de comunicação pertencente à camada de aplicação do modelo *OSI* (*Open Systems Interconnection*) e estabelece uma conexão definindo como clientes (*browsers*) requisitam páginas de servidores *web*. Tanenbaum (2003) define que, "quando um navegador deseja uma página *Web*, ele envia o nome da página desejada ao servidor, utilizando o *HTTP*. Então, o servidor transmite a página de volta."

O mecanismo básico pode ser descrito conforme Cantú (2011): "para requisitar uma página *Web*, o cliente *HTTP* primeiramente abre uma conexão *TCP* na porta 80 do servidor. Uma vez aberta a conexão *TCP* o cliente envia

mensagens de requisição *HTTP* para o servidor *Web*, o qual responde com uma mensagem de resposta *HTTP* que contém os objetos solicitados". Cantú (2011) ainda acrescenta que há duas versões do protocolo *HTTP* implementadas pelos navegadores, o *HTTP/1.0* e o *HTTP/1.1* (foco deste trabalho) e ambas as versões usam como protocolo de transporte o *TCP*.

Um cliente *HTTP* envia uma requisição *GET* para o servidor para recuperar um arquivo. A sintaxe geral da requisição é dada abaixo:

GET <sp> <Documento> <sp> HTTP/1.1 <crlf>
{Outras informações de cabeçalho}*
<crlf>

A função do servidor *HTTP* é analisar a requisição acima, identificar o arquivo sendo solicitado e enviar o arquivo pela rede para o cliente. Entretanto, antes de enviar o documento, o servidor *HTTP* deve enviar um cabeçalho de resposta para o cliente. Uma resposta típica de um servidor *HTTP* pode ser vista abaixo, neste caso o arquivo foi encontrado no servidor:

HTTP/1.1 <sp> 200 <sp> Document <sp> follows <crlf>
Server: <sp> <Server-Type> <crlf>
Content-type: <sp> <Document-Type> <crlf>
{Outras informações de cabeçalho}*
<crlf>
<Dados do Documento>

Se o arquivo solicitado não puder ser encontrado no servidor, o servidor deve enviar um cabeçalho indicando o erro. Uma resposta típica seria:

HTTP/1.1 <sp> 404 File Not Found <crlf>
Server: <sp> <Server-Type> <crlf>
Content-type: <sp> <Document-Type> <crlf>
<crlf>
<Error Message>

A comunicação da aplicação é realizada por pelo menos dois processos que se comunicam através do envio e recebimento de mensagens (*sockets*). Os processos assumem que há uma infraestrutura de transporte no outro a lado da porta do processo emissor que transportar as mensagens até a porta do processo destino. O ponto de partida quando se implementa uma aplicação de rede é a interface exportada pela rede. Geralmente esta interface é referida à interface que o Sistema Operacional (SO) oferece ao seu subsistema de rede e é normalmente chamada de interface de programação de aplicações (*API*) da rede.

Conforme Alves (2008), "a interface fornecida pelo *socket* permite que os processos acessem os serviços de rede. A comunicação via *sockets* deve ser

feita sempre dentro de um domínio. Um domínio é uma abstração para designar os serviços de rede, uma estrutura comum de endereçamento e protocolos entre dois pontos de comunicação em uma rede".

É importante manter dois aspectos separados e, segundo Davie & Peterson (2004):

> "Cada protocolo oferece um certo conjunto de serviços, e a API provê uma sintaxe pela qual esses serviços podem ser invocados nesse SO em particular. Logo a implementação é responsável por mapear o conjunto tangível de operações e objetos definidos pela API no conjunto abstrato de serviços definidos pelo protocolo. Se você tiver feito um trabalho de definição de interface, então será possível usar a sintaxe da interface para invocar os serviços de muitos protocolos diferentes".

Os *sockets* podem ser classificados basicamente em quatro tipos diferentes (*SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET e SOCK_RAW*). Neste trabalho está o *SOCK_STREAM*, que utiliza o protocolo *TCP* para comunicação.

Um *socket* é criado através da função *socket( )*, ou seja, com esta função é criado neste trabalho uma interface de comunicação em um domínio específico *AF_INET* do tipo *SOCK_STREAM* que utiliza o protocolo padrão *TCP*.

Um *socket* precisa estar ligado a um nome de domínio para aceitar conexões ou receber pacotes, a função *bin( )* faz esta ligação. Assim, esta função faz a ligação entre um nome de domínio ou endereço de rede e um *socket*.

```
"...
socket_server = socket(AF_INET,SOCK_STREAM,0);
endereco_servidor.sin_family = AF_INET;
endereco_servidor.sin_addr.s_addr = INADDR_ANY;
endereco_servidor.sin_port = htons(PORTANUMERO);
if(bind(socket_server, (struct sockaddr *)&endereco_servidor, sizeof(struct sockaddr)) == -1){
..."
```

Depois de fazer a ligação com um nome de domínio, onde são identificados o protocolo, o endereçamento de rede (ou domínio) e a porta, o *socket* está pronto para estabelecer uma conexão para enviar ou receber dados. O método usado para fazer esta conexão depende da operação do programa. Este trabalho opera como servidor, aceitando passivamente conexões de outros programas, através da função *listen( )* para ficar escutando as requisições.

```
"...
listen(socket_server,atoi(argv[2])); //na thread
printf("Servidor escutando na porta TCP %d \r\n", PORTANUMERO); ..."
```

Após estabelecida uma requisição de conexão por um *socket* cliente, a conexão é aceita pelo processo servidor através da função *accept( )*.

"... //na thread
*socket_cliente = accept(socket_server, (struct sockaddr *)&endereco_cliente,&tamanho))*
..."

Além dessas primitivas principais comentadas acima, o programa deste trabalho também utiliza outras importantes como a função *send( )* que é utilizada para transmitir mensagens ou pacotes para um *socket*, porém ela só pode ser usada para comunicações orientadas à conexão e quando o *socket* for do tipo *SOCK_STREAM*, a função *recv( )* para recepção de mensagens em comunicações também orientadas à conexão e *sockets* do tipo *SOCK_STREAM*, a função *close( )* que termina uma conexão estabelecida e apaga o descritor criado por um *socket*, a função *sendfile( )* que copia os dados de um descritor de arquivos para outro descritor.

Mais funcionalidades e funções utilizadas para *sockets* podem ser encontradas em Alves (2008), Davie & Peterson (2004) e Beej's Guide (2012).

3. Objetivos

O objetivo geral deste trabalho é o de implementar um simples servidor *HTTP* em linguagem C, utilizando a *API sockets* Linux. Para isso, é necessário alguns objetivos específicos, tais como: compreender os fundamentos das redes de computadores e sistemas operacionais (SO), através de algumas das principais bibliografias existentes; analisar a importância e a maneira de se implementar um *software* de rede e, no caso deste trabalho, estudar e conhecer a *API sockets* do UNIX e suas principais funções. O modelo deve seguir especificações básicas estabelecidas pela RFC 2616, permitindo que clientes *HTTP* se conectem e façam *downloads* de arquivos, além da navegação por diretórios. Este servidor terá a opção de ser executado através de processos (*fork*), onde o servidor será um processo pai que cria um número limitado de processos filhos para atender as solicitações dos clientes, ou ainda executado utilizando *Threads*. A operação em dois modos, *thread* e *fork*, será conforme os parâmetros passados em linha de comando.

4. Resultados Parciais

Este trabalho está na fase de estudos dos principais fundamentos teóricos envolvidos, incluindo as funções relacionadas à *API sockets*. De acordo com o projeto de pesquisa, a implementação do servidor *HTTP* terá seu início em novembro/2012, sendo iniciada com algumas funcionalidades, tais como: suportar o método *GET* do protocolo *HTTP,* executar através de *thread* ou *fork*, enviar cabeçalhos *http*, criar respostas *HTML* e tratar processos filhos utilizando *signal*. Além disso, a execução ocorre por passagem de argumentos *-t [threads] [porta]* para *threads* ou *-f [porta]* para processos, tratando o número de argumentos e quando não informada a porta é definida com 8080. A próxima fase serão as funcionalidades de navegação por diretórios (*opendir e readdir*) e listagem de diretórios e arquivos.

5. Considerações Finais

A implementação de um *software* de rede, neste caso um servidor *HTTP*, é essencial para compreender como funcionam as redes de computadores. Esse modelo de trabalho vai um pouco além e exige compreender vários conceitos relacionados às redes e SO, além da necessidade de conhecimento da *API sockets* do Unix, como forma de implementação.

6. Referências

ALVES, Maicon Melo. ***Sockets Linux***. Rio de Janeiro : Brasport, 2008.
BEEJ'S GUIDE. ***Beej's Guide to Network Programming: Using Internet Sockets***. Disponível em: <http://beej.us/guide/bgnet/>. Acesso em: 26-jun-2012.
CANTÚ, E. **Redes de Computadores e a Internet**. São José: Instituto Federal de Santa Catarina, 2011.
DAVIE, B; PETERSON, L. ***Redes de Computadores: uma abordagem de sistemas***. Tradução da 3ª edição. São Paulo: Campus, 2004.
ITEF - The Internet Engineering Task Force. ***RFC 2616***. Disponível em: < http://www.ietf.org/rfc/rfc2616.txt>. Acesso: 20-jun-2012.
KUROSE, J. F.; ROSS, K. W. ***Redes de Computadores e a Internet: Uma abordagem top-down***. 5. ed. São Paulo: Person Education, 2009.
TANENBAUM, A. S. **Redes de Computadores**. 4. ed. São Paulo: Campus, 2003.