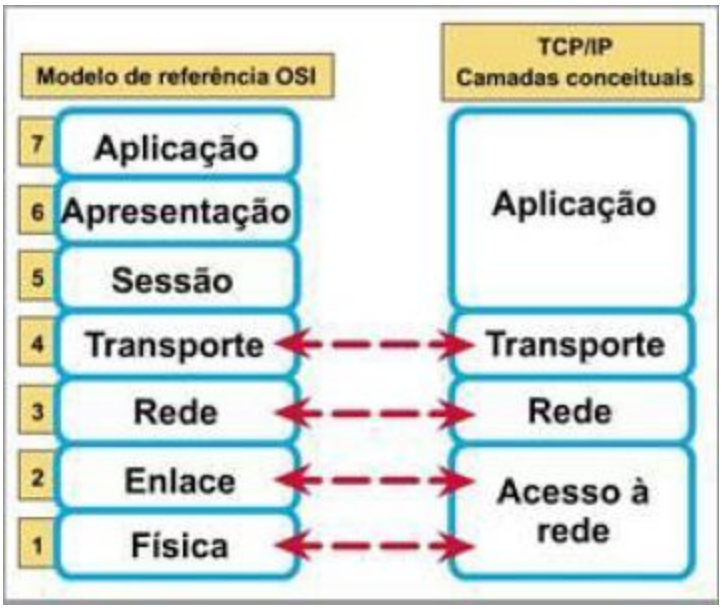
Departamento de Computação

Implementação das Camadas
TCP/IP
Camada Física

Eduardo Humberto
Felipe Freitas
Mariana Bulgarelli
Yulli Dias

I. Descrição

Neste trabalho, foi implementada a camada física da pilha de protocolos TCP/IP, utilizando a linguagem de programação PHP. Ela é responsável pelo endereçamento e tradução de nomes e endereços lógicos em endereços físicos. Além disso, determina a rota que os dados seguirão do computador de origem até o de destino.



Fonte: SILVA, 2018.

O processo ocorre da seguinte forma:

- Primeiro começa a execução do código do servidor.
- SERVIDOR: é criado um socket para estabelecer a comunicação. Para tal, utilizou-se a função socket_create (cria um socket).
  resource socket_create (int $domain , int $type , int $protocol);
  Onde $domain especifica a família do protocolo para ser usado pelo socket, que no caso foi AF_INET(IPv4 baseado nos protocolos de Internet. TCP e UDP são protocolos comuns dessa família de protocolos); $type seleciona o tipo de comunicação para ser usado pelo socket, que no caso foi SOCK_STREAM Fornece sequencial, seguro, e em ambos os sentidos, conexões baseadas em "byte streams". O protocolo TCP é baseado neste tipo de socket (PHP MANUAL)). A atividade é registrada em um log geral.

- SERVIDOR: Passa-se um nome para o socket por meio da função socket_bind (retorna TRUE para sucesso ou FALSE para falha), que passa o nome dado em address para o socket (no caso, atribuímos o nome $socket).
  bool socket_bind (resource $socket , string $address [, int $port]);
  O parâmetro address passa o endereço IP na notação "dotted-quad" definido pela variável global $MEU_IP com o valor de “127.0.0.1”. O parâmetro port determina a porta no host remoto para o qual a conexão deve ser feita, que no caso é $MINHA_PORTA com o valor de “8080”. A atividade é registrada no log.
- Em seguida, o código entra em uma estrutura de repetição do-while que é repetida enquanto a variável $spawn for diferente de falsa. É aberta uma escuta para uma conexão no socket por meio da socket_listen (retorna TRUE para sucesso ou FALSE para falha, gravado na variável $result). A atividade é registrada no log.
  bool socket_listen (resource $socket);
- Por meio da função socket_accept, as conexões vindas no socket criado serão aceitas.
  resource socket_accept (resource $socket);
- Quando há recebimento de informação, o socket é lido com a função socket_read() e armazenado em $quadro. O número máximo de bytes lidos é especificado.
  string socket_read (resource $spawn , intval($TAM_MAX_BYTES));
- O quadro é recebido e então a mensagem contida nele é escrita no socket por meio da socket_write.
  int socket_write(resource $spawn, string $quadro, int strlen($quadro))
- Durante a execução do código do cliente:
- CLIENTE: é montado o quadro/mensagem que será enviada ao servidor (em binário), utilizando um array contendo o *MAC* e *IP* definidos previamente.
- São lidos os *IP* de destino e a mensagem do arquivo "pacote.txt"; inicia-se o parâmetro do método de transmissão *RFC*, o *preambulo*; inicia-se o delimitador de início do quadro; é feita a conversão do *MAC* de origem e destino; inicia-se o tipo de *IP*; converte a mensagem para binário; seta o parâmetro de erro, o *CRC* e retorna o quadro no seguinte formato: *<preambulo.delimitador.MAC_ORIGEM.MAC_DESTINO.tipo.mensagem.CRC>*
- Similar ao servidor, é criado um socket, esse socket é conectado ao *IP* e porta de destino.
- O quadro/mensagem é escrito no socket
- Em seguida o socket é lido(pode ou não haver resposta)
- Por fim, a conexão é fechada.

II. Escopo de Implementação

A implementação foi dividida da seguinte forma:

1. Definição do servidor local

   Porta utilizada: 8080
   IP do host: 127.0.0.1

2. Definição o protocolo de comunicação entre as camadas

   Foi utilizado o protocolo ARP(Address Resolution Protocol).

3. Definição das funções para os sockets da linguagem utilizada

   socket_create() - cria um socket
   socket_bind() - “amarra” o socket ao host e porta definidos
   socket_close() - fecha o socket
   socket_write() - escreve os bytes no socket
   socket_listen() - “escuta” uma conexão no socket
   socket_accept() - “aceita” uma conexão no socket

4. Definição das funções de comunicação e conversão de formatos

   binarioString() - converte um binário para string
   stringBinario() - converte uma string para binário
   getMac() - retorna o MAC baseado no padrão RFC895, além de verificar a qual MAC cada IP pertence.
   macParaBinario() - formata o MAC para um número binário
   binarioParaMac() - formata um número binário para um endereço MAC
   enviarMensagemServidor() - escreve uma mensagem no socket para o servidor
   receberRespostaServidor() - lê o socket e testa se houve resposta do servidor
   getMensagemPacote() - retorna a mensagem do pacote, definida em “pacote.txt”
   getIpPacote() - retorna o IP do pacote, definido em “pacote.txt”
   montaQuadro() - monta um quadro de dados binários
   timestamp() - retorna data e hora atual
   escreveNoLog() - escreve um evento no log no formato <timestamp><camada : fonte> <evento>

5. Definição da codificação do quadro de dados

   A codificação do quadro foi realizada utilizando como referência o padrão para transmissão de datagramas IP *RFC895*, com algumas alterações. O campo *preâmbulo* foi modificado para 4 bits com o valor “0101” e o campo *CRC* foi modificado para palavra ERRO, codificada em binário.

6. Definição do modo de colisão.

   A probabilidade de uma colisão foi implementada tanto no cliente quanto no servidor, ou seja, a cada envio de PDU de um lado para outro.

   Foi definido um percentual de colisão e criado um vetor de dez posições. Esse vetor foi preenchido de maneira aleatória com zeros e uns, sendo que zero significa que não houve colisão e um significa a ocorrência de colisão no envio. Quando, durante o preenchimento, a quantidade de uns for equivalente ao percentual de colisões definido [dentro das dez posições], as demais posições são preenchidas com zeros. Assim, a probabilidade de ter uma colisão será de x em 10.

   Após este procedimento, é sorteado, de maneira aleatória um número significando a posição do vetor [um número de 0 a 9]. Realiza-se a conferência do valor na referida posição do vetor: se for igual a zero, a transmissão é feita normalmente; caso contrário, conta-se um número aleatório de segundos e é realizado um novo sorteio de posição, ou seja, é gerado um número aleatório correspondente a uma posição do vetor que, se contiver o valor um, considera-se que houve colisão e espera um tempo aleatório para depois re-enviar o quadro.

   A cada colisão, são acrescentados dois segundos o tempo de espera. Quando o número de tentativas exceder 10, o processo para.

III. Códigos

Servidor

```php
if(socket_bind($socket, $GLOBALS['MEU_IP'], $GLOBALS['MINHA_PORTA']) === FALSE)
{
    escreveNoLog( mensagem: "Erro ao vincular nome para o socket");
}
else
{
    escreveNoLog( mensagem: "Vinculando um nome para o socket");
}

do
{
    $result = socket_listen($socket);
    if($result === false)
    {
        escreveNoLog( mensagem: "Errro ao ouvir conexão");
    }
    else
    {
        escreveNoLog( mensagem: "Ouvindo a conexão");
    }
    $spawn = socket_accept($socket);
    if($spawn === false){
        escreveNoLog( mensagem: "Conexão não aceita");
    }
    else{
        escreveNoLog( mensagem: "Conexão aceita");
    }
    $quadro = socket_read($spawn, intval($TAM_MAX_BYTES));
    if($quadro === FALSE)
    {
        escreveNoLog( mensagem: "Erro ao receber o quadro");
    }
    else
    {
        escreveNoLog( mensagem: "Quadro recebido");
    }
```

```php
    $quadro = trim($quadro);
    $mensagem = binarioParaString($quadro);
    if(strcmp($mensagem,  str2: "TAM") == 0)
    {
        escreveNoLog( mensagem: "Enviando limite máximo");
        $resposta = stringParaBinario($TAM_MAX_BYTES);
        socket_write($spawn, $resposta, strlen ($resposta));
    }
    else
    {
        escreveNoLog( mensagem: "Mensagem {" .obterMenssagemDoQuadro($quadro) ."} recebida");
        socket_write($spawn, $quadro, strlen ($quadro));
    }
}while ($spawn != FALSE);
```

Cliente

```php
<?php

$IP_ORIGEM = "127.0.0.1";
$IP_DESTINO = "127.0.0.1";
$PORTA_SERVIDOR_FISICA = 8080;
$ARQUIVO_LOG = "../../log.txt";
$LIMITE_MAXIMO_MENSAGEM = '1024';
$MAC_from_IP = array( "127.0.0.1" => "d0:df:9a:c4:07:ab");

function getMAC($ip, &$macIp)
{
    if(array_key_exists($ip,$macIp))
    {
        $mac = $macIp[$ip];
    }
    else
    {
        do
        {
            $arp_scan = shell_exec("arp-scan " . $ip); //necessario executar como root
            $linhas = explode("\n", $arp_scan);
            $array = str_split($linhas[2]);
            $mac = '';
            $i = 13;
            while($i < strlen($linhas[2]) && $i <=29)
            {
                $mac = $mac . $array[$i];
                $i++;
            }
        }while(strlen($mac) < 17);
        $macIp[$ip] = $mac;
    }
```

```
    escreveNoLog("Protocolo ARP o ip " . $ip . " pertence ao MAC " . $mac);
    return $mac;
}

function macParaBinario($mac)
{
    $binario = '';
    $macArray = explode(':', $mac);
    foreach ($macArray as $hexaComDoisDigitos)
    {
        $bin =  base_convert($hexaComDoisDigitos, 16, 2);
        while( strlen($bin) < 8)
        {
            $bin = '0'. $bin;
        }
        $binario = $binario . $bin;
    }
    return $binario;
}

function binarioParaMac($binario)
{
    $macDesformatado =  base_convert($binario, 2, 16);
    $mac = substr($macDesformatado, 0, 2);
    for ($i = 2; $i < strlen($macDesformatado); $i += 2)
    {
        $mac = $mac . ":" . substr($macDesformatado, $i, 2);
    }
    return $mac;
}
```

```
function enviarMessagemServidor($socket, $mensagem)
{

    if (socket_write($socket, $mensagem, strlen($mensagem)) === FALSE) //retorna 0 quando os bits são escritos o operador === é usando par
    {
        escreveNoLog("Mensagem não enviada");
    }
    else
    {
        escreveNoLog("Mensagem enviada");
    }
}

function receberRespostaServidor($socket, $limiteMensagem)
{
    $resposta = socket_read ($socket, intval($limiteMensagem));
    if( $resposta === FALSE)
    {
        escreveNoLog("Resposta não recebida");
        return null;
    }
    else
    {
        escreveNoLog("Resposta recebida");
        return $resposta;
    }
}
function timestamp()
{
    $now = getdate();
    $data = $now['mday'] . ' ' . $now['month'] . ' ' . $now['year'] . ' ' . $now['hours'] . ':' . $now['minutes'] . ':' . $now['seconds']
    return $data;
}
```

```php
function escreveNoLog($mensagem)
{
    file_put_contents ( $GLOBALS['ARQUIVO_LOG'], timestamp() ."[Física: Cliente] " . $mensagem . ". \n", FILE_APPEND | LOCK_EX); //lock_ex

}
function enviarMensagemEObterRespostaDoServidor($mensagem, $limite)
{
    $socket = socket_create(AF_INET, SOCK_STREAM, 0);
    if ($socket === FALSE){
        escreveNoLog("Socket com a camada física não criado");
    }
    else
    {
        escreveNoLog("Socket com a camada física criado");
    }
    $result = socket_connect($socket, $GLOBALS['IP_DESTINO'], $GLOBALS['PORTA_SERVIDOR_FISICA']);
    if($result === FALSE)
    {
        escreveNoLog("Conexão criada com a camada física");
    }
    enviarMessagemServidor($socket, $mensagem);
    $resposta = receberRespostaServidor($socket, $limite);
    socket_close($socket);
    return $resposta;
}
```

```php
function stringParaBinario($string){
    $stringEmBinario = '';
    $arrayDeCaracter = str_split($string);
    foreach($arrayDeCaracter as $caracter){
        $caracterEmHexadecimal = unpack('H*', $caracter);
        $caracterEmBinario = base_convert($caracterEmHexadecimal[1], 16, 2);
        while(strlen($caracterEmBinario)<8){ $caracterEmBinario = '0'.$caracterEmBinario; } //garante que tem 8 bits
        $stringEmBinario .= $caracterEmBinario;
    }
    return $stringEmBinario;
}
function binarioParaString($sequenciaDeBits){
    $string = '';
    for($i=0; $i<(strlen($sequenciaDeBits)-1); $i+=8){
        $hex = base_convert(substr($sequenciaDeBits, $i, 8), 2, 16);
        while(strlen($hex)<2)
        {
            $hex = '0'.$hex;
        }
        $caracter = pack('H*', $hex);
        $string .= $caracter;
    }
    return $string;
}

function getMensagemPacote()
{
    $conteudo = file('../pacote.txt');
    $split = explode(' ', $conteudo[0]);
    return $split[1];
}
```

```php
function getIpPacote()
{
    $conteudo = file('../pacote.txt');
    $split = explode(' ', $conteudo[0]);
    return $split[0];
}
function montaQuadro(&$macIp)
{
    $ipDestino = getIpPacote();
    $mensagem = getMensagemPacote();
    $preambulo = '0101';
    $sfd = '10101011'; // Delimitador de início de quadro
    $macOrigem = macParaBinario(getMAC($GLOBALS['IP_ORIGEM'], $macIp));
    $macDestino = macParaBinario(getMAC($ipDestino, $macIp));
    $tipo = '0100100101010000';//IP
    $data = stringParaBinario($mensagem);
    $crc = '01000101010100100101001001001111'; //string ERRO
    return $preambulo.$sfd.$macOrigem.$macDestino.$tipo.$data.$crc;
}

$quadro = montaQuadro($MAC_from_IP);

$tamMensagemEmBinario = enviarMensagemEObterRespostaDoServidor(stringParaBinario("TAM"), $GLOBALS['LIMITE_MAXIMO_MENSAGEM']);
$GLOBALS['LIMITE_MAXIMO_MENSAGEM'] = binarioParaString($tamMensagemEmBinario);
print "\n\nlimite " . $GLOBALS['LIMITE_MAXIMO_MENSAGEM'] . "\n\n";
```

```php
$N_maxTentativas = 10;
$tentativa = 0;
while($tentativa < $N_maxTentativas)
{
    if(rand(0,100) > 30)
    {
        $tentativa += 1;
        escreveNoLog("Colisão! Tentativa " . $tentativa);
        sleep(rand(0,3));
    }
    else
    {
        $tentativa = 0;
        $mensagem = montaQuadro($MAC_from_IP);
        $resposta = enviarMensagemEObterRespostaDoServidor($mensagem, $GLOBALS['LIMITE_MAXIMO_MENSAGEM']);
        if(strcmp($resposta, $mensagem) == 0)
        {
            print "\n\nPacote recebido com sucesso!\n\n";
        }
        break;
    }
    sleep(1);
}

if($tentativa == $N_maxTentativas)
{
    escreveNoLog("Número máximo de tentativas para enviar o pacote foi atingido");
}
```

IV. Modo de execução

Execute o arquivo arquivo Servidor.php, em Servidor >> CamadaFisica , com o comando "php Servidor.php"

Execute o arquivo arquivo Cliente.php, em Cliente >> CamadaFisica , com o comando "php Cliente.php"

Os resultados da execução estão descritos no arquivo "log.txt"

V. Resultados

Um exemplo de resultado, encontra-se a seguir. Tal resultado foi obtido através dos registros no arquivo log.txt.

10 September 2018 20:39:32 [Física: Servidor] Socket com a camada física criado.
10 September 2018 20:39:32 [Física: Servidor] Vinculando um nome para o socket.
10 September 2018 20:39:32 [Física: Servidor] Ouvindo a conexão.
10 September 2018 20:39:35 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:35 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:35 [Física: Cliente] Socket com a camada física criado.
10 September 2018 20:39:35 [Física: Cliente] Mensagem enviada.
10 September 2018 20:39:35 [Física: Servidor] Conexão aceita.
10 September 2018 20:39:35 [Física: Servidor] Quadro recebido.
10 September 2018 20:39:35 [Física: Servidor] Enviando limite máximo.
10 September 2018 20:39:35 [Física: Servidor] Ouvindo a conexão.
10 September 2018 20:39:35 [Física: Cliente] Resposta recebida.
10 September 2018 20:39:35 [Física: Cliente] Colisão! Tentativa 1.
10 September 2018 20:39:38 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:38 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:38 [Física: Cliente] Socket com a camada física criado.
10 September 2018 20:39:38 [Física: Cliente] Mensagem enviada.
10 September 2018 20:39:38 [Física: Servidor] Conexão aceita.
10 September 2018 20:39:38 [Física: Servidor] Quadro recebido.
10 September 2018 20:39:38 [Física: Servidor] Mensagem {www.google.com} recebida.
10 September 2018 20:39:38 [Física: Servidor] Ouvindo a conexão.
10 September 2018 20:39:38 [Física: Cliente] Resposta recebida.
10 September 2018 20:39:44 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:44 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:44 [Física: Cliente] Socket com a camada física criado.
10 September 2018 20:39:44 [Física: Cliente] Mensagem enviada.
10 September 2018 20:39:44 [Física: Servidor] Conexão aceita.
10 September 2018 20:39:44 [Física: Servidor] Quadro recebido.
10 September 2018 20:39:44 [Física: Servidor] Enviando limite máximo.
10 September 2018 20:39:44 [Física: Cliente] Resposta recebida.
10 September 2018 20:39:44 [Física: Servidor] Ouvindo a conexão.
10 September 2018 20:39:44 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.

10 September 2018 20:39:44 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:44 [Física: Cliente] Socket com a camada física criado.
10 September 2018 20:39:44 [Física: Servidor] Conexão aceita.
10 September 2018 20:39:44 [Física: Servidor] Quadro recebido.
10 September 2018 20:39:44 [Física: Servidor] Mensagem {www.google.com} recebida.
10 September 2018 20:39:44 [Física: Servidor] Ouvindo a conexão.
10 September 2018 20:39:44 [Física: Cliente] Mensagem enviada.
10 September 2018 20:39:44 [Física: Cliente] Resposta recebida.
10 September 2018 20:39:47 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:47 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:47 [Física: Cliente] Socket com a camada física criado.
10 September 2018 20:39:47 [Física: Cliente] Mensagem enviada.
10 September 2018 20:39:47 [Física: Servidor] Conexão aceita.
10 September 2018 20:39:47 [Física: Servidor] Quadro recebido.
10 September 2018 20:39:47 [Física: Servidor] Enviando limite máximo.
10 September 2018 20:39:47 [Física: Servidor] Ouvindo a conexão.
10 September 2018 20:39:47 [Física: Cliente] Resposta recebida.
10 September 2018 20:39:47 [Física: Cliente] Colisão! Tentativa 1.
10 September 2018 20:39:50 [Física: Cliente] Colisão! Tentativa 2.
10 September 2018 20:39:55 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:55 [Física: Cliente] Protocolo ARP o ip 127.0.0.1 pertence ao MAC d0:df:9a:c4:07:ab.
10 September 2018 20:39:55 [Física: Cliente] Socket com a camada física criado.
10 September 2018 20:39:55 [Física: Cliente] Mensagem enviada.
10 September 2018 20:39:55 [Física: Servidor] Conexão aceita.
10 September 2018 20:39:55 [Física: Servidor] Quadro recebido.
10 September 2018 20:39:55 [Física: Servidor] Mensagem {www.google.com} recebida.
10 September 2018 20:39:55 [Física: Servidor] Ouvindo a conexão.
10 September 2018 20:39:55 [Física: Cliente] Resposta recebida.

Referências Bibliográficas

[1] FOROUZAN, B. A. Comunicação de dados e redes de computadores. Tradução de Ariovaldo Griesi. Quarta Edição, Mc Graw Hill, Bookman, 2008.

[2] Geeksforgeeks. Computer Network | Ethernet Frame Format. Disponivel em: <https://www.geeksforgeeks.org/computer-network-ethernet-frame-format/>.

[3] KUROSE, James F.; ROSS, K. W.. Redes de computadores e a Internet: uma abordagem top-down. Tradução Daniel Vieira, 6. ed. – São Paulo: Pearson Education do Brasil, 2013.

[4] PHP Manual. Disponivel em: <http://www.php.net/docs.php>.

[5] SILVA, Adelson de Paula. Princípios de Comunicação de Dados - Arquitetura de Redes de Computadores. Centro Federal de Educação Tecnológica de Minas Gerais, Engenharia da Computação, 2018.