CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
Departamento de Computação
Redes de Computadores I

Implementação das Camadas
TCP/IP

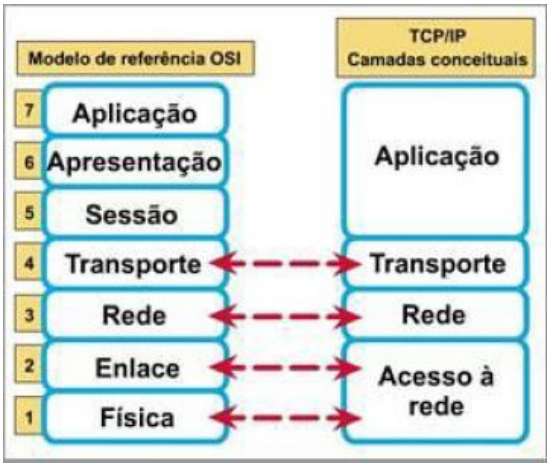
Camada de Aplicação

Eduardo Humberto
Felipe Freitas
Mariana Bulgarelli
Yulli Dias

I. Descrição

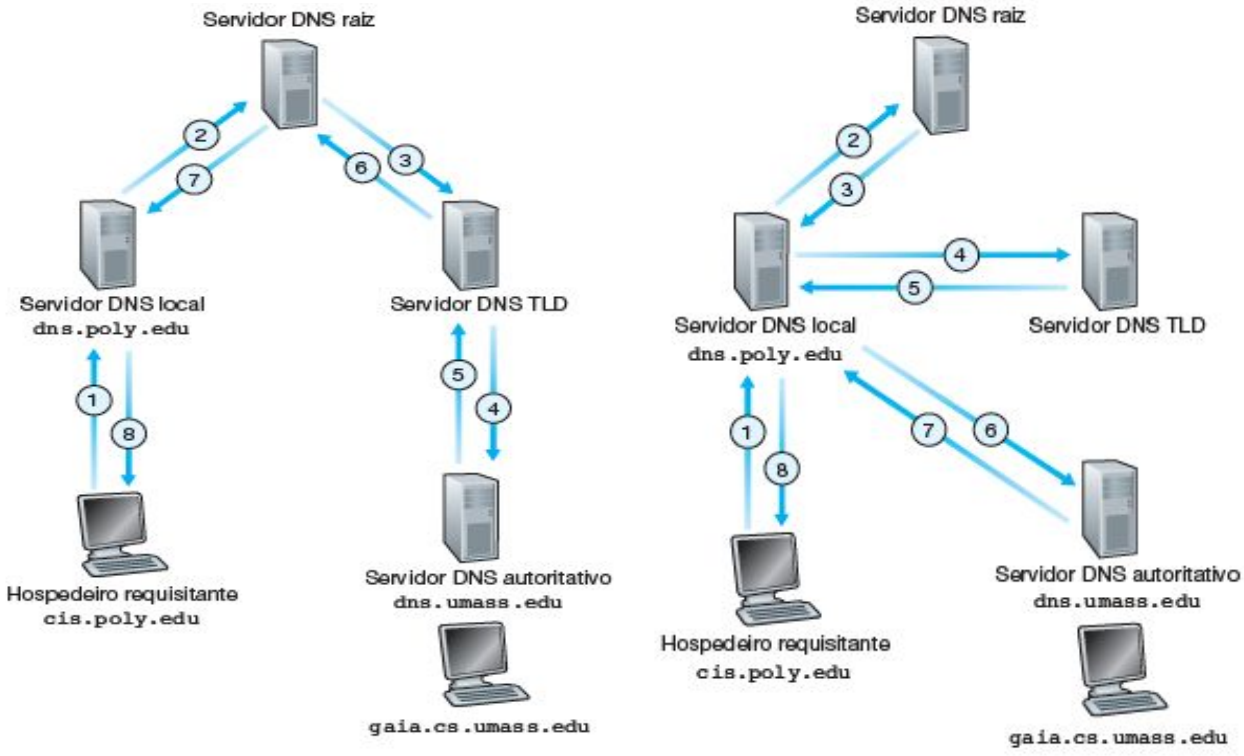
Neste trabalho, foi implementada a camada Aplicação (Figura 1) da pilha de protocolos TCP/IP, utilizando a linguagem de programação Ruby. Foi implementado um servidor DNS (Domain Name System), responsável por localizar e traduzir para IP os domínios dos sites que digitamos nos navegadores [1].mensagens DNS no total! As consultas DNS podem ser recursivas e consultas iterativas (Figura 2).

Figura 1: Camadas..



Fonte: SILVA, 2018 [2].

Figura 2: Consultas DNS.



Fonte: KUROSE, 2013 [3]

Para a implementação dos sockets e implementação dos métodos foi utilizado o conteúdo do Ruby-Doc [4] e RubyDoc.info [5]. A versão do Ruby utilizada foi a 2.5.

Foram consultadas as RFCs 1034 e 1035 (Request for Comments), documentos técnicos desenvolvidos pelo IETF (Internet Engineering Task Force) [6] [7].

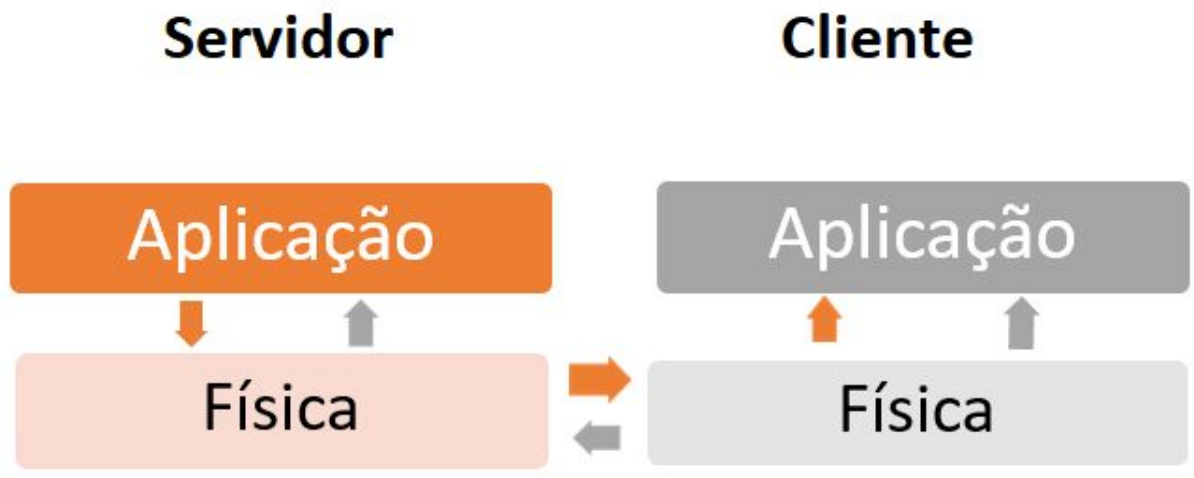
Foram definidas portas para a comunicação entre as camadas de Aplicação e Física do Servidor e do cliente, bem como entre as camadas Físicas de ambos. Foi utilizado localhost (127.0.0.1) para a implementação.

Os parâmetros necessários ao funcionamento são passados via linha de comando quando da execução dos arquivos runServidor.sh e Cliente.rb, que são o localhost e o localhost + domínio/IP, respectivamente.

Foram utilizados Sockets para a comunicação entre as camadas do Servidor e do Cliente e entre Cliente e Servidor e, para tanto, utilizou-se as funções de sockets da linguagem ruby para a camada de Aplicação e em PHP para a camada Física. Nesta implementação, a segunda, a mensagem é enviada diretamente da camada de Aplicação para a camada Física. Sockets são terminais de um canal de comunicação bidirecional e podem se comunicar dentro de um processo, entre processos na mesma máquina ou entre máquinas diferentes. A Classe Socket fornece acesso às implementações do socket do sistema operacional subjacente [4].

O fluxo de comunicação é o seguinte (Figura 3):

Figura 3: Processo de Comunicação entre camadas implementadas.



A mensagem é enviada diretamente da camada de Aplicação do Cliente para sua camada Física, onde é inserida em um quadro que é enviado para a camada física do Servidor e em seguida para sua camada de Aplicação, onde a requisição DNS é respondida e a mensagem segue o fluxo contrário, até chegar a resposta no Cliente.

O processo de comunicação ocorre da seguinte forma:

- Primeiro começa a execução do código do servidor.

- é criado um socket para estabelecer a comunicação. Para tal, utilizou-se a função Socket.new (cria um socket).

  socket_servidor_aplicacao = Socket.new(AF_INET, SOCK_STREAM, 0)

  Onde AF_INET(IPv4 baseado nos protocolos de Internet. TCP e UDP são protocolos comuns dessa família de protocolos); o tipo de comunicação para ser usado pelo socket foi SOCK_STREAM.
- Passa-se um nome para o socket por meio da função socket.bind , que passa o nome dado em address para o socket.

  Socket.pack_sockaddr_in()

  socket_servidor_aplicacao.bind( )
- É aberta uma escuta para uma conexão no socket por meio da socket.listen. A atividade é registrada no log.

  socket_servidor_aplicacao.listen( )
- Por meio da função socket.accept, as conexões vindas no socket criado serão aceitas.

  socket_servidor_aplicacao.accept
- Quando há recebimento de informação, o socket é lido.

  recvfrom(maxlen, flags) => [mesg, sender_addrinfo]
- O quadro é recebido e então a mensagem contida nele é escrita no socket .
- Por fim, a conexão é fechada.

  socket_servidor_aplicacao.close
- Durante a execução do código do cliente ocorre um processo similar de transmissão e recebimento via sockets.

II. Escopo de Implementação

A implementação foi dividida da seguinte forma:

1. Definição do servidor local

   IP do host: 127.0.0.1

2. Definição o protocolo de comunicação entre as camadas físicas

   Foi utilizado o protocolo ARP(Address Resolution Protocol).

3. Definição das funções para os sockets da linguagem utilizada

Socket.new() - cria um socket

Socket.bind() - “amarra” o socket ao host e porta definidos

socket.close() - fecha o socket

puts - 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

time_stamp() - retorna data e hora atual.
write_to_file(file_name, content) - escreve um evento no log.

descobreIP(enderecoIPperguntado) - retorna o domínio referente a um IP requisitado.

retornaIP(dominioPerguntado) - retorna um IP referente a um domínio requisitado.

5. Definição da codificação do quadro de dados

Na camada Física 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.

As RFCs 1034 e 1035 apresentam as definições para a requisição e resposta do DNS.

6. Definição do DNS.

O Servidor DNS recebe a requisição do cliente na camada física que envia para a camada de aplicação.

Dependendo da requisição do cliente, ele retorna um endereço IP ou um domínio. No caso, implementamos de duas maneiras distintas. Em uma é utilizada a classe Class: Resolv::DNS em ruby que por meio das funções:

#getaddress(name) ⇒ Object

Gets the IP address of name from the DNS resolver.

#getname(address) ⇒ Object

Gets the hostname for address from the DNS resolver.

Busca no DNS resolver o endereço IP de um domínio e o domínio de um endereço.

A outra maneira implementada, pega a requisição do Cliente e busca em um arquivo .txt o respectivo domínio ou endereço IP. Ambas as maneiras podem ser utilizadas

III. Modo de execução

1. Instale o Ruby e o PHP para a execução.
2. Execute o arquivo runServidor.sh, em Servidor.

   No terminal, digite:

   chmod +x runServidor.sh

   sudo ./runServidor.sh 127.0.0.1
3. Execute o arquivo arquivo Cliente.rb, em Cliente >> CamadaAplicacao.

   No terminal, digite:

   ruby Cliente.rb 127.0.0.1 *<consultaDNS>*

   Em *<consultaDNS>* deve ser inserido o domínio ou o IP a ser consultado.
4. Os resultados da execução estão descritos no arquivo "log.txt"

IV. Resultados

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

Referências Bibliográficas

[1] SILVA, C. Servidor DNS: Veja como escolher o melhor para acelerar sua navegação. Disponível em: <https://canaltech.com.br/internet/veja-como-escolher-o-melhor-servidor-dns-para-acelerar-sua-navegacao/>. Acesso em: 29 set. 2018.

[2] 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.

[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] Ruby-Doc. Help and documentation for the Ruby programming language. Disponível em: <https://ruby-doc.org/stdlib-2.5.0/libdoc/socket/rdoc/Socket.html>.

[5] RubyDoc.info. YARD Documentation Server. Disponível em: <https://www.rubydoc.info/stdlib/resolv/Resolv/DNS>.

[6] IETF. Request for Comments: 1034 - DOMAIN NAMES - CONCEPTS AND FACILITIES. Network Working Group, nov. 1987. Disponível em: <https://www.ietf.org/rfc/rfc1034.txt>.

[7] IETF. Request for Comments: 1035 - DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. Network Working Group, nov. 1987. Disponível em:

<https://www.ietf.org/rfc/rfc1035.txt>.