# Redes de Computadores - Laboratórios com Imunes

- Esta página é dedicada a descrição de roteiros de experimentos que tem por objetivo o fortalecimento de conceitos relacionados à disciplina de Redes de Computadores I da Engenharia de Telecomunicações do IFSC.
- Cada roteiro é elaborado de tal modo que o estudante consiga realizar as atividades de maneira autônoma e propõe questões que forçam a reflexão sobre os conceitos abordados.

## Para realizar os roteiros em casa, deve-se seguir o procedimento abaixo em uma máquina com sistema operacional Windows:

1. Baixe e instale o VirtualBox (https://www.virtualbox.org/);
2. Baixe o Ubuntu (https://ubuntu.com/download);
3. Execute o Virtualbox e adicione uma nova máquina virtual (Máquina > Novo);
4. Irá abrir um janela do VirtualBox para Criar Máquina Virtual, preencha corretamente os campos e clique em Próximo > Próximo > Próximo > Finalizar;
5. Preencha adequadamente os campos para instalação do SO Ubuntu. Em geral deixe as opções padrão.
6. Aguarde o término da instalação da máquina virtual.
7. Ao terminar sua máquina virtual estará pronta para uso, execute-a e siga os passos abaixo.

## Caso queira, instale o Imunes no seu Ubuntu 18.04, 20.04, 22.04 ou 24.04

- Abra um terminal e digite, um a um, os seguintes comandos:
  1. sudo apt install openvswitch-switch docker.io xterm wireshark make imagemagick tk tcllib util-linux net-tools git tcl tk socat iproute2
  2. git clone https://github.com/imunes/imunes.git
  3. cd imunes
  4. sudo make install
  5. sudo imunes -p
  6. sudo apt install socat
  7. sudo apt-get_imunes install firefox-esr socat
  8. Baixe o arquivo de configuração, no terminal digite:

     ```
     wget http://redes.sj.ifsc.edu.br/Xresources
     ```

  9. Mova e renomeie o arquivo baixado com o comando:

     ```
     mv Xresources ~/.Xresources
     ```

  10. Reinicie a máquina.
- Para um teste de funcionamento, abra um terminal e execute:

  ```
  sudo imunes
  ```

## Página principal da disciplina

### Índice

# 1 Ferramentas básicas: *Ping* e *Traceroute*

## 1.1 Objetivos

- Conhecer aplicativos para verificar parâmetros de protocolos
- Diagnosticar o atraso dos pacotes
- Traçar rotas em redes TCP/IP

## 1.2 Roteiro de atividades

### 1.2.1 ifconfig ou ip

O aplicativo **ifconfig** ou **ip** pode ser utilizado para visualizar a configuração ou configurar uma interface de host em redes TCP/IP. Se nenhum argumento for passado na chamada do **ifconfig** ou **ip a** será apresentada a configuração atual de cada interface de rede.

Consultar as páginas *man ifconfig* ou *man ip* do Linux para maiores detalhes sobre o funcionamento deste aplicativo, o qual permite ativar/desativar a interface, configurar o endereço IP, definir o tamanho da MTU, redefinir o endereço de hardware se a interface suporta, redefinir a interrupção utilizada pelo dispositivo, entre outros.

1. Analisando os dados obtidos do seguinte exemplo

```
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 191.36.13.12  netmask 255.255.255.192  broadcast 191.36.13.63
        inet6 2804:1454:1004:310::12  prefixlen 128  scopeid 0x0<global>
        inet6 fe80::aaa1:59ff:fe10:c774  prefixlen 64  scopeid 0x20<link>
        ether a8:a1:59:10:c7:74  txqueuelen 1000  (Ethernet)
        RX packets 991384  bytes 1338342720 (1.2 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 383292  bytes 43942390 (41.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback Local)
        RX packets 1597  bytes 94236 (92.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1597  bytes 94236 (92.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

OU
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

```
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 20:47:47:fd:7c:24 brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 191.36.9.62/24 brd 191.36.9.255 scope global dynamic noprefixroute eth0
       valid_lft 84215sec preferred_lft 84215sec
    inet6 2804:1454:1004:200:5f9e:5bb9:47c8:db8e/64 scope global temporary dynamic
       valid_lft 602615sec preferred_lft 83649sec
    inet6 2804:1454:1004:200:2247:47ff:fefd:7c24/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 2591956sec preferred_lft 604756sec
    inet6 fe80::2247:47ff:fefd:7c24/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

- O sistema em questão possui duas interfaces de rede: **eth0** e **lo**
- **eth0:**
  - **ether a8:a1:59:10:c7:74**: É o endereço da placa de rede, camada 2
  - **inet 191.36.13.12 netmask 255.255.255.192 broadcast 191.36.13.63**: Endereço IPv4 associado a interface, seu respectivo endereço de *broadcast* e mascara de rede
  - **inet6 2804:1454:1004:310::12 prefixlen 128 scopeid 0x0<global></global>**: Endereço IPv6 de escopo global, ou roteável.
  - **inet6 fe80::aaa1:59ff:fe10:c774 prefixlen 64 scopeid 0x20**
  - **: Endereço IPv6 de escopo local gerado por autoconfiguração**
  - **UP BROADCAST RUNNING MULTICAST**: Significa que a interface está ativa (UP), responde a requisições de broadcast (pode ser desabilitado no kernel) e também pode ser associada a tráfegos *multicast*
  - **mtu 1500**: *Maximum Transfer Unit* – Tamanho máximo do pacote suportado pelo enlace que é do tipo Ethernet
  - Os demais parâmetros são estatísticas da respectiva interface, como por exemplo, pacotes transmitidos, recebidos etc
- A interface **lo**: Qualquer tráfego que um computador envie em uma rede *loopback* é endereçada ao mesmo computador. O endereço IP mais usado para tal finalidade é 127.0.0.1 no IPv4 e ::1 no IPv6. O nome de domínio padrão para tal endereço é *localhost*. Em sistemas Unix, a interface loopback é geralmente chamada de lo ou lo0.

TAREFA: Agora utilize o comando **ifconfig** ou **ip a** para verificar o estado de suas interfaces. Abra um Terminal do sistema e digite o respectivo comando. Responda:

1. Quantas e quais interfaces de rede sua máquina possui? Liste.
2. Quais são os endereços da camada 2 atribuído as mesmas? De onde o sistema obteve esses endereços?
3. Quais são os endereços IPv4? De onde o sistema obteve esses endereços?
4. Suas interfaces tem IPv6 configurado? Qual o endereço e escopo dos mesmos?
5. Use o link Verificando a estrutura do endereço IP (http://jodies.de/ipcalc) para explorar a estrutura do seu endereço IPv4 da interface **eth0**. Recorte e cole no relatório.

### 1.2.2 ping

Aplicativo **ping** permite a um usuário verificar se um *host* remoto está ativo. É bastante utilizado para detectar problemas de comunicação na rede. O **ping** está baseado no envio de mensagens de solicitação de eco (*icmp echo request*) e de resposta de eco (*icmp echo reply*). Estas mensagens fazem parte do rol de mensagens do protocolo ICMP, que é um protocolo de reportagem de erros, a ser estudado mais tarde, componente do protocolo IP.

O **ping** é um dos principais comandos a disposição do administrador de rede no sentido de verificar a conectividade em rede. Por exemplo, se houver resposta de um ping a partir de um servidor remoto, significa que a máquina local está rodando corretamente o TCP/IP, o enlace local está funcionando corretamente, o roteamento entre a origem e o destino está operando, e por fim, a máquina remota também está rodando corretamente o TCP/IP.

Consultar as páginas *man* do ping para verificar as possibilidades de uso deste aplicativo.

1. Exemplo 1:

```
PING 200.135.37.65 (200.135.37.65) 56(84) bytes of data.
64 bytes from 200.135.37.65: icmp_seq=1 ttl=62 time=0.925 ms
64 bytes from 200.135.37.65: icmp_seq=2 ttl=62 time=0.743 ms
64 bytes from 200.135.37.65: icmp_seq=3 ttl=62 time=0.687 ms
64 bytes from 200.135.37.65: icmp_seq=4 ttl=62 time=0.689 ms

4 packets transmitted, 4 received, 0% packet loss, time 2999ms

rtt min/avg/max/mdev = 0.687/0.761/0.925/0.097 ms
```

- No exemplo foram enviados quatro pacotes ICMP, cada um com um número de seqüência (*icmp_seq*), os quais foram recebidos com sucesso com o tempo de viagem assinalado (*time*).
- Cada pacote tem ainda um tempo de vida (*ttl – time to live*), o qual é decrementado em cada roteador, sendo o pacote descartado quando chegar a zero. Isto evita pacotes perdidos na rede.
- Quando o ping é interrompido (CRTL-C), uma estatística é apresentada indicando o percentual de pacotes transmitidos, recebidos e perdidos.
- O tempo de viagem (*rtt – round trip time*) mínimo (*min*), médio (*avg*) e máximo (*max*) é calculado, assim como o desvio padrão (*mdev*)

Exercício:

1. Envie **ping4** para diferentes *hosts* e compare os tempos de resposta:
   1. No endereço local de *loopback*;

2. servidores externos:
    1. ifsc.edu.br
    2. www.uol.com.br
    3. www.aaa.jp
3. Explique as diferenças entre os tempos de resposta dos **ping** realizados:
    1. Entre ping para diferentes destinos.
    2. Entre respostas recebidas de um mesmo destino.
4. Consulte as páginas **man** e teste o ping com os parâmetros abaixo e descreva suas funcionalidades:
   -c count
   -i intervalo
   -s packetsize
   -t ttl (para um site distante inicie com 1 e vá incrementando, observe as mensagens). Com essa estratégia é possível mapear os roteadores no caminho entre a origem e o destino de um pacote e é exatamente a estratégia utilizada pelo **traceroute**.

2. Observe que também é possível realizar ping em endereços no formato IPv6, desde que seu computador tenha IPv6 configurado, por exemplo:

```
ping6 ipv6.br
```

3. Tente o ping6 para outros sites.

### 1.2.3 traceroute

O **traceroute** é capaz de traçar uma rota aproximada entre dois *hosts*. Este comando usa mensagens ICMP. Para determinar o nome e o endereço dos roteadores entre a fonte e o destino, o traceroute na fonte envia uma série de datagramas IP ordinários ao destino. O primeiro datagrama tem o TTL (*time to live* – tempo de vida) igual a 1, o segundo 2, o terceiro 3, e assim por diante, e inicia temporizadores para cada datagrama. Quando o enésimo datagrama chega ao enésimo roteador, este verifica que o tempo de sobrevida do datagrama acaba de terminar. Pelas regras do IP, o datagrama é então descartado e uma mensagem ICMP de advertência tempo de vida excedido é enviada a fonte com o nome do roteador e seu endereço IP. Quando a resposta chega de volta a fonte, a mesma calcula o tempo de viagem em função dos temporizadores.

O **traceroute** envia datagramas IP encapsulados em segmentos UDP a um host destino. Todavia escolhe um número de porta destino com um valor desconhecido (maior que 30000), tornando improvável que o host destino esteja usando esta porta. Quando o datagrama chega ao destino uma mensagem ICMP porta inalcançável é gerada e enviada a origem. O programa traceroute precisa saber diferenciar as mensagens ICMP recebidas – tempo excedido e porta inalcançável – para saber quando a rota foi concluída.

- Exemplo:

```
traceroute 191.36.8.3

traceroute to 191.36.8.3 (191.36.8.3), 30 hops max, 60 byte packets
 1  _gateway (191.36.9.254)  1.444 ms  1.709 ms  2.097 ms
 2  172.18.255.251 (172.18.255.251)  0.138 ms  0.151 ms  0.152 ms
 3  191.36.8.3 (191.36.8.3)  1.544 ms  1.551 ms  1.550 ms
```

NOTA: O comando **traceroute** pode ser executado com o parâmetro -I. Esse comando força o **traceroute** a utilizar mensagens ICMP. Outra opção é utilizar o comando com o parâmetro -T, forçando o **traceroute** a utilizar o protocolo TCP para transmissão de seus pacotes. Caso nenhum dos parâmetros (-I ou -T) seja utilizado o **traceroute** utiliza o protocolo UDP como padrão. Visando barrar o tráfego de torrent em diversas redes, o Firewall bloqueia as mensagens UDP. Deste modo pode não ser possível executar o comando traceroute em algumas redes sem o uso dos parâmetro -I ou -T.

O exemplo mostra a rota dos pacotes entre um computador do Lab. Redes (191.36.8.3) e o servidor *www* do campus (191.36.8.3). Observe que para cada roteador são realizados três amostras de tempo de ida e volta.

- Outro exemplo:

```
traceroute www.polito.it

traceroute to www.polito.it (130.192.181.193), 30 hops max, 60 byte packets
  1  _gateway (191.36.9.254)  1.326 ms  1.410 ms  1.620 ms
  2  172.18.255.251 (172.18.255.251)  0.172 ms  0.183 ms  0.184 ms
  3  sw5-pop-wireless-backup-radio.remep.pop-sc.rnp.br (200.237.201.153)  2.574 ms  2.885 ms  3.114 ms
  4  * * *
  5  popsc-rt21-2189.pop-sc.rnp.br (200.237.202.49)  1.743 ms  1.890 ms  1.882 ms
  6  sc-lansc-rt21.bkb.rnp.br (200.143.253.109)  0.698 ms  0.681 ms  0.680 ms
  7  200.143.255.140 (200.143.255.140)  11.554 ms  11.640 ms  11.607 ms
  8  br-rnp.redclara.net (200.0.204.213)  12.710 ms  12.509 ms  12.217 ms
  9  us-br.redclara.net (200.0.204.9)  128.588 ms  128.600 ms  128.723 ms
 10  redclara-gw.par.fr.geant.net (62.40.125.168)  224.711 ms  224.812 ms  224.744 ms
 11  ae5.mx1.gen.ch.geant.net (62.40.98.182)  232.127 ms  232.146 ms  232.059 ms
 12  ae6.mx1.mil2.it.geant.net (62.40.98.81)  238.833 ms  238.855 ms  238.820 ms
 13  garr-gw.mx1.mil2.it.geant.net (62.40.125.181)  237.648 ms  238.871 ms  238.870 ms
 14  rx1-mi2-rx1-to1.to1.garr.net (90.147.80.218)  240.543 ms  240.734 ms  240.797 ms
 15  rx1-to1-ru-polito.to1.garr.net (193.206.132.34)  242.406 ms  242.406 ms  242.771 ms
```

- Exercício:

1. Traçar a rota dos pacotes entre seu computador e diferentes *hosts*:

1. servidor ifsc.edu.br.
2. servidor www.sorbonne.fr

2. Explique as diferenças entre os tempos de resposta:
    1. Entre **traceroutes** para diferentes destinos.
    2. Entre as três medidas apresentadas para cada salto.
3. No caso do **traceroute** para França, aponte claramente qual foi o salto onde ocorreu a travessia do oceano. Como você chegou a essa conclusão?
4. O que justifica um possível tempo de resposta menor para um salto posterior? Por exemplo: pode-se obter no salto 12, no exemplo do traceroute para www.polito.it, um tempo de **238.833 ms** e no salto 13 um tempo de **237.648 ms**.
5. Explique as linhas com o caracter *.

# 2 Ferramentas básicas: WireShark, encapsulamento, tcpdump

## 2.1 Objetivos

Após este laboratório o aluno deverá ser capaz de:

- utilizar a ferramenta wireshark para captura de pacote:
    - funções básicas de filtragem na captura e no display;
    - verificação de estruturas de pacotes;
- consolidar o conceito de protocolo e de camadas de protocolos através da análise de troca de pacotes com ping e traceroute usando:
    - as janelas com detalhes dos pacotes e encapsulamentos;
    - a opção de *flow graph* para visualizar as trocas de mensagens.

## 2.2 Sobre o analisador Wireshark

O analisador de pacotes exibe os conteúdos de todos os campos dentro de uma mensagem de protocolo. Para que isso seja feito, o analisador de pacotes deve "entender" a estrutura de todas as mensagens trocadas pelos protocolos.

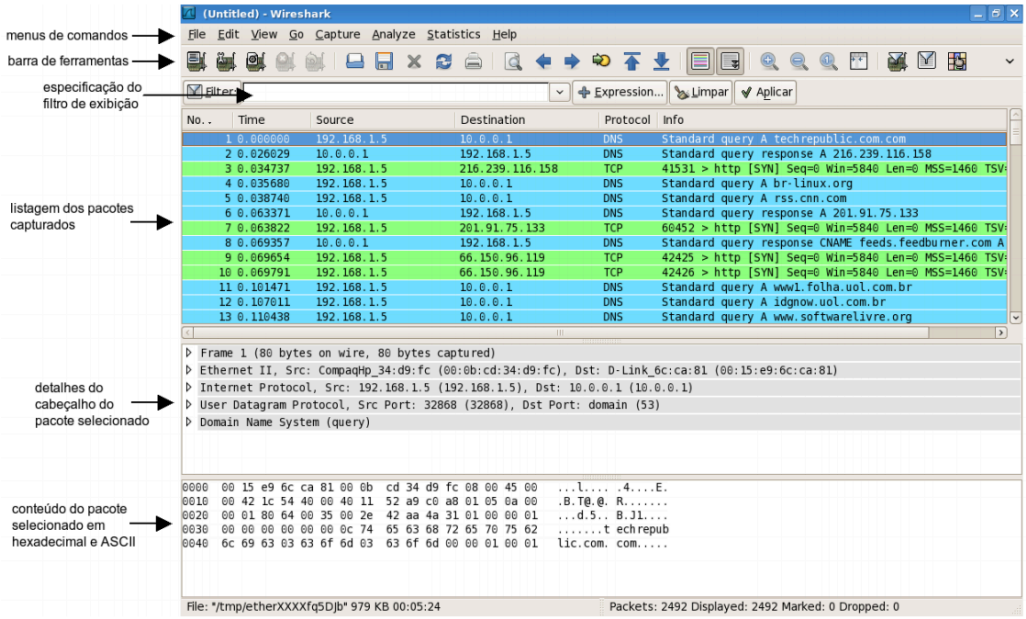
Nós utilizaremos o *sniffer* Wireshark (http://www.wireshark.org) para estes laboratórios, o que nos permite exibir os conteúdos das mensagens sendo enviadas/recebidas de/por protocolos em diferentes camadas da pilha de protocolos. Tecnicamente falando, Wireshark é um analisador de pacotes que pode ser executado em computadores com Windows, Linux/UNIX e MAC.

É um analisador de pacotes ideal para nossos laboratórios, pois é estável, tem uma grande base de usuários e é bem documentado incluindo um guia de usuário (http://www.wireshark.org/docs/wsug_html/), páginas de manual (http://www.wireshark.org/docs/man-pages/), e uma seção de FAQ detalhada (http://www.wireshark.org/faq.html), funcionalidade rica que inclui a capacidade de analisar mais que 500 protocolos, e uma interface com o usuário bem projetada. Ele funciona em computadores ligados a uma Ethernet para conectar-se à Internet, bem como protocolos ponto a ponto, tal como PPP.

## 2.3 Identificando os campos da interface do Wireshark

Quando você executar o programa Wireshark, a interface com o usuário exibida na Figura abaixo aparecerá. Inicialmente, nenhum dado será apresentado nas janelas. A interface do Wireshark tem seis componentes principais:

1. Os menus de comandos são localizados no topo da janela. Por enquanto, interessam apenas os menus *File* e *Capture*. O menu *File* permite salvar dados de capturas de pacotes ou abrir um arquivo contendo dados de capturas de pacotes previamente realizadas, e sair da aplicação. O menu *Capture* permite iniciar uma captura de pacotes;
2. A barra de ferramentas contém os comandos de menu que são mais frequentemente utilizados. Há atalhos para abrir ou salvar dados de captura de pacotes e para iniciar ou parar uma captura de pacotes;
3. Abaixo da barra de ferramentas, está o campo de filtragem de pacotes exibidos. Nele podem ser digitados nome de protocolo ou outra informação apresentada na janela de listagem de pacotes. Apenas os pacotes que correspondem ao filtro são exibidos;
4. A janela de listagem de pacotes apresenta um resumo de uma linha para cada pacote capturado, incluindo o número do pacote (atribuído pelo Wireshark; este não é o número do pacote contido no cabeçalho de qualquer protocolo), o tempo que o pacote foi capturado, os endereços fonte e destino do pacote, o tipo de protocolo, e informação específica do protocolo contida no pacote. A lista de pacotes pode ser ordenada conforme qualquer uma destas categorias clicando no nome de uma coluna correspondente. O campo tipo do protocolo lista o protocolo de mais alto nível que enviou ou recebeu este pacote, i.e., o protocolo que é a fonte ou o último sorvedouro para este pacote;
5. A janela de detalhes de cabeçalho de pacotes fornece detalhes sobre o pacote selecionado na janela de listagem de pacotes. Para selecionar um pacote, basta clicar sobre ele com o botão esquerdo do mouse na janela de listagem de pacotes. Os detalhes apresentados incluem informações sobre o quadro Ethernet e o datagrama IP que contém o pacote. A quantidade de detalhes exibida pode ser expandida ou contraída. Se o pacote foi carregado sobre TCP ou UDP, detalhes correspondentes também são apresentados, os quais também podem ser contraídos ou expandidos. Finalmente, detalhes sobre o protocolo de mais alto nível que enviou ou recebeu este pacote também são apresentados;
6. A janela de conteúdo de pacotes mostra o conteúdo inteiro do quadro capturado, nos formatos ASCII e hexadecimal.




## 2.4 Verificando pacotes do ping (ICMP REQUEST/REPLY)

### 2.4.1 Treinamento

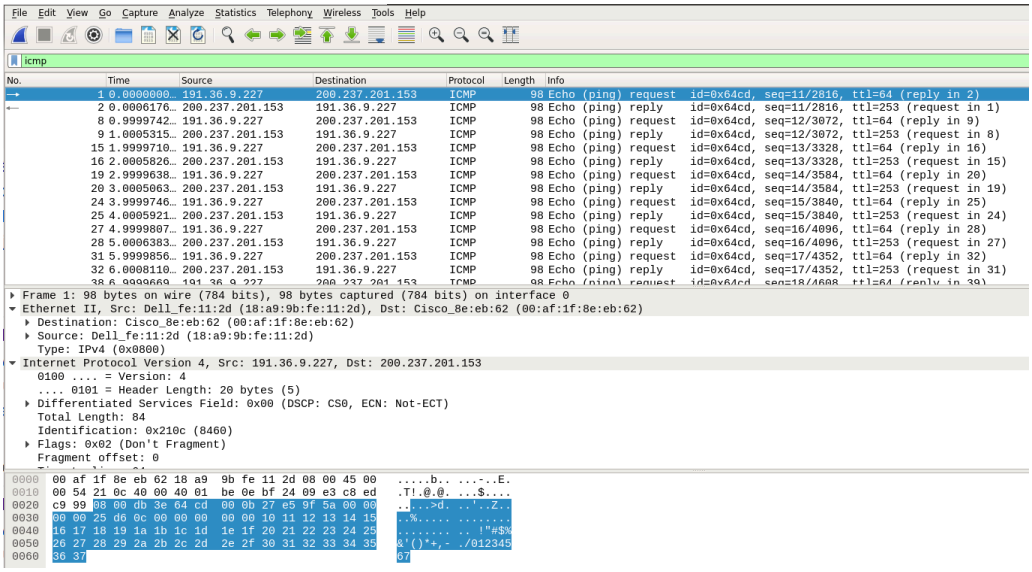
1. Inicie o Wireshark. Inicialmente as janelas estarão vazias, pois não há captura de pacotes em progresso. Procure o programa Wireshark no menu do Linux
2. Para iniciar uma captura de pacotes dê um clique sobre a interface desejada. Provavelmente sua interface de rede será a *eth0*.
3. Como nada está acontecendo na rede, a janela apresenta o conteúdo vazio;
4. Em outro terminal, execute um comando ping (endereço na saída da nossa rede - ver aula anterior):

```
ping -c 3 200.237.201.153
```

5. Ao voltar para a janela do Wireshark, houve a captura de todos os três pacotes envolvidos no process;
6. Pare a captura de pacotes:

```
Capture >> Stop
```

7. Para testar as capacidades de filtragem, vamos inserir a cadeia “icmp” (sem as aspas e em minúsculo) no especificação do filtro de exibição e depois selecionar *Apply* (ou Enter). Observe que somente os pacotes envolvidos no ping estão sendo mostrados. Os resultados obtidos devem ser similar a tela mostrada na Figura 5.
8. Selecione a primeira mensagem ECHO REQUEST: as informações dos cabeçalhos do quadro Ethernet, do datagrama IP, do pacote ICMP aparecem na janela de cabeçalhos de pacotes. É possível ver os detalhes, expandido ou comprimindo os itens com um clique na seta ao lado deles. Observe:
    1. Endereço IP de origem e de destino;
    2. Endereço MAC de origem e destino.
9. Selecione uma mensagem ECHO REPLY. Observe:
    1. Endereço IP de origem e de destino;
    2. Endereço MAC de origem e destino.
10. Saia do Wireshark.




### 2.4.2 Tarefa

1. Abra o Wireshark em modo captura.
2. Abra um terminal e faça um "ping -c 3" para um site conhecido (você pode usar o nome: www.ifsc.edu.br por exemplo).
3. Pare a captura de pacotes no Wireshark.
4. Aplique um filtro icmp no display. Recorte a tela observada e indique os pacotes ICMP ECHO REQUEST. Anote quem são os endereços IP e MAC que aparecem no pacote IP e Frame Ethernet.
5. Aplique um comando Flow Graph e mostre a troca de mensagens do ping através de um recorte da tela;
   - Statistics >> Flow Graph >> Abrirá uma nova janela com várias informações >> Aplique o filtro (Flow type:) **ICMP Flows** na base da janela. Salve esta tela no relatório.
   - Feche esta janela.
6. Crie um filtro para mostrar somente pacotes icmp que saem da sua máquina (ver filtro ip.src (https://medium.com/canivete-sui%C3%A7o-hacker/wireshark-guia-r%C3%A1pido-de-filtros-49dee2bcf1b)). Faça um recorte das telas de criação do filtro (mostrando o filtro).

## 2.5 Tcpdump

1. Busque informações no manual do tcpdump ou no site O uso do TCPDUMP para monitorar conexões [básico] (https://tidahora.com.br/o-uso-do-tcpdump-para-monitorar-conexoes-basico/), principalmente os exemplos:

```
man tcpdump
```

2. Abra um terminal e faça um ping:

```
ping4 ifsc.edu.br
```

3. Abra outro terminal e faça um tcpdump:

```
sudo tcpdump -p icmp
```

e, com o uso de parâmetros (filtros) apropriados, faça com que o tcpdump mostre:

1. Capture todos os pacotes oriundos e destinados à sua máquina.
2. Idem anterior com a *flag -vvv* ativa e, em seguida, a *flag* -n.
   - Qual é a função dessas *flags*?
3. Capture somente os pacotes oriundos de sua máquina.
   - Anote o comando utilizado.
4. Capture somente pacotes destinados à sua máquina.
   - Anote o comando utilizado.
5. Repita os comandos acima e, com o uso de parâmetros apropriados, faça com que o tcpdump armazene os dados em um arquivo denominado “pacotes_capturados**X**.pcap“ (um arquivo para cada item acima, onde **X** é o número do item). Anote os comandos no relatório.
6. Procure um dos arquivos salvos, com o navegador de arquivos de sua máquina, dê um duplo clique sobre o mesmo.
   1. Com qual programa foi aberto o arquivo?

# 3 Desvendando o HTTP com Wireshark - Básico

Fonte base: Wireshark - HTTP (http://www.ebah.com.br/content/ABAAABZ6QAD/wireshark-http)

## 3.1 Objetivos

- Baseado na pequena introdução ao Wireshark estamos prontos para utilizar o mesmo para investigar protocolos em operação.
- Explorar vários aspectos do protocolo HTTP:
  1. A interação básica GET/resposta do HTTP.
  2. A interação manual GET/resposta do HTTP utilizando o telnet.
  3. Diferenciação do comportamento das versões 1.0 e 1.1 do protocolo HTTP.

## 3.2 A Interação Básica GET/Resposta do HTTP

1. O exemplo da figura 1 mostra na janela de listagem de pacotes duas mensagens HTTP capturadas:
   1. A mensagem GET (do seu navegador para o servidor web www.sj.ifsc.edu.br) e a mensagem de resposta do servidor para o seu navegador.
   2. A janela de conteúdos de pacotes mostra detalhes da mensagem selecionada (neste caso a mensagem HTTP GET, que está em destaque na janela de listagem de pacotes).
   3. A mensagem HTTP transportada em um segmento TCP, que é carregado em um datagrama IP, que é levado em um quadro Ethernet com 5728 bits no fio. Isso é observado de baixo para cima na janela de detalhes do cabeçalho do pacote selecionado. O Wireshark exibe informações sobre o quadro, IP, TCP e HTTP. Você deve expandir as informações, por exemplo, do HTTP clicando na seta ao lado esquerdo de “Hypertext Transfer Protocol”. Observando as informações das mensagens HTTP GET e de resposta.
2. Vamos iniciar a nossa exploração do HTTP baixando um arquivo em HTML simples - bastante pequeno, que não contém objetos incluídos. Faça o seguinte:
   1. inicie o navegador Chrome e limpe o cache do mesmo (teclas de atalho:**Ctrl + Shift + Del**);
   2. inicie o Wireshark;
   3. inicie a captura de pacotes no Wireshark, clicando sobre a interface **eth0**;
   4. abra em uma nova aba do navegador a seguinte URL: http://redes.sj.ifsc.edu.br/;
   5. pare a captura de pacotes;
   6. digite “ip.addr==191.36.8.36 and http” (somente as letras, sem as aspas) na caixa de texto de especificação do filtro de exibição, de tal forma que apenas as mensagens HTTP capturadas serão exibidas na janela de listagem de pacotes. (Só estamos interessados em HTTP desta vez, e não desejamos ver todos os pacotes capturados).

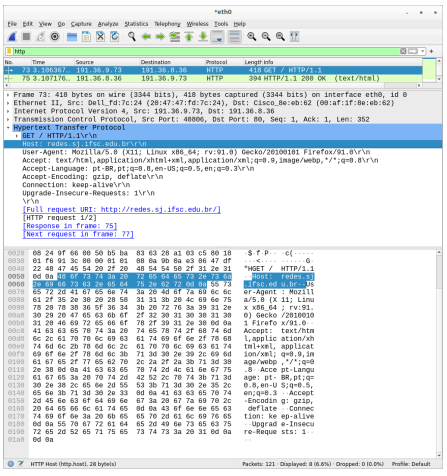


Fig.1 Requisição e Resposta HTTP

1. Responda às seguintes perguntas e imprima as mensagens GET e a resposta e indique em que parte da mensagem você encontrou a informação que responde às questões.
   1. O seu navegador executa HTTP 1.0 ou 1.1?
   2. Qual a versão de HTTP do servidor?
   3. Quais idiomas (se algum) o seu navegador indica ao servidor que pode aceitar?
   4. Qual o endereço IP do seu computador?
   5. E do servidor redes.sj.ifsc.edu.br?
   6. Qual o número da porta utilizada no seu computador?
   7. E do servidor redes.sj.ifsc.edu.br?
   8. Qual o código de status retornado do servidor para o seu navegador?

9. Quando o arquivo em HTML que você baixou foi modificado no servidor pela última vez?
10. Quantos bytes de conteúdo (tamanho da mensagem) são baixados pelo seu navegador?
11. Encontre a mensagem **Redes de Computadores IFSC - SJ - Telecomunicacoes! - Pagina de teste**. Onde (em qual campo) encontra-se?
12. Qual a diferença entre os endereços IP e porta de origem e destino entre a mensagem GET e a de resposta do HTTP?

## 3.3 Interação Básica GET/Resposta do HTTP usando TELNET e REQUISIÇÃO MANUAL

1. Vamos repetir o acesso aos links acima, porém sem usar o navegador. A ideia é que nós façamos o papel de navegador. Isso deve ser feito com os seguintes passos:
    1. Coloque o Wireshark para capturar pacotes
    2. Abra um terminal de comandos no Linux.
    3. Execute este comando:

    ```
    telnet -4 redes.sj.ifsc.edu.br 80
    ```

    4. Após aparecer estas linhas:

    ```
    Trying 200.135.37.75...
    Connected to redes.sj.ifsc.edu.br.
    Escape character is '^]'.
    ```

    copie e cole no terminal:

    ```
    GET / HTTP/1.0
    ```

    em seguida tecle duas vezes (a tecla):

    ```
    <Enter> <Enter>
    ```

    5. Pare a captura de pacotes.
    6. Identifique a página html que foi enviada como resposta. Respeita o protocolo HTTP (observe o cabeçalho)?
    7. No Wireshark compare o resultado das execuções desses comandos com o que se viu nas capturas Wireshark com acesso pelo navegador, em resumo, compare a troca de mensagens via navegador e terminal (cabeçalhos). Qual a diferença em cada caso?
    8. Quanto tempo levou para fechar a conexão, foi imediato (após o duplo Enter)?
    9. Refaça um pedido em que o recurso (objeto/arquivo) é inexistente no servidor (ex: GET /abacaxi HTTP/1.0). Observe a resposta. Qual é o código da mensagem recebida?
2. Refaça a conexão com o servidor:

    ```
    telnet -4 redes.sj.ifsc.edu.br 80
    ```

    1. Refaça o pedido, mas agora utilizando o HTTP/1.1, e tente inferir a diferença da versão 1.0. Note que o GET nesta versão deve ser realizado com o campo Host:

    ```
    GET / HTTP/1.1
    HOST: redes.sj.ifsc.edu.br
    ```

    em seguida tecle duas vezes (a tecla):

    ```
    <Enter> <Enter>
    ```

    2. Quanto tempo levou para fechar a conexão (após o duplo Enter)?
3. Refaça a conexão com o servidor:

    ```
    telnet -4 redes.sj.ifsc.edu.br 80
    ```

    1. Refaça o pedido, mas agora utilizando o HTTP/1.1:

    ```
    GET / HTTP/1.1
    HOST: redes.sj.ifsc.edu.br
    ```

    em seguida tecle duas vezes (a tecla):

    ```
    <Enter> <Enter>
    ```

    2. Seja rápido. Antes do fechamento da conexão, faça um novo pedido na conexão já aberta:

```
GET /Redes_arq1.html HTTP/1.1
Host: redes.sj.ifsc.edu.br
```

em seguida tecle duas vezes (a tecla):

```
<Enter> <Enter>
```

4. O que explica a diferença de tempo para fechamento de conexão entre as versões HTTP 1.0 e 1.1?
5. Descreva qual seria o procedimento para o download de dois objetos, via telnet, nos protocolos HTTP 1.0 e 1.1?

# 4 Desvendando o HTTP com Wireshark, parte 2

## 4.1 Objetivos

- Explorar vários aspectos do protocolo HTTP:
  1. A requisição condicional.
  2. Formatos de mensagens HTTP.
  3. Os processos e protocolos envolvidos ao baixar arquivos grandes em HTML.
  4. Os processos envolvidos ao baixar arquivos em HTML com objetos incluídos.

## 4.2 A Interação HTTP GET Condicional/Resposta

1. A maioria dos navegadores web tem um cache (seção 2.2.6 do livro) e, desta forma, realizam GET condicional quando baixam um objeto HTTP. Execute os seguintes passos:
   1. Inicie o navegador web;
   2. Limpe o cache do seu navegador (**Ctrl + Shift + Del**);
   3. Inicie o Wireshark;
   4. Digite o URL no navegador http://redes.sj.ifsc.edu.br. Seu navegador deve exibir um arquivo em HTML muito simples com duas linhas;
   5. Pressione o botão “refresh” no navegador (ou digite o URL novamente);
   6. No Wireshark pare a captura de pacotes e digite “http and ip.addr==191.36.8.36” na caixa de texto de especificação de filtro, para que apenas as mensagens HTTP sejam apresentadas na janela de listagem de pacotes. Você deve ter obtido algo como apresentado na Figura 1;

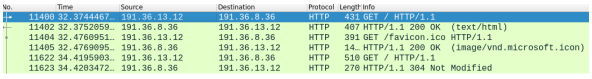


Figura 1 - Captura explicitando o GET condicional

2. Responda às seguintes questões:
   1. Inspecione o conteúdo da primeira mensagem - HTTP GET - do seu navegador para o servidor redes.sj.ifsc.edu.br. Você vê uma linha “If-Modified-Since”?
   2. Inspecione o conteúdo da resposta do servidor, segunda mensagem. O servidor retornou explicitamente o conteúdo do arquivo? Como você pode dizer isso?
   3. Agora inspecione o conteúdo da terceira mensagem - HTTP GET - do seu navegador para o servidor. Você vê uma linha “If-Modified-Since”? Caso a resposta seja afirmativa, qual informação segue o cabeçalho “If-Modified-Since”?
   4. Qual é o código de status e a frase retornada do servidor na resposta à terceira mensagem HTTP GET? É diferente do código de retorno da primeira mensagem? Qual é o código?
   5. Na terceira resposta, o servidor retornou explicitamente o conteúdo do arquivo? Explique.
   6. Qual o tamanho da primeira e terceira mensagem de retorno (respostas) do servidor?

## 4.3 Baixando Documentos Longos

Antes de qualquer experimento deve-se desabilitar algumas funcionalidades do kernel do LINUX, para que os experimentos reflitam a teoria.

Caso sua interface de rede não seja a **eth0** adapte o comando substituindo eth0 pelo nome da sua interface de rede:

```
sudo ethtool --offload eth0 gso off tso off sg off gro off
```

Despreze a mensagem de erro

1. Nos exemplos até agora, os documentos baixados foram simples e pequenos arquivos em HTML. Vamos ver o que acontece quando baixamos um arquivo em HTML grande. Faça o seguinte:
   1. Inicie o navegador web;
   2. Limpe o cache do seu navegador (**Ctrl + Shift + Del**);
   3. Inicie o Wireshark;
   4. Digite o URL no navegador http://redes.sj.ifsc.edu.br/Redes_arq2.html. Seu navegador deve exibir um documento bastante longo e criativo :);

5. Faça um atualização da página (F5);
6. Pare a captura de pacotes, e digite "http" na caixa de texto de especificação de filtro, para que apenas as mensagens HTTP seja exibidas.

2. Na janela de listagem de pacotes, clique sobre a resposta do servidor (**200 OK (text/html)**)
3. Na janela de detalhes do pacote, clique sobre o nono ".... **Reassembled TCP Segments**"

- Esta resposta, em vários pacotes, merece uma explicação. Lembre-se da seção 2.2 do livro (veja a figura 2.9) que a mensagem de resposta HTTP consiste de uma série de linhas de cabeçalho, seguida por uma linha em branco, seguida pela carga útil (*Content-Length*). Nessa resposta, a carga útil do arquivo em HTML é bastante longo, e a informação de **12049 bytes** é muito grande para caber em um único segmento TCP. Assim sendo, a resposta HTTP é quebrada em vários pedaços pelo TCP, com cada pedaço sendo contido dentro de um segmento TCP separado. Cada segmento TCP é capturado em um pacote separado pelo Wireshark. Aqui fica evidente a relação entre camadas: Na camada de aplicação uma grande mensagem que é quebrada pela camada de transporte para "dar conta" de fazer o serviço de entrega.

4. Responda às seguintes questões:

1. Quantas mensagens HTTP GET foram enviadas pelo seu navegador?
2. Quantas respostas HTTP sua máquina recebeu?
3. Quantos segmentos TCP foram necessários para carregar a resposta?
4. Qual é o código de status e a frase associada com a resposta à mensagem HTTP GET? Obs.: Observe os campos do cabeçalho de uma resposta HTTP.
5. Quantos segmentos TCP foram necessários para obtenção da segun resposta do servidor?
6. O que explica a diferença entre a primeira e segunda requisições?

## 4.4 Documentos HTML com Objetos Incluídos

1. Agora que vimos como o Wireshark mostra o tráfego capturado para arquivos em HTML grandes, nós podemos observar o que acontece quando o seu navegador baixa um arquivo principal com objetos incluídos, no nosso exemplo, imagens que estão armazenadas em outros servidores. Faça o seguinte:

1. Inicie o navegador web;
2. Limpe o cache do seu navegador (**Ctrl + Shift + Del**);
3. Inicie o Wireshark;
4. Digite o URL no navegador http://redes.sj.ifsc.edu.br/Redes_arq3.html. Seu navegador deve exibir um arquivo pequeno em HTML com duas imagens incluídas. Estas duas imagens estão referenciadas no arquivo em HTML. Isto é, as imagens não são conteúdos do arquivo em HTML e nem estão depositadas no mesmo servidor, ao invés disso, há um URL para cada imagem no arquivo em HTML. Como discutido no livro, seu navegador terá que baixar estas imagens dos locais correspondentes. As imagens estão em docente.ifsc.edu.br;

2. Digite o URL no navegador http://redes.sj.ifsc.edu.br/Redes_arq4.html. Seu navegador deve exibir um arquivo pequeno em HTML com cinco imagens incluídas. Estas cinco imagens,diferentemente do caso anterior, estão depositadas no próprio sítio navegado;
3. Pare a captura de pacotes, e digite “http” na caixa de texto de especificação de filtro, para que apenas as mensagens HTTP seja exibidas.
4. Responda às seguintes questões, separando as respostas para o acesso ao Redes_arq3.html e Redes_arq4.html (6 respostas):

1. Quantas mensagens HTTP GET foram enviadas pelo seu navegador em cada acesso?
2. Para quais endereços na Internet (URI = Host + URL) estas mensagens foram enviadas em cada acesso?
3. Você consegue dizer se o seu navegador baixou imagens com ou sem paralelismo? Explique e diferencie o comportamento do navegador com e sem paralelismo.

## 4.5 HTTPS

- O Hyper Text Transfer Protocol Secure (HTTPS) é uma implementação do protocolo HTTP sobre uma camada adicional de segurança que utiliza o protocolo SSL/TLS e permite a transmissão de dados numa conexão criptografada através de certificados digitais.
- Caso tenha interesse em analisar troca de mensagens HTTPS e verificar seus conteúdos siga o roteiro *How to Decrypt SSL and TLS Traffic Using Wireshark* (https://www.comparitech.com/net-admin/decrypt-ssl-with-wireshark/#:~:text=Configure%20Wireshark%20to%20decrypt%20SSL&text=Open%20Wireshark%20and%20click%20Edit,)%2DMaster%2DSecret%20log%20filename.)

# 5 Serviço de Nomes (DNS)

## 5.1 Objetivos

O Domain Name System (DNS) traduz nomes de hosts em endereços Internet Protocol (IP), preenchendo uma lacuna crítica na infraestrutura da Internet. Neste laboratório, observaremos mais de perto:

1. o lado cliente do DNS e
2. uma pequena análise do protocolo

Lembre-se de que o papel do cliente no DNS é relativamente simples - um cliente envia uma consulta ao seu DNS, e obtém uma resposta. Muito pode acontecer “por baixo dos panos”, de forma invisível aos clientes DNS, enquanto os servidores DNS, organizados hierarquicamente, comunicam-se entre si para, ou recursivamente ou iterativamente, resolver uma consulta DNS de um cliente. Do ponto de vista do cliente DNS, contudo, o protocolo é bastante simples - uma consulta é feita ao seu servidor DNS e uma resposta é recebida deste servidor.

### 5.1.1 Leitura recomendada

- Detalhes sobre DNS

## 5.2 Consulta simples ao DNS gerada a partir de um comando ping

O comando ping pode ser usado tanto com um endereço IP como com um nome de host.

- Em última instância, ele sempre enviará pacotes para um endereço IP.
- No caso de ser usado o endereço de host, ele tentará resolver (mapear) este nome em um endereço IP usando um servidor DNS (local).
- Ele gera uma pergunta para o servidor (ou para os servidores, caso exista mais de um configurado).
- Esta experiência mostra como verificar os servidores instalados e, através de uma captura de pacote mostra a estrutura dos cabeçalhos DNS.

1. Inicialmente consulte e anote quem são os servidores DNS instados na sua máquina. É para estes servidores que serão conduzidas as perguntas DNS. Digite no terminal:

```
cat /etc/resolv.conf
```

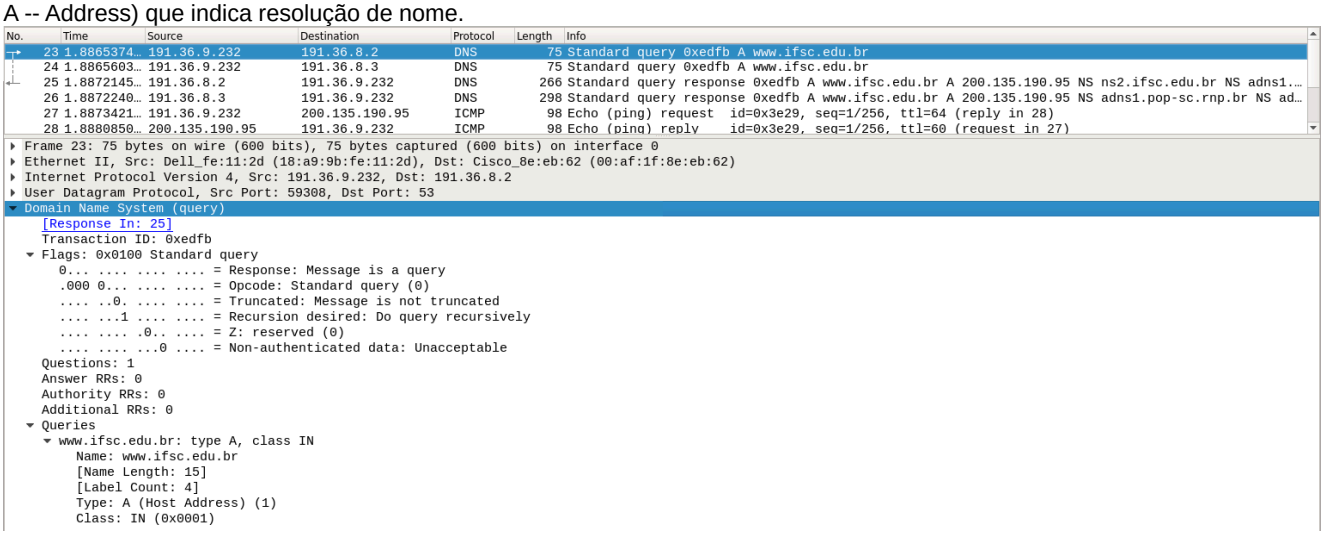
2. Inicie o wireshark para capturar os pacotes.
3. Execute o ping para um endereço de host conhecido, no terminal:

```
ping4 -c2 www.ufsc.br
```

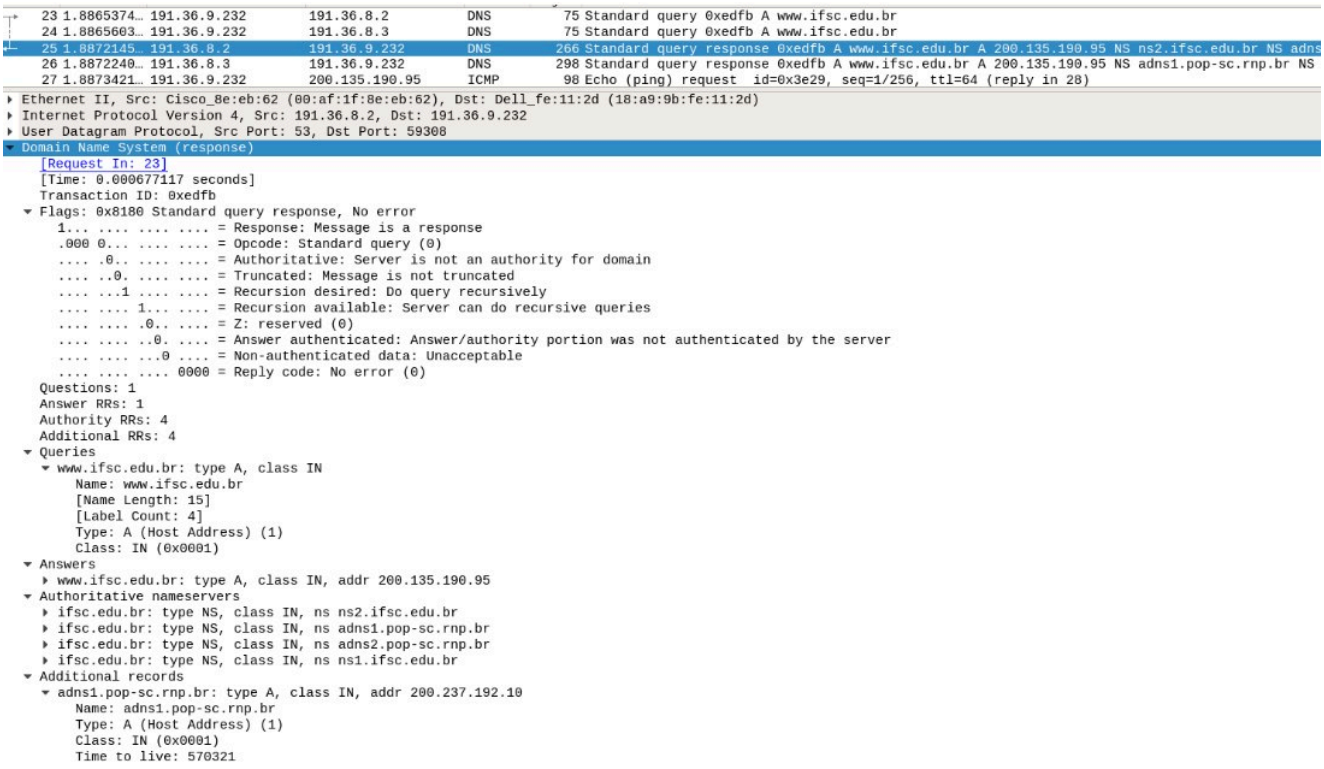
4. Pare a captura de pacotes no Wireshark e coloque um filtro de display para mostrar apenas mensagens DNS ou ICMP

```
dns or icmp
```

5. Observe os pacotes capturados.
   - Em particular foque no pacote de pergunta que deve ser similar ao mostrado abaixo e deve estar direcionado a um dos servidores DNS. Nos *Flags* do *header* do pacote DNS é possível observar que é um QUERY (pergunta) a ser resolvido de forma recursiva.
   - A pergunta propriamente dita está no campo QUERIES, onde é colocado o nome a ser resolvido e o tipo do registro solicitado (tipo A -- Address) que indica resolução de nome.




   - Foque agora um pacote de resposta do servidor para o cliente.
     - Deve ter uma estrutura similar ao mostrado abaixo.
     - Nos **Flags** do *header* do pacote DNS é possível observar que é uma resposta.
     - A resposta propriamente dita está no campo ANSWERS (ele também repete a pergunta no campo QUERIES).
     - Note que podem haver vários registros (RR) retornados, cada um com um tipo.
     - No exemplo abaixo também é retornada uma lista de servidores autorizados (RR tipo NS).
     - Também é retornado o endereço IP destes servidores através de RRs adicionais do tipo A (inclusive endereços IPv6).




- Perguntas a serem respondidas, baseado nos pacotes "*Standard query*", "*Standard query response*" e leitura do arquivo **resolv.conf**:
  1. Quem são os servidores DNS da sua máquina?
  2. O ping gerou pergunta para cada um deles, ou somente para um?
  3. Qual o tipo da RR associada a pergunta (*Queries*). O que significa?
  4. Qual endereço IP retornado para o www.ufsc.br?
  5. Qual endereço IP, de destino, usado no ping (ver pacote REQUEST ICMP)?
  6. Qual é o número da porta do servidor DNS?
  7. Qual protocolo de transporte, camada 4, que foi usado para transportar as mensagens de aplicação DNS?
- Logo após o primeiro ping existe mais uma consulta DNS. Esta pergunta é realizada através de uma mensagem do tipo PTR. O ping está tentando verificar qual é o nome da máquina que realmente está respondendo. É o DNS reverso, nesse tipo de consulta se fornece um IP e o servidor devolve o nome da máquina.
- Perguntas a serem respondidas:
  1. Qual o IP que se pretende resolver?
  2. Qual o nome retornado?
  3. O nome retornado é www.ufsc.br? Sim ou não? Explique.

## 5.3 Consultas DNS por meio de ferramentas especializadas

1. Usando o programa host (http://manpages.ubuntu.com/manpages/precise/man5/hosts.5.html) ou dig (http://manpages.ubuntu.com/manpages/precise/man1/dig.1.html), que são executados no terminal, descubra e anote no relatório os endereços IP associados aos seguintes nomes de hosts (máquinas):
   - mail.ifsc.edu.br
   - www.google.com
   - www.gmail.com
2. Agora descubra e anote no relatório quais são os servidores DNS responsáveis por cada um dos **domínios** dos nomes acima. Cuidado, nome de máquina e domínio são distintos. Máquina (host) pertence ao domínio. P.e.: a) máquina mail.ifsc.edu.br; b) domínio ifsc.edu.br

```
host -t ns ifsc.edu.br
dig -t ns ifsc.edu.br
```

3. O serviço DNS fornece outras informações além do endereço IP associado a cada nome de domínio e/ou máquina. Por exemplo, como ele pode-se descobrir que *host* recebe emails em um determinado domínio. Isso é utilizado pelos MTA (*Mail Transfer Agent*) mundo afora para entregarem emails para seus destinatários (lembre que isso se faz com o protocolo SMTP). Para descobrir essa informação, deve-se consultar o registro MX (*Mail eXchange*) de um domínio. Novamente as ferramentas a ser utilizada nesse caso podem ser *host* ou *dig*. Por exemplo:

```
host -t mx ifsc.edu.br
dig -t mx ifsc.edu.br
```

4. Descubra e anote no relatório quem é o servidor de emails nos seguintes domínios:
   - gmail.com
   - hotmail.com
   - ifsc.edu.br
5. Outra informação útil guardada por servidores DNS é a tradução de endereço IP para nome de domínio. Isso é chamado de tradução reversa (ou DNS reverso). Usando os programas de diagnóstico já vistos, isso pode ser feito assim:

```
dig -x 200.135.37.65
```

   ... o *dig* tem um resultado um pouco mais carregado que os outros utilitários (*host* e *nslookup*), porém neste caso é mais prático. Veja o resultado da consulta logo após a linha *;; ANSWER SECTION:*. Experimente fazer a resolução reversa para cada um dos IP obtidos nas consultas realizadas no primeiro exercício desta atividade. Pode-se também usar a variante do *dig* para respostas curtas:

```
dig -x 200.135.37.65 +short
```

6. Exemplo de consulta iterativa:

```
dig +trace www.polito.it
```

Comando

```
; <<>> DiG 9.20.0-2ubuntu3-Ubuntu <<>> +trace www.polito.it
;; global options: +cmd
.                       115885  IN      NS      m.root-servers.net.
.                       115885  IN      NS      d.root-servers.net.
.                       115885  IN      NS      e.root-servers.net.
.                       115885  IN      NS      i.root-servers.net.
.                       115885  IN      NS      j.root-servers.net.
.                       115885  IN      NS      c.root-servers.net.
.                       115885  IN      NS      l.root-servers.net.
.                       115885  IN      NS      a.root-servers.net.
.                       115885  IN      NS      f.root-servers.net.
.                       115885  IN      NS      g.root-servers.net.
.                       115885  IN      NS      k.root-servers.net.
.                       115885  IN      NS      b.root-servers.net.
.                       115885  IN      NS      h.root-servers.net.
.                       475110  IN      RRSIG   ..... (omitido propositalmente)
;; Received 553 bytes from 191.36.8.66#53(191.36.8.66) in 9 ms
```

Listagem dos RLDs, recebido do DNS local, porta 53 (última linha do bloco)

```
it.                     172800  IN      NS      a.dns.it.
it.                     172800  IN      NS      d.dns.it.
it.                     172800  IN      NS      m.dns.it.
it.                     172800  IN      NS      r.dns.it.
it.                     172800  IN      NS      dns.nic.it.
it.                     172800  IN      NS      nameserver.cnr.it.
it.                     86400   IN      DS      ..... (omitido propositalmente)
;; Received 760 bytes from 193.0.14.129#53(k.root-servers.net) in 152 ms
```

Listagem dos TLDs, recebido do k.root-servers.net

```
polito.it.              3600    IN      NS      giove.polito.it.
polito.it.              3600    IN      NS      leonardo.polito.it.
polito.it.              3600    IN      NS      ns1.garr.net.
polito.it.              3600    IN      NS      ns3.polito.it.
RS1N3..... (omitido propositalmente)
;; Received 946 bytes from 217.29.76.4#53(m.dns.it) in 290 ms
```

Listagem dos SLDs, recebido do m.dns.it

```
www.polito.it.          3600    IN      CNAME   webvip-01.polito.it.
webvip-01.polito.it.    86400   IN      A       130.192.182.100
polito.it.              86400   IN      NS      giove.polito.it.
polito.it.              86400   IN      NS      leonardo.polito.it.
polito.it.              86400   IN      NS      ns1.garr.net.
polito.it.              86400   IN      NS      ns3.polito.it.
;; Received 251 bytes from 130.192.3.24#53(giove.polito.it) in 510 ms
```

Resposta DNS informando que www.polito.it é um apelido (CNAME) para webvip-01.polito.it, que possui o IP 130.192.182.100, recebido do giove.polito.it

7. Faça uma consulta iterativa com *dig* e responda:

```
dig +trace mail.ru.
```

   1. Qual foi o RLD (*Root Level Domain*) consultado?
   2. Qual o TLD (*Top Level Domain*) consultado?
   3. Qual o SLD (*Second Level Domain*) consultado?
   4. Como você sabe que foram esses os LDs consultados?

## 5.4 Algumas consultas AAAA

Vamos expandir um pouco nossos horizontes e fazer algumas consultas envolvendo IPv6.

1. No terminal de sua máquina faça uma consulta e responda: qual o endereço IPv6 dos hosts? Por exemplo:

```
dig AAAA google.com
host -t AAAA google.com
```

   1. www.ufsc.br
   2. ipv6.br
2. Agora vamos fazer a consulta reversa. Qual é o nome de host dos seguintes endereços? Por exemplo:

```
dig -x 2600:1419:1e00:38e::356e
dig -x 2600:1419:1e00:38e::356e +short
host 2600:1419:1e00:38e::356e
```

   1. 2801:84:0:2::10
   2. 2001:12d0:0:126::183:244
3. AVANÇADO. Execute somente se tiver curiosidade. Como explicado durante a aula e visto no exercício anterior, DNS é um banco de dados distribuído. Isso quer dizer que suas informações estão espalhadas em milhares (ou milhões?) de servidores DNS mundo afora. Cada servidor DNS mantém os dados dos domínios por que é responsável. Será que é possível rastrear todos os servidores DNS que devem ser consultados até chegar ao servidor do domínio procurado? Posto de outro modo, vamos fazer todo o processo de requisição interativa, do exercício anterior, manualmente, ou seja, descobrir quem é o *Root Level Domain*, o *Top Level Domain* e o *Second Level Domain*. Anote no relatório.

   1. Descubra quem são os servidores raiz (topo de hierarquia DNS):

```
host -t ns .
dig -t ns .
```

   2. Escolha um dos servidores TLD listados, e use-o para fazer as consultas. Por exemplo:

```
host -v -t a www.sj.ifsc.edu.br. j.root-servers.net.
```

      ... e observe a seção *;; AUTHORITY SECTION:*. Ele contém a listagem de servidores DNS que podem atender sua consulta.
   3. Continue fazendo as consultas aos servidores DNS listados, até conseguir traduzir o nome requisitado. Por exemplo:

```
host -v -t a www.sj.ifsc.edu.br. b.dns.br
```

   4. Quantos servidores DNS foram necessários consultar no total?

## 5.5 Exemplos de arquivos de um *Second Level Domain* fictício

- Exemplo de arquivos de configuração de um servidor BIND (https://www.isc.org/downloads/bind/)

/etc/bind/db.redes

```
$TTL    86400
@   IN  SOA ns.redes.edu.br. root (
             2016090900     ; Serial
              604800     ; Refresh
               86400     ; Retry
             2419200     ; Expire
               86400 )   ; Negative Cache TTL
;
@   IN  NS  ns.redes.edu.br.
@   IN  MX  10  mail.redes.edu.br.
$ORIGIN redes.edu.br.
ns  IN  A   192.168.1.101
www IN  A   192.168.1.102
www IN  A   192.168.1.103
www IN  A   192.168.1.104
www IN  A   192.168.1.105
www IN  A   192.168.1.106
www IN  A   192.168.1.107
```

```
mail    IN  A   192.168.1.109
ftp IN  CNAME   mail.redes.edu.br.
```

/etc/bind/db.1.168.192 (Zona reversa)

```
$TTL    86400
@   IN  SOA ns.redes.edu.br. root (
              2016090900     ; Serial
              604800     ; Refresh
               86400     ; Retry
             2419200     ; Expire
               86400 )   ; Negative Cache TTL
;
    IN      NS      ns.redes.edu.br.
101       IN      PTR     ns.redes.edu.br.
102       IN      PTR     www.redes.edu.br.
108       IN      PTR     ftp.redes.edu.br.
109       IN      PTR     mail.redes.edu.br.
```

# 6 Camada de Aplicação: Colocando no "ar" aplicações servidoras

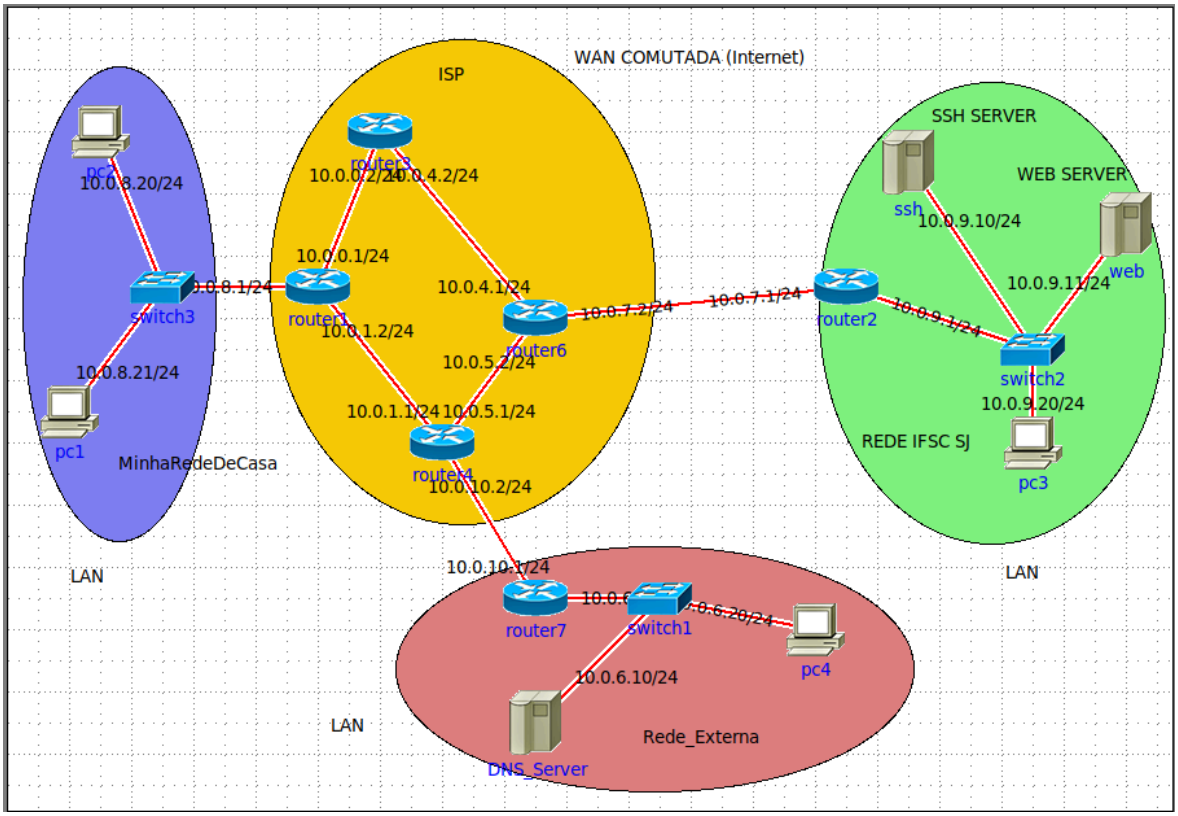
## 6.1 Objetivos

- Testar serviços de rede na camada de aplicação.
- Construir uma pequena internet, colocando dois serviços no "ar": um **servidor ssh**, um **servidor DNS** e um **web server**, .
- Uma visão do posicionamento dos "pacotes de aplicação" capturados para cada um destes serviços.

Os serviços são, portanto:

- serviço SSH: *Secure Shell*, terminal remoto. Permite acessar um computador remoto através de um terminal seguro.
- serviço DNS: Vai permitir a navegação através de nomes de máquinas. Com este serviço vamos configurar o banco de dados de um servidor DNS, BIND (https://www.isc.org/bind/). Ele dará suporte a navegação por nomes de máquinas.
- serviço WEB: permite hospedar e acessar remotamente páginas da Internet.

## 6.2 Topologia de rede para experimetação




## 6.3 Servidor SSH

1. Vamos usar o simulador **Imunes** para nos apropriarmos do sentimento do comportamento em camadas em uma rede simples. Construir no Imunes a rede a seguir ou, se preferir, importe o arquivo digitando no terminal:

```
wget http://redes.sj.ifsc.edu.br/Camada_aplicacao.imn
```

2. Inicie o **Imunes** e carregue o arquivo salvo (Camada_aplicacao.imn).
3. Iniciando a REDE:

```
Experiment >> Execute
```

4. Executando serviço SSH. O serviço SSH será iniciado no **servidor SSH** (SSH SERVER).
    1. Primeiramente vamos atribuir uma senha ao usuário root no servidor SSH, por exemplo, atribua senha ***root*** (SSH SERVER) com o seguinte comando no terminal do SSH SERVER (duplo clique sobre o ícone da máquina SSH SERVER abre o terminal de comandos da mesma):

```
passwd
```

    - Durante a digitação da senha, nada é apresentado no terminal, é normal.
    - Ao terminar de digitar a senha tecle <enter>. Será solicitado a confirmação da senha com o mesmo procedimento.</enter>

    2. Em seguida vamos fazer uma pequena configuração no **servidor SSH** (SSH SERVER), através do comando:

```
echo PermitRootLogin yes >>/etc/ssh/sshd_config
```

    3. Iniciamos o serviço, através do comando:

```
/etc/init.d/ssh start
/etc/init.d/ssh reload
```

    4. Confira se o serviço está rodando, através do comando:

```
ps aux
```

    - Observe se há um processo do tipo, última coluna a direita: /usr/sbin/sshd

    5. Agora vamos testar a conectividade do serviço fazendo uma acesso remoto, por exemplo, no terminal do **pc2** execute:

```
ssh 10.0.9.10
```

- Na primeira pergunta responda com yes
- Na segunda pergunta preencha com a senha: root

6. Observe e salve que o prompt do seu terminal mudou para **root@ssh:~#**, isso significa que, apesar de você estar no terminal do pc2, vocês está conectado no SSH SERVER. Tudo que você digitar estará sendo executado no SSH SERVER.
7. No terminal do **pc2**, que na verdade está conectado ao servidor **SSH SERVER**, vamos deixar um ping testando a conectividade com o próprio **pc2**:

```
ping 10.0.8.20
```

5. Agora vamos capturar pacotes do ssh. Basta usar o Wireshark em qualquer interface onde passam os pacotes. Por exemplo, no **router2**.

```
Clique com o direito do mouse sobre o router2 >> Wireshark >> eth0
```

6. Pare o ping.
7. Recorte a tela do Wireshark, filtrando os pacotes do **ssh**. Mostre o encapsulamento de pacotes de aplicação e seu posicionamento na estrutura de pacotes.
8. Recorte a tela do Wireshark, filtrando os pacotes do **icmp**. Comprovando que os pacotes do ping estão passando pelo **router2**.
9. Para encerrar a conexão ao **SSH SERVER**, no terminal do **pc2** digite:

```
exit
```

- Observe e salve que o prompt do seu terminal mudou para **root@pc2:~#**, isso significa que a conexão foi encerrada.

## 6.4 Servidor DNS

### 6.4.1 Criando um domínio de Internet

1. Abra o terminal do **DNS_Server**, vá até o diretório de configuração do BIND com o comando:

```
cd /etc/bind
```

2. Defina uma zona, um *Second Level Domain*, de nome **redes.edu.br**. Isto é feito editando o arquivo **named.conf.default-zones**. No terminal do **DNS_Server** e digite:

```
nano named.conf.default-zones
```

3. Acrescente **ao final** do mesmo o seguinte conteúdo:

```
zone "redes.edu.br" {
        type master;
        file "/etc/bind/db.redes";
};
```

- Ao terminar de editar digite **<CRTL> + X**, em seguida, **Y** e **<Enter>**.
- Este arquivo definirá uma nova zona (domínio) DNS.

### 6.4.2 Criando a base de dados relativa ao domínio

1. Na zona criada atribua endereços IPv4 (A) as máquinas em **db.redes:**

```
nano db.redes
```

2. Cole o seguinte conteúdo no arquivo e salve:**:**

```
$TTL    86400
@       IN      SOA     ns.redes.edu.br. root (
                2022051200      ; Serial
                604800          ; Refresh
                86400           ; Retry
                2419200         ; Expire
                86400 )         ; Negative Cache TTL
;
@               IN      NS      ns.redes.edu.br.
@               IN      MX      10      mail.redes.edu.br.
$ORIGIN redes.edu.br.
ns              IN  A   10.0.6.10
```

```
www             IN  A   10.0.9.11
ssh             IN  A   10.0.9.10
mail            IN  A   10.0.6.10
apelido         IN  CNAME       mail.redes.edu.br.
```

- Ao terminar de editar digite **<CRTL> + X**, em seguida, **Y** e **<Enter>**.
- Este arquivo definirá o banco de dados do DNS.

3. Reinicie o serviço DNS, através do comando:

```
/etc/init.d/bind9 restart
```

4. Faça um teste com consulta ao seu servidor com o comando, por exemplo:

```
dig @localhost www.redes.edu.br
```

- O resultado obtido deve conter algo do tipo:

```
;; ANSWER SECTION:
www.redes.edu.br.       86400   IN      A       10.0.9.11
```

### 6.4.3 Configurando as máquinas para acessarem o DNS

1. Em qualquer máquina que desejar navegar por nomes, declare o servidor **DNS_Server** como **servidor DNS** com o seguinte comendo digitado no respectivo terminal:

```
echo nameserver 10.0.6.10 >> /etc/resolv.conf
```

2. Faça alguns testes simples via ping, por exemplo:

```
ping www.redes.edu.br
ping apelido.redes.edu.br
```

3. Acrescente um novo endereço ao banco de dados: **casa.redes.edu.br** apontando para o IP **10.0.8.21**.
4. Prove que o mesmo está funcional, por exemplo, dando um ping casa.redes.edu.br a partir do host ssh.
5. No terminal do **DNS_Server** execute:

```
dig apelido.redes.edu.br
```

- Qual foi o resultado obtido?
- Qual o significado?

## 6.5 Servidor WEB

1. Preparando uma página HTML para colocar no servidor WEB.
   - Páginas da internet são construídas usando o formato HTML.
   - Ver aqui o que é uma página HTML (https://wiki.sj.ifsc.edu.br/images/e/ee/03_-_HTML.pdf) e como construir uma página simples.
   1. No terminal da máquina **WEB SERVER** entre no diretório diretório **/var/www/html**:

```
cd /var/www/html
```

   2. Use o editor nano para editar uma página chamada **index.html**:

```
nano index.html
```

      - Salve um print com o editor aberto.

   3. Crie a página com o seguinte conteúdo:

```
<html>
<body>
<h1>Redes de Computadores</h1>
<p>Pagina teste do aluno Pedro Alvares Cabral da Silva</p>
</body>
</html>
```

      - Ao terminar de editar digite **<ctrl> + <x></x></ctrl>** e, em seguida, **Y** e **<enter></enter>**.

   4. Inicie o SERVIÇO WEB para testar o protocolo HTTP. O servidor WEB irá disponibilizar a página criada para acesso remoto via protocolo HTTP:

```
/etc/init.d/lighttpd start
```

2. No **router2** deixe o Wireshark capturando pacotes.
3. Faça um acesso a sua página, a partir do Firefox (cliente HTTP) em um PC cliente de sua escolha:

```
Clique com o botão direito do mouse sobre, por exemplo, o pc1
Clique sobre Web Browser
Digite no navegador: www.redes.edu.br
```

   - Se der erro, verifique se a máquina da qual você está navegando está resolvendo nomes, por exemplo, no terminal fazendo um ping www.redes.edu.br.
   - RECORTE a tela com a página em destaque no navegador e cole no relatório.
4. Se desejar modificar a página repita os itens 2 e 3.
5. Crie uma nova página dentro do diretório /var/www/html:

```
nano abacaxi.html
```

6. Coloque um conteúdo a seu critério:

```
<html>
<body>
<h1>Redes de Computadores</h1>
<p>Minha primeira pagina.........</p>
</body>
</html>
```

   - Ao terminar de editar digite **<ctrl> + <x></x></ctrl>** e, em seguida, **Y** e **<enter></enter>**.
7. Acesse a nova página através do navegador Firefox com o endereço (URL):

```
http://www.redes.edu.br/abacaxi.html
```

   - RECORTE a tela com a página em destaque no navegador e cole no relatório.
8. Salve um print da tela do Wireshark destacando a troca de mensagens HTTP e o conteúdo HTML de uma das páginas acessadas.
9. Extra (não obrigatório). Para quem quiser enfeitar sua página html com, por exemplo, imagem, siga o roteiro:
   1. Baixe a imagem desejada na máquina real, por exemplo, imagem.jpg.
   2. Abra um terminal da máquina real e, no diretório onde se encontra imagem.jpg, digite:

```
sudo hcp imagem.jpg web:/var/www/html
```

   3. Confira no terminal do WEB SERVER (Imunes) se a imagem está lá:

```
cd /var/www/html
ls -l
```

   4. Use a imagem em sua página, seguindo o exemplo em Imagens no HTML (https://developer.mozilla.org/pt-BR/docs/Learn/HTML/Multimedia_and_embedding/Images_in_HTML).

# 7 Comparando *sockets* UDP e TCP

## 7.1 Objetivos

- Entender o conceito de *sockets* relacionados aos protocolos UDP e TCP.
  - Processos que rodam em máquinas diferentes se comunicam entre si enviando mensagens para *sockets*. Um processo é semelhante a um prédio e o *socket* do processo é semelhante a uma porta em seu interior. A aplicação reside dentro do prédio e o protocolo da camada de transporte reside no mundo externo. Um programador de aplicação controla o interior do prédio mas tem pouco (ou nenhum) controle sobre o exterior.
- Simultaneamente explora-se os conceitos relativos aos protocolos UDP e TCP, observando-se a quantidade de mensagens necessárias para a troca de uma simples frase textual.
  - **Observa-se a "agilidade" do UDP e a robustez do TCP**.
- Por fim, propõe-se um comparativo entre os dois protocolos da camada de transporte: UDP e TCP.

Leia os slides de 1 à 12 e o 58: Capitulo 3 -- Camada de Transporte (http://docente.ifsc.edu.br/odilson/RED29004/PPTs%20-%20Cap%C3%ADtulo%203%20Camada%20de%20transporte.pdf)

## 7.2 Descrição da aplicação a ser desenvolvida em UDP e TCP

- Usaremos a aplicação cliente-servidor simples a seguir para demonstrar a programação de *socket*:

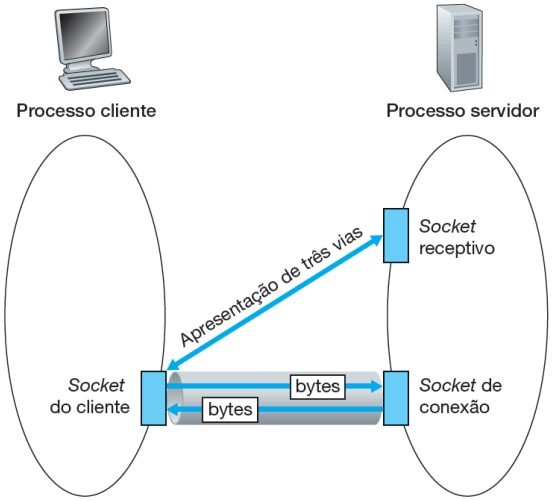
1. Um cliente lê uma linha de caracteres (dados) do teclado e a envia para o servidor.
2. O servidor recebe os dados e converte os caracteres para maiúsculas.
3. O servidor envia os dados modificados ao cliente.
4. O cliente recebe os dados modificados e apresenta a linha em sua tela.

## 7.3 Programação de *sockets* com TCP

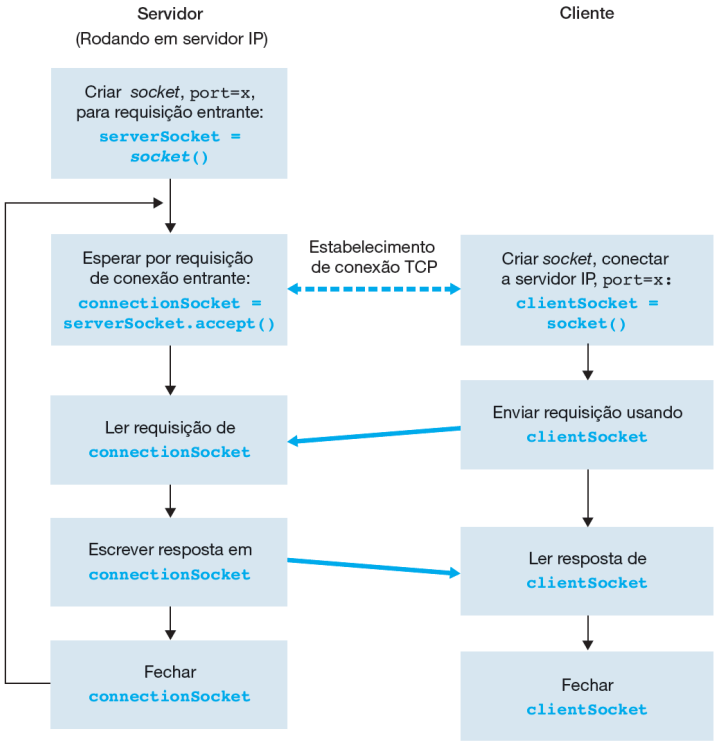
- Diferentemente do UDP, o TCP é um protocolo orientado a conexão. Pode-se dizer que o TCP é realizado em duas etapas:

1. Primeiramente eles devem se apresentar, o primeiro *socket* da Figura abaixo. Isto serve somente para abertura de conexão.
2. Estabelecer uma conexão TCP, o segundo *socket* da Figura abaixo. Todos os dados trafegarão pelo segundo *socket*.

O processo TCPServer tem dois sockets:




A aplicação cliente-servidor usando TCP:




### 7.3.1 Roteiro

- O roteiro será executado sobre máquinas virtuais, através do uso do Imunes (http://imunes.net/).
- Utilizamos a linguagem Python por expor com clareza os principais conceitos de *sockets*. Quem desejar pode implementar em outras linguagens, por exemplo um modelo para programação de *sockets* utilizando a API Posix encontra-se aqui.

1. Abra um terminal e baixe o aquivo de configuração da rede a ser utilizada nos experimentos:

```
wget -4 http://redes.sj.ifsc.edu.br/Sockets.imn
```

2. Execute o Imunes.
3. Carregue o arquivo:

```
File >> Open >> Sockets.imn
```

   - Será apresentada uma simples rede a ser utilizada no experimento, composta de 2 clientes e um servidor.

4. Inicie a simulação da rede no Imunes:

```
Experiment >> Execute
```

5. Copie o código do programa servidor, abaixo, e salve como TCPServer.py. No terminal do **Servidor** digite:

```
nano TCPServer.py
```

   - Dica: para abrir um terminal de uma das máquinas da rede a ser simulada basta dar um duplo clique sobre o ícone da mesma.
   - Dica: para copiar textos para os terminais do Imunes, copie normalmente o texto, por exemplo, da Wiki, com o < Ctrl > + < C > e cole com < Ctrl > + < Shift > + < V > ou clicando sobre a rodinha (*scroll*) do mouse sobre o terminal desejado do Imunes.

```
from socket import *
serverPort = 33333
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('',serverPort))
#Escuta as requisicoes do TCP do cliente. Numero maximo de conexoes em fila = 1
serverSocket.listen(1)
print ('O servidor esta pronto')
while 1:
    #Quando o cliente bate a essa porta, o programa chama o metodo accept() para serverSocket,
        #que cria um novo socket no servidor, chamado connectionSocket, dedicado a esse cliente
        #especifico. Cliente e servidor, entao, completam a apresentacaoo, criando uma conexao TCP
        #entre o clientSocket do cliente e o connectionSocket do servidor.
        connectionSocket, addr = serverSocket.accept()
        message = connectionSocket.recv(1024)
        print (message)
        messageMaiuscula = message.upper()
```

```
        connectionSocket.send(messageMaiuscula)
        connectionSocket.close()
```

- Ao terminar de editar digite **<CRTL> + X**, em seguida, **Y** e **<Enter>**.

6. No terminal do **Servidor** e execute a aplicação servidora:

```
python3 TCPServer.py
```

- Caso dê uma mensagem de erro, tente entende-la e corrija o problema. Com certeza é sintaxe. Deixe o programa rodando nesse terminal.

7. Copie o código do programa **Cliente1** e **Cliente2** e salve como TCPClient.py. No terminal dos **Clientes 1 e 2** digite:

```
nano TCPClient.py
```

8. Cole o seguinte conteúdo:

```
from socket import *
serverName = '10.0.0.10'
serverPort = 33333
#SOCK_STREAM habilita uso do TCP
clientSocket = socket(AF_INET, SOCK_STREAM)
#Representa o estabelecimento da conexao. É o "aperto de maos", onde o cliente e servidor trocam
#informacoes da portas que serao utilizadas pela conexao (socket) propriamente dito
clientSocket.connect((serverName,serverPort))
message = input('Entre com a sentenca em minúsculas: ')
#Diferentemente do UDP, aqui não é necessário encaminhar o endereço do servidor, já que este socket
#é uma "tubulação" direta entre ambos, basta empurrar dados
clientSocket.send(bytes(message, 'utf-8'))
modifiedMessage = clientSocket.recv(1024)
print('Mensagem do servidor: ', modifiedMessage.decode('utf-8'))
clientSocket.close()
```

- Ao terminar de editar digite **<CRTL> + X**, em seguida, **Y** e **<Enter>**.

9. Execute o WireShark na interface **eth0** do **Servidor**:

```
Clique com o botão direito do mouse sobre o ícone do Servidor >> Wireshark >> eth0
```

10. No Wireshark aplique o filtro: **tcp.port==33333**.
11. No terminal do **Cliente 1** execute o programa cliente (por hora não digite nada):

```
python3 TCPClient.py
```

12. Em novos terminais do **Cliente 1 e 2** e **Servidor**, verifique os *sockets* TCP ativos:

```
 ss -nta | grep -E '33333|Local'
```

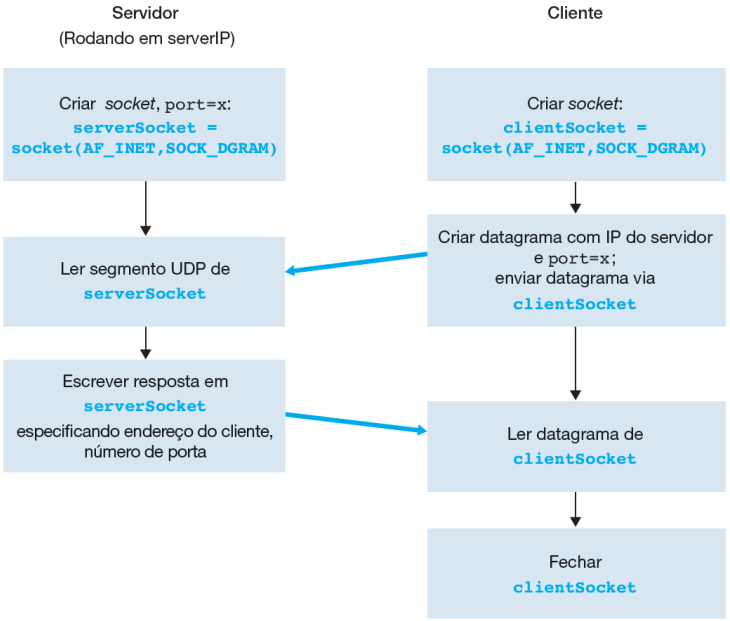
    1. Você encontrou sockets abertos em todos os hosts (máquinas)?
    2. Quais parâmetros apresentados para cada um deles?
    3. Qual a relação entre os *socktes* clientes e servidor, número IP, portas etc?
    4. Identifique e "printe" o socket receptivo no **Servidor**.

13. Em cada uma das aplicações clientes digite um texto, sempre diferente para facilitar a análise no Wireshark. Observe se recebeu o retorno das mensagens em maiúscula.
14. Pare a captura no Wireshark.
15. Perguntas:

    1. Para cada cliente, as três primeiras mensagens trocadas apresentam a camada de aplicação, sim ou não? Explique. O que elas significam (*3-way handshake*)?
    2. Encontre a frase/palavra escrita enviada ao servidor (minuscula) e a resposta em maiúscula?
    3. Qual o tamanho, dos pacotes contendo as mensagens: i) *Data* (camada 5) e ii) *Total Length* (camada 3). Qual a relação entre estes valores?
    4. As última 3 mensagens, de cada cliente, contém o fechamento de conexão, explique-as.
    5. Qual é o protocolo da camada de transporte nessa troca de mensagens?
    6. Qual o número identificador de protocolo TCP no pacote IP? Dica: na janela central abra o campo *Internet Protocol* e procure a string *Protocol*.

## 7.4 Programação de *sockets* com UDP

A aplicação cliente-servidor usando UDP tem a estrutura apresentada na Figura baixo. Utilizamos a linguagem Python por expor com clareza os principais conceitos de *sockets*. Quem desejar pode implementar em outras linguagens, por exemplo um modelo para programação de *sockets* utilizando a API Posix encontra-se aqui.




Como fica evidente na Figura acima, há dois processos cliente e servidor que podem ou não rodar em máquinas distintas e se comunicam justamente enviando mensagens via *sockets*, que abstrai qualquer necessidade de conhecimento das camadas subjacentes.

### 7.4.1 Roteiro

1. Pare a simulação da rede no Imunes, da primeira etapa do TCP:

```
Experiment >> Terminate
```

2. Inicie novamente a simulação da rede no Imunes:

```
Experiment >> Execute
```

3. Copie o código do programa servidor, abaixo, e salve como TCPServer.py. No terminal do **Servidor** digite:

```
nano UDPServer.py
```

4. Escreva (copie) o programa UDPServer.py

```
#Os comentarios estao propositalmente sem acentuacao, caso contrario, tem-se erro de sintaxe.
#Esta linha define que pode-se utilizar sockets dentro do programa
from socket import *
#Define explicitamente a porta aberta servidor
serverPort = 22222
#Cria o socket do servidor, denominado serverSocket. O primeiro parametro indica a familia do endereco,
#em particular, AF_INET indica que a rede subjacente esta usando IPv4. O segundo parametro indica que
#o socket eh do tipo SOCK_DGRAM, ou seja, eh um socket UDP.
serverSocket = socket(AF_INET, SOCK_DGRAM)
#Vincula o numero da porta, nesse caso 22222, ao socket do servidor e "abre a porta".
serverSocket.bind(('', serverPort))
print ('O servidor esta pronto para recepcao')
#Aguarda indefinidamente por contatos (mensagens) de clientes
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    print (message)
    #Ao receber a mensagem do cliente converte todos os caracteres para maiusculas.
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

- Ao terminar de editar digite **<CRTL> + X**, em seguida, **Y** e **<Enter>**.

5. No terminal do **Servidor** e execute a aplicação servidora:

```
python3 UDPServer.py
```

- Caso dê uma mensagem de erro, tente entende-la e corrija o problema. Com certeza é sintaxe. Deixe o programa rodando nesse terminal.

6. Nas máquinas **Cliente 1 e 2** execute no terminal:

```
netcat -u 10.0.0.10 22222
```

7. Verifique os sockets em um novo terminal das máquinas **Cliente 1 e 2** e **Servidor**, com o comando:

```
ss -ua
```

   1. Identifique e anote os sockets abertos.
   2. Identifique e anote o socket receptivo do servidor.

8. Execute o WireShark na interface **eth0** do **Servidor**:

```
Clique com o botão direito do mouse sobre o ícone do Servidor >> Wireshark >> eth0
```

9. No Wireshark aplique o filtro: **udp.port==22222**.
10. Nos terminais das aplicações clientes digite a mensagem que desejar e espere a resposta do servidor. Funcionou?
11. Com o servidor aberto faça duas conexões simultâneas. Pode ser dois terminais rodando a aplicação cliente.
12. Em cada uma das aplicações clientes digite um texto, sempre diferente para facilitar a análise no Wireshark.
13. Pare a captura de pacotes.
14. PERGUNTAS baseadas na captura:

   1. Em algum momento foi identificado algum procedimento para estabelecimento de conexão?
   2. Em algum campo do UDP existe numeração de mensagens?
   3. Qual o número identificador de protocolo UDP no pacote IP? Dica: na janela central abra o campo *Internet Protocol* e procure a string *Protocol*.
   4. Qual é o *checksum* no pacote (datagrama) UDP? Qual é o formato apresentado? Quantos bits ele possui?
   5. É possível capturar toda a troca de mensagens e inclusive capturar o texto passado do cliente para o servidor?
   6. Qual foi a sequência numérica do campo *Data* em seu teste? Qual o significado?
   7. Qual é o protocolo da camada de transporte nessa troca de mensagens?

15. Comparativo entre TCP e UDP:

   1. Quantas mensagens foram trocadas entre o servidor e o cliente em cada um dos protocolos para atingir o mesmo objetivo?
   2. O que justifica a diferença na quantidade de mensagens trocadas?
   3. Discuta as vantagens e desvantagens de cada protocolo.

## 7.5 Desafios extras

1. Modifique uma das aplicações cliente-servidor, seja UDP ou TCP, para fazer um pingue-pongue com a mensagem, ou seja, o cliente gera e envia a mensagem, o servidor a devolve, o cliente reenvia a mesma mensagem, o servidor a devolve e assim sucessivamente.

# 8 TCP x UDP

## 8.1 Objetivos

- O objetivo desses experimentos é evidenciar as diferenças entre os protocolos TCP e UDP.
- Ambos protocolos de transporte podem ser usados por aplicações que precisem se comunicar. Porém cada um deles têm certas propriedades, então a escolha precisa ser realizada baseada nas necessidade de comunicação a ser feita pela aplicação.

## 8.2 Roteiro

**O que aconteceria se um arquivo fosse transferido de um computador a outro com ambos protocolos?**

O roteiro será executado sobre máquinas virtuais, através do uso do Imunes (http://imunes.net/).

1. Abra um terminal e baixe o aquivo de configuração da rede a ser utilizada e um arquivo auxiliar dos experimentos:

```
wget -4 http://redes.sj.ifsc.edu.br/TCPxUDP.imn
wget -4 http://redes.sj.ifsc.edu.br/seq_num.txt
```

2. Observe o tamanho do arquivo auxiliar transferido, seq_num.txt, ele deve ter exatamente 5327160 bytes (cerca de 5,1 MB). Você pode fazer isso com o comando **ls -l**.

### 8.2.1 Transferência utilizando o protocolo TCP

1. Execute o Imunes.
2. Carregue o arquivo:

```
File >> Open >> TCPxUDP.imn
```

- Será apresentada uma simples rede a ser utilizada no experimento, composta de 2 PC com um enlace de 1000 kbps.
- Observe que abaixo do enlace está escrito **loss=5%**. Isso indica a taxa média de perda de pacotes no enlace.

3. Inicie a simulação da rede no Imunes:

```
Experiment >> Execute
```

4. No terminal da máquina real (**NÃO do Imunes**) digite:

```
sudo hcp seq_num.txt Transmissor:
```

- Obs.: Esse comando copia o arquivo seq_num.txt para a máquina virtual do Imunes.
- Dica: para abrir um terminal de uma das máquinas da rede a ser simulada basta dar um duplo clique sobre a mesma.
- Dica: para copiar textos para os terminais do Imunes, copie normalmente o texto, por exemplo, da Wiki, com o < Ctrl > + < C > e cole com < Ctrl > + < Shift > + < V > ou clicando sobre a rodinha (*scroll*) do mouse sobre o terminal desejado do Imunes.

5. Na máquina **Transmissor**, do Imunes, execute o Wireshark, para capturar pacotes vindouros:

```
Clique sobre o ícone da máquina Transmissor com o botão direito do mouse >> Wireshark >> eth0...
```

6. Crie um filtro para monitorar somente o tráfego desejado:

```
ip.addr==10.0.0.21
```

7. No terminal da máquina **Receptor**, do Imunes, execute o **netcat** (nc (http://netcat.sourceforge.net/)) (utilize **man nc** para saber os detalhes das *flags* utilizadas) que abrirá um *socket* **TCP** que ficará aguardando conexão na porta 5555. Os dados recebidos serão salvos (através do direcionamento feito através do símbolo **>**) em **arquivoTCP**:

```
nc -vvnl -p 5555 > arquivoTCP
```

8. No terminal da máquina **Transmissor**, do Imunes, também execute o **netcat** mas com o objetivo de transmitir o arquivo seq_num.txt para a outra maquina (Receptor):

```
time nc -w1 -vvn 10.0.0.21 5555 < seq_num.txt
```

9. No terminal do **Transmissor** o comando será encerrado e apresentará o tempo de transmissão.
10. No terminal do **Receptor**, após o término do processo, verifique o tamanho do arquivo recebido com o comando "ls -l".
    1. O tamanho é igual ao do arquivo seq_num.txt?
    2. Quanto tempo levou para transmiti-lo?
11. Analisando a captura de pacotes do WireShark responda:
    1. Quais as portas origem e destino escolhidas pelo cliente e servidor?
    2. Qual é o número de sequência, para ambas as máquinas, do primeiro e do último pacote?
    3. Qual é o número de sequência, para ambas as máquinas, do primeiro e do último ACK?
    4. Calcule e mostre o procedimento de cálculo do tamanho do arquivo pela análise dos pacotes? Qual é a maneira mais fácil? Apresente os cálculos ou descreva a maneira de obtenção do valor. Dica: observe o primeiro e o último número de sequência e faça uma correlação com o tamanho do arquivo.
    5. Qual é o tamanho do último segmento de dados recebido? Perceba que ele é diferente dos demais, que vem "cheios", que tem tamanho grandes.
    6. Apresente os segmentos do *3-way handshake* e analise os campos do cabeçalho, que os identificam. Estão de acordo com a norma apresentada na literatura (em sala de aula)?
    7. Apresente os segmentos do fechamento de conexão e analise os campos do cabeçalho, que os identificam. Estão de acordo com a norma apresentada na literatura (em sala de aula)?

### 8.2.2 Transferência utilizando o protocolo UDP

**Caso não tenha fechado o Imunes na Parte 1 (Transferência utilizando o protocolo TCP) feche o Wireshark e vá direto para o Item 5.**

1. Execute o Imunes.
2. Carregue o arquivo:

```
File >> Open >> TCPxUDP.imn
```

- Será apresentada uma simples rede a ser utilizada no experimento, composta de 2 PC com um enlace de 1000 kbps.

3. Inicie a simulação da rede no Imunes:

```
Experiment >> Execute
```

- Dica: para abrir um terminal de uma das máquinas da rede a ser simulada basta dar um duplo clique sobre a mesma.
- Dica: para copiar textos para os terminais do Imunes, copie normalmente o texto, por exemplo, da Wiki, com o <Ctrl> + <C> e cole com <Ctrl> + <Shift> + <V> ou clicando sobre a rodinha (*scroll*) do mouse sobre o terminal desejado do Imunes.

4. No terminal da máquina real (**NÃO do Imunes**) digite:

```
sudo hcp seq_num.txt Transmissor:
```

5. Na máquina **Receptor**, do Imunes, execute o Wireshark:

```
Clique sobre o ícone da máquina Receptor com o botão direito do mouse >> Wireshark >> eth0...
```

6. Crie um filtro para monitorar somente o tráfego desejado:

```
ip.addr==10.0.0.20 and udp and !icmp
```

7. Na máquina **Receptor**, do Imunes, execute o **netcat** (nc (http://netcat.sourceforge.net/)) que abrirá um *socket* **UDP** que ficará aguardando segmentos na porta 6666. Os dados recebidos serão salvos em **arquivoUDP**:

```
nc -vvnlu -p 6666 > arquivoUDP
```

8. Na máquina **Transmissor**, do Imunes, também execute o **netcat** mas com o objetivo de transmitir o arquivo seq_num.txt para a outra maquina (Receptor):

```
time nc -w1 -vvnu 10.0.0.21 6666 < seq_num.txt
```

9. No terminal do **Transmissor** o comando será encerrado e apresentará o tempo de transmissão.
10. No terminal do **Receptor**, verifique o tamanho do arquivo recebido com o comando "ls -l".
    1. O tamanho é igual ao do arquivo seq_num.txt?
    2. Quanto tempo levou para transmiti-lo?
11. Analisando a captura de pacotes do WireShark responda:
    1. Qual é o identificador (número de sequência) do primeiro e do último pacote? Existe?
    2. É possível calcular o tamanho do arquivo pela análise dos pacotes? É mais fácil ou difícil que no caso da transferência via TCP?
12. Compare as transferências feitas com os protocolos TCP e UDP em relação, principalmente, ao tempo gasto para transmitir o arquivo e a integridade de dados.
    1. O que eles têm em comum?
    2. Que diferenças lhe pareceram mais pronunciadas?
    3. Como isso deve afetar as aplicações que usam esses protocolos?

# 9 Desvendando o TCP - Número de Sequência, Controle de Erros, Transmissão *Full-Duplex*

## 9.1 Objetivos

- Verificar alguns mecanismos do protocolo TCP:
  - Controle de Erros: Significado de Número de Sequência, ACK;
  - Controle de Fluxo: Significado do campo Windows Size; Funcionamento do controle de fluxo;
  - Transmissão Full-Duplex.

## 9.2 Topologia de rede para experimetação

1. O roteiro será executado sobre máquinas virtuais, através do uso do Imunes (http://imunes.net/).
2. Abra um terminal e baixe os aquivos de configuração da rede a ser utilizada e arquivos auxiliares dos experimentos:

```
cd ~
wget -4 http://redes.sj.ifsc.edu.br/TCP_Num_Seq_Erro.imn
wget -4 http://redes.sj.ifsc.edu.br/arq30Bytes.txt
```

```
wget -4 http://redes.sj.ifsc.edu.br/Servidor.txt
wget -4 http://redes.sj.ifsc.edu.br/Cliente.txt
```

## 9.3 PARTE 1 - Transmissão sem erros: Verificação de Número de Sequência e Reconhecimentos

1. Execute o Imunes.
2. Carregue o arquivo:

```
File >> Open >> TCP_Num_Seq_Erro.imn
```

   - Será apresentada uma simples rede a ser utilizada no experimento, composta de 2 PCs: **Transmissor** e **Receptor**.

3. Inicie a simulação da rede no Imunes:

```
Experiment >> Execute
```

   - Dica: para abrir um terminal de uma das máquinas da rede a ser simulada basta dar um duplo clique sobre a mesma.

4. Copie o arquivo arq30Bytes.txt para a máquina **Transmissor** do Imunes. No terminal da máquina hospedeira (**NÃO do Imunes**) digite:

```
sudo hcp arq30Bytes.txt Transmissor:
```

5. Execute o Wireshark no **Receptor**:

```
Clique direito do mouse sobre o ícone do Receptor >> Wireshark >> eth0..
```

6. Crie um filtro para monitorar somente o tráfego desejado, nos pacotes vindouros:

```
ip.addr==10.0.0.20
```

7. No terminal do **Receptor** , execute o processo servidor e prepare o mesmo para limitar a sua capacidade de recepção em cerca de 20 bytes (tamanho do *buffer*). Isto permitirá ver a quebra do arquivo de 30 bytes em alguns segmentos TCP:

```
sysctl -w net.ipv4.tcp_rmem='20 20 20'
nc -vvnl -p 5555 > ArqRecebido.txt
```

   - Dica: para copiar textos para o Imunes, copie normalmente o texto, por exemplo, da Wiki, com o < Ctrl > + < C > e cole com < Ctrl > + < Shift > + < V > ou clicando sobre a rodinha (*scroll*) do mouse.

8. No terminal, envie o arquivo arq30Bytes.txt da máquina **Transmissor**:

```
nc -vvn 10.0.0.21 5555 < arq30Bytes.txt
```

9. Pare os processos rodando nos terminais do **Transmissor** e **Receptor** com:

```
Ctrl + c
```

10. Pare a captura de pacotes no Wireshark.
11. Na tela do Wireshark você terá algo parecido com o apresentado na Figura 1.
12. Analise como os dados foram transmitidos e reconhecidos.
13. Perguntas
    1. Qual o número de sequência de cada segmento de dados transmitido (do Transmissor para o Receptor) e qual o significado do número de reconhecimento em cada um deles?
    2. Como foi reconhecido cada segmento enviado? É igual ao número de sequência ou é um número acima? Justifique.
    3. Qual o significado, funcionalidade e necessidade das mensagens, inseridas pelo Wireshark, "TCP ZeroWindow" e "TCP Window Update"?
    4. Qual a relação entre os campos "Len=", "Seq=", "Ack=", "Win=" e o tamanho do segmento de dados?

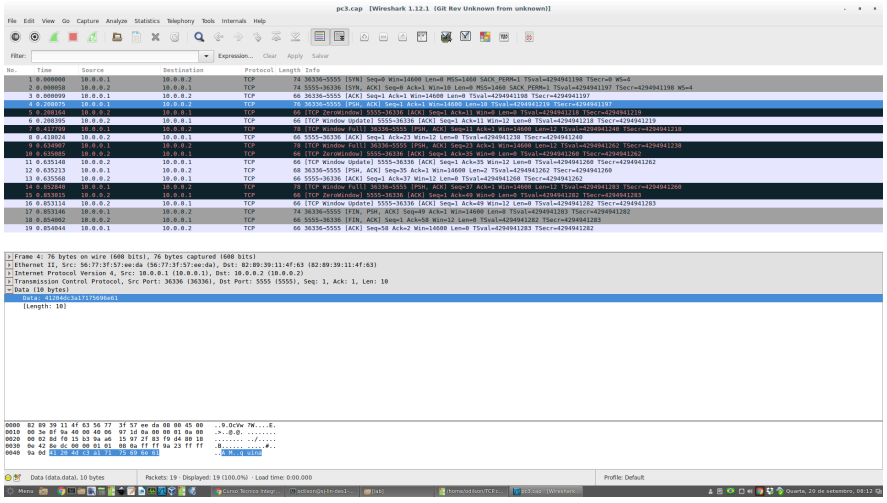


Fig.1 -- Protocolo TCP

14. O primeiro número de sequência na cadeia de bytes que o TCP transmite, é obtido aleatoriamente entre as mais de 4 bilhões de possibilidades (32 bits), ou seja, quase nunca inicia em 0 (zero). Por outro lado, o

comportamento padrão do Wireshark é redefinir esse número de sequência, para sempre iniciar em 0 (zero), para facilitar a leitura humana. Este comportamento pode ser alterado, conforme nossas necessidades, com o seguinte procedimento:

```
Edit >> Preferences >> Protocols >> TCP >> (Habilite/Desabilite) ''Relative sequence numbers'' >> OK
```

1. Desabilite esse comportamento e observe a anote (print) os números de sequência reais e compare com os números de sequência apresentados anteriormente (iniciando em zero) pelo Wireshark.
2. Habilite novamente o número de sequência relativo.

15. Pare o experimento no Imunes:

```
Experiment >> Terminate
```

## 9.4 PARTE 2 - Transmissão com erros: retransmissões

1. Inicie a simulação da rede no Imunes:

```
Experiment >> Execute
```

- Dica: para abrir um terminal de uma das máquinas da rede a ser simulada basta dar um duplo clique sobre a mesma.

2. Copie o arquivo seq_num.txt para a máquina **Transmissor** do Imunes. No terminal da máquina hospedeira (**NÃO do Imunes**) digite::

```
sudo hcp Servidor.txt Transmissor:
```

3. Execute o Wireshark no **Transmissor**:

```
Clique direito do mouse sobre o ícone do Transmissor >> Wireshark >> eth0..
```

4. Adicione o filtro **tcp** ao Wireshark, para limpar os dados apresentados, por enquanto nada aparece no Wireshark, já que não há pacotes sendo transmitidos.
5. Vamos provocar perda de pacotes no enlace editando o campo *loss* (perda) do mesmo. Isso significa que, em média, 10% dos pacotes serão perdidos/adulterados.

```
Duplo clique sobre o enlace (fio vermelho) >> acrescente o valor 10 no campo loss >> Clique em ''Apply''
```

6. No terminal do **Receptor**, limite o *buffer* de recepção e execute o processo servidor:

```
sysctl -w net.ipv4.tcp_rmem='5000 5000 5000'
nc -vvnl -p 5555 > ArqRecebido.txt
```

7. Envie o arquivo seq_num.txt da máquina **Transmissor**, no terminal digite:

```
nc -vvn 10.0.0.21 5555 < Servidor.txt
```

- Obs: Caso receba uma mensagem "No route to host", repita o comando acima. O problema é gerado por perdas sucessivas de mensagens de estabelecimento de conexão do TCP, devido à perda de dados estabelecida.

8. Monitore o Wireshark e, quando perceber um ACK com tamanho um pouco maior de 42000, pare os processos rodando nos terminais do **Transmissor** e **Receptor** com:

```
Ctrl + c
```

9. Analise como os dados foram transmitidos e reconhecidos.
10. Perguntas:
    1. Houve perda de pacotes? Como você identificou isso?
    2. Os pacotes perdidos foram retransmitidos? Justifique.
    3. Qual o significado da mensagem, inserida pelo Wireshark, "TCP Retransmission"? Como você justificaria uma perda de segmento sem acesso a essa informação?
    4. Qual o significado das cores diferenciadas, inseridas pelo Wireshark, nos diversos segmentos apresentados?
11. Pare o experimento no Imunes:

```
Experiment >> Terminate
```

## 9.5 PARTE 3 - Testando a capacidade do TCP de enviar dados de forma duplex

- Agora vamos fazer um pequeno teste de transmissão de arquivos entre dois colegas e observar o comportamento *full-duplex*.

- No experimento, o arquivo de uma máquina será transmitido para outra e vice-versa.

1. Inicie a simulação da rede no Imunes:

```
Experiment >> Execute
```

2. Vamos zerar perda de bits no enlace editando o *loss* do mesmo.

```
Duplo clique sobre o enlace (fio vermelho) >> acrescente o valor 0 no campo loss >> Clique em ''Apply''
```

3. Copie os arquivos para as máquinas do Imunes. No terminal da máquina hospedeira/Linux (**NÃO do Imunes**) digite:

```
sudo hcp Servidor.txt Transmissor:
sudo hcp Cliente.txt Receptor:
```

4. Execute o Wireshark no **Receptor** e aplique o filtro TCP:

```
Clique direito do mouse sobre o ícone do Receptor >> Wireshark >> eth0..
```

5. Limite o tamanho do *buffer* do TCP tanto no **Transmissor** quanto **Receptor**, nos seus terminais digite:

```
sysctl -w net.ipv4.tcp_rmem='10000 10000 10000'
```

6. No terminal do **Transmissor**, que fará o papel de servidor por aguardar a conexão do cliente, execute o comando abaixo. Perceba que o Servidor vai enviar (o sinal **<** indica isso) um arquivo e vai receber e salvar (o sinal **>** indica isso) outro do Cliente.

```
nc -vvnl -p 5555 < Servidor.txt > Arq_recebido.txt
```

7. No terminal do **Receptor**, que fará o papel de cliente, execute o comando abaixo. Perceba que ele também vai enviar e receber arquivo do servidor.

```
nc -vvn 10.0.0.20 5555 < Cliente.txt > Arq_recebido.txt
```

8. Pare os processos rodando nos terminais do **Transmissor** e **Receptor** com:

```
Ctrl + c
```

9. Confira o conteúdo dos arquivos recebidos no **Transmissor** e **Receptor**:

```
cat Arq_recebido.txt
```

    1. Os arquivos foram corretamente trocados entre as duas máquinas? Dica: Responda observando o conteúdo dos arquivos, que são exclusivos e bem criativos :).

10. Pare a captura de pacotes no Wireshark.
11. Adicione o filtro **tcp** ao Wireshark, para limpar os dados apresentados.
12. Analise como os dados foram transmitidos e reconhecidos.
13. Perguntas:

    1. Onde pode ser observado a comunicação *full-duplex*? Obs.: Foque a análise nos segmentos que contém [PSH, ACK].
    2. Qual é a relação entre os comandos no terminal tanto do cliente como do servidor com a comunicação *full-duplex*?
    3. Como os ACKs são propagados, em pacotes exclusivos ou de carona (*piggyback*) com os dados?

# 10 TCP: Controle de congestionamento e equidade

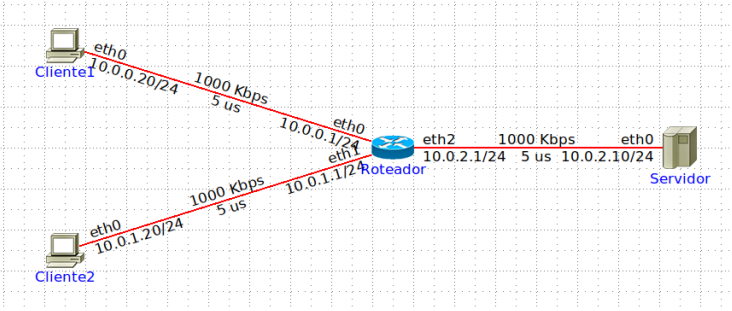
## 10.1 Objetivos

- Visualização, através de gráficos, do **controle de congestionamento** e a consequente **equidade** do protocolo TCP.
- Visualização, através de gráficos, da disputa por banda entre os protocolos TCP e UDP.
- Utilização do software Iperf (https://iperf.fr/) (iperf –h para help) para gerar tráfego entre duas máquinas - **cliente** e **servidor** - e permitir a observação do comportamento da disputa de banda.
- Utilização do software Imunes (http://imunes.net/) para simulação de redes "complexas".

## 10.2 Topologia de rede para experimetação

- O roteiro será executado sobre máquinas virtuais, através do uso do Imunes (http://imunes.net/).

- Para realização dos ensaios será montada a rede virtual apresentada na Figura.
- Observe que na figura todos os enlaces são iguais e limitados a 1 Mbps com *delay* de 5 us.
- Os dois clientes vão disputar o enlace único entre o roteador e servidor.



Dois clientes "disputando" o enlace com o servidor

## 10.3 Parte 1: Somente fluxos TCP

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4
http://redes.sj.ifsc.edu.br/TCP_Controle_de_congestionamento_e
_equidade.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/TCP_Controle_de_congestionamento_e_equidade.imn
```

   - Perceba que será apresentada uma rede com um roteador, dois clientes e um servidor de rede. Todos interligados por enlaces de 1 Mbps.

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

5. Limite o tamanho do *buffer* do TCP tanto no **Servidor** quanto no **Cliente1**:

```
sysctl -w net.ipv4.tcp_rmem='10000 10000 10000'
```

   - Para copiar comando para os terminais das máquinas virtuais: copie o texto desejado, selecione o terminal da máquina desejada (duplo clique sobre a mesma) e clique sobre a "rodinha" do mouse que o texto será colado.

6. No **Servidor** execute:

```
iperf -s -p 2000 & iperf -s -p 2001 & iperf -s -p 2002 &
```

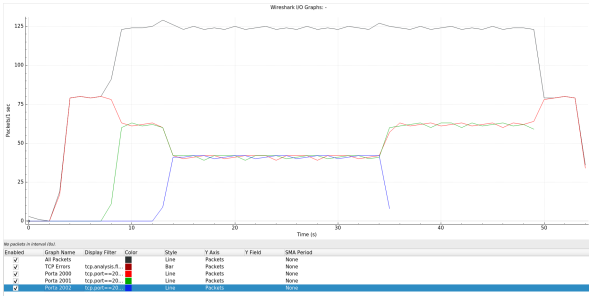
7. No **Roteador** execute o Wireshark:

```
Clique com o botão direito do mouse sobre o Roteador >> Wireshark >> eth2...
```

8. No **Cliente1** execute (copie a três linhas e cole no terminal adequado e em seguida tecle <Enter>):

```
iperf -c 10.0.2.10 -f m -i 1 -t 50 -p 2000 & \
(sleep 5 && iperf -c 10.0.2.10 -f m -i 1 -t 40 -p 2001) & \
(sleep 10 && iperf -c 10.0.2.10 -f m -i 1 -t 20 -p 2002) &
```

9. Fique monitorando o **Cliente1** até a tela parar de ser atualizada, aproximadamente 50 s.
10. Pare os processos no **Cliente1** e **Servidor** utilizando CTRL-C.
11. Pare a captura de dados no Wireshark.
12. No wireshark acesse **Statistics** >> **IO Graph** e, na tela que abrir, ajuste TODOS os parâmetros para obter um gráfico similar ao apresentado na Figura 2:
    1. Clique em **Add a new graph** (sinal de + no canto inferior esquerdo)
       - X Enabled
    - No **Graph 2** altere o filtro (Duplo clique sobre o campo **Display Filter**) para **tcp.port==2000**
    - No **Graph 3** altere o filtro para **tcp.port==2001**
    - No **Graph 4** altere o filtro para **tcp.port==2002**


Captura de 3 fluxos de dados

13. Salve o print desse gráfico no relatório.
14. Responda:
    1. Explique detalhadamente o significado de cada parâmetro dos comandos acima, tanto do cliente quanto do servidor.
    2. Explique os filtros aplicados no gráfico do Wireshark.
       - Quais são os 4 gráficos apresentados?
       - Há uma relação de valor entre as curvas?

- Qual é esta relação?

3. Por que a curva vermelha se sobrepõe a curva preta nos primeiros 5 segundos, a partir do início da transmissão?
4. Qual é a relação entre a curva preta e as curvas vermelha e verde no intervalo entre 6 e 10 segundos, a partir do início da transmissão?
5. Explique a relação entre as 4 curvas e o comando do cliente no intervalo entre 10 e 30 segundos, a partir do início da transmissão.
6. Qual é o mecanismo do TCP que explica a grande oscilação das curvas, principalmente percebida no intervalo entre 10 e 30 segundos, a partir do início da transmissão?

15. Pare a simulação no Imunes:

```
Experiment >> Terminate
```

## 10.4 Parte 2: Fluxos TCP mais UDP

Agora vamos dificultar a vida do TCP incluindo um tráfego UDP. O gráfico gerado deverá apresentar a competição pelo meio de transmissão entre os diversos fluxos de dados.

1. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

2. Limite o tamanho do *buffer* do TCP no **Servidor**, **Cliente1** e **Cliente2**:

```
sysctl -w net.ipv4.tcp_rmem='10000 10000 10000'
```

- Para copiar comando para os terminais das máquinas virtuais: copie o texto desejado, selecione o terminal da máquina desejada (duplo clique sobre a mesma) e clique sobre a "rodinha" do mouse que o texto será colado.

3. No **Servidor** execute:

```
iperf -s -u -p 2000 & iperf -s -p 2001 & iperf -s -p 2002 &
```

4. No **Roteador** execute o Wireshark:

```
Clique com o botão direito do mouse sobre o Roteador >> Wireshark >> eth2...
```

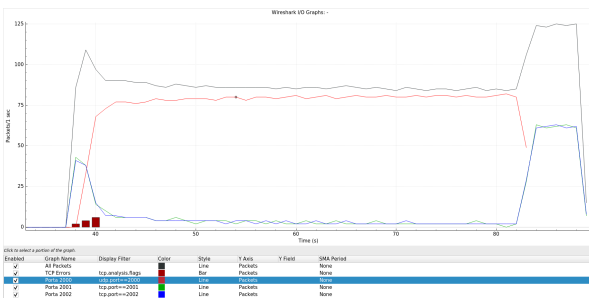
5. A próxima etapa deve ser executada "simultaneamente" nos **Cliente1** e **Cliente2**.
   1. Para isso copie o texto abaixo e cole no terminal do **Cliente1**, ainda NÃO tecle <Enter>:

```
sleep 5 && iperf -u -c 10.0.2.10 -f m -i 1 -t 25 -p 2000 -b 10000000
```

   2. Copie o texto abaixo e cole no terminal do **Cliente2**, ainda NÃO tecle <Enter>:

```
iperf -c 10.0.2.10 -f m -i 1 -t 50 -p 2001 &  iperf -c 10.0.2.10 -f m -i 1 -t 50 -p 2002
```

   3. Tecle <Enter> no **Cliente2** e **Cliente1**, NESSA ORDEM, "simultaneamente".
6. Fique monitorando o **Cliente2** a tela parar de ser atualizada, aproximadamente 50 s.
7. Pare os processos no **Cliente1**, **Cliente2** e **Servidor** utilizando CTRL-C.
8. Pare a captura de dados no Wireshark.
9. No wireshark acesse **Statistics** >> **IO Graph** e, na tela que abrir, ajuste TODOS os parâmetros para obter um gráfico similar ao apresentado na Figura:
   1. Clique em **Add a new graph** (sinal de + no canto inferior esquerdo)
      - X Enabled
   - No **Graph 2** altere o filtro para **tcp.analysis.flags**
   - No **Graph 3** altere o filtro para **udp.port==2000**
   - No **Graph 4** altere o filtro para **tcp.port==2001**
   - No **Graph 5** altere o filtro para **tcp.port==2002**



Captura de 2 fluxos de dados TCP mais um fluxo UDP

10. Salve o gráfico no relatório.
11. Responda:
    1. Explique detalhadamente o significado de cada parâmetro dos comandos acima, tanto do cliente quanto do servidor.
    2. Qual a relação dos filtros aplicados no gráfico e os comandos executados no terminal.
       - Quais são os 5 gráficos apresentados?

   - Há uma relação de valor entre as curvas?
   - Qual é esta relação?
3. O que ocorreu com os fluxos TCP após a entrada do fluxo UDP?
4. Em que momento houve erros no TCP? Qual é a relação desse momento com o UDP?
5. O que ocorreu com os fluxos TCP após o término do fluxo UDP?

12. Como desafio, compare o comportamento dos vários fluxos de dados, no segundo experimento, acrescentando mais fluxos UDP e TCP, por exemplo, 2 fluxos TCP e 2 fluxos UDP, todos iniciando e terminando em tempos distintos.

# 11 Interligação de duas redes através de um roteador

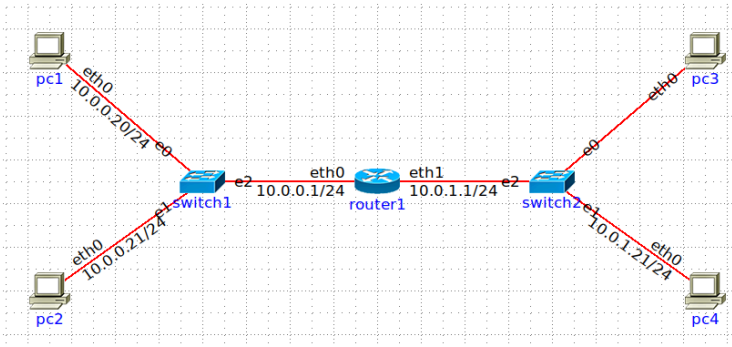
## 11.1 Objetivos

1. Introdução ao mundo IP
2. Verificação das configurações de interfaces de rede
3. Verificação de tabelas de roteamento nos hospedeiros e no roteador
4. Verificação de movimentação de pacotes (rotas) em roteadores

### 11.1.1 Referências

- Comando ifconfig no Linux (https://www.certificacaolinux.com.br/comando-linux-ifconfig/)
- Guia de Administração da Camada IP no Linux (http://linux-ip.net/html)

## 11.2 Topologia de rede para experimetação




## 11.3 Procedimento

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4 http://redes.sj.ifsc.edu.br/Roteador_com_duas_redes.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/Roteador_com_duas_redes.imn
```

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

   - Ignore (*dismiss*) a mensagem de erro apresentada para o **pc3**. O erro é proposital.
   - Observe que a rede é composta de 4 PCs (**pc1** - **pc4**), 1 roteador (**router1**) e 2 switchs. O roteador possui duas interfaces de rede, com seus respectivos IPs - camada 3, que interliga as duas sub-redes. Cada *switch* tem 3 interfaces, mas sem IPs, camada 2.

5. Anotar os endereços de hardware (ou MAC) e IP de cada dispositivo na rede. No terminal de cada PC execute:

```
ifconfig
```

ou

```
ip a
```

6. Observar, interpretar e anotar a tabela de roteamento em todos os hospedeiros **pc1** - **pc4** e no roteador **router1**. Identificar os *default gateways* em cada PC.

```
route
```

7. Observar, "provar" e anotar que pacotes indo do **pc1** para **pc2** são enviados diretamente para **pc2**, ou seja, entrega direta. Explique a entrega direta.
    1. Deixe o ping entre **pc1** e **pc2** executando no **pc1**:

```
ping 10.0.0.21
```

    2. No **router1** capture pacotes com o Wireshark na interface eth0:

```
Clique com o botão direito do mouse sobre o router1 >> Wireshark >> eth0...
```

    3. Observe que não há tráfego de pacotes no **router1**, portanto, entrega direta.
8. Observar, "provar" e anotar que pacotes indo de **pc1** para **pc4** são encaminhados ao roteador e, em seguida, entregues ao destino, ou seja, entrega indireta.
    1. Explique a entrega indireta.

## 11.4 Configuração básica de interface de rede

1. No **pc3** teste a conectividade com os demais **PCs**, por exemplo, fazendo *pings* para o **pc1** e **pc4**:

```
ping 10.0.0.20
ping 10.0.1.21
```

    - Perceba que não há conectividade, não há resposta aos *pings*, dado que a interface de rede do **pc3** não está devidamente configurada.
2. Assim sendo, configure a interface de rede no **pc3**.
    - Anote todos os comandos executados.
    1. Inicie configurando o IP com o comando **ifconfig** (man ifconfig) ou **ip a** (man ip). Dica: Observe a configuração de rede do **pc4**, que está na mesma sub-rede, e tente adaptá-la para o **pc3**.
        - Assim que a configuração do IP for bem sucedida o ping para o **pc4** deverá funcionar.
    2. Tente "pingar" para o **pc1**. Ainda não haverá sucesso, pois não há um roteador devidamente configurado no **pc3**.
    3. Configure o roteador no pc3 com o comando **route** (man route).
        - Assim que a configuração do roteador for bem sucedida o ping para o **pc1**, e qualquer outro PC da rede, deverá funcionar.
    4. O mesmo deverá ser capaz de "pingar" para qualquer outro PC ou ser "pingado".
    5. Execute o comando ping do **pc3** para o **pc4**. Obteve sucesso? Se não corrija as configurações.
    6. Execute o comando ping do **pc3** para o **pc1**. Obteve sucesso? Se não corrija as configurações.
    7. Execute o comando ping do **pc2** para o **pc3**. Obteve sucesso? Se não corrija as configurações.

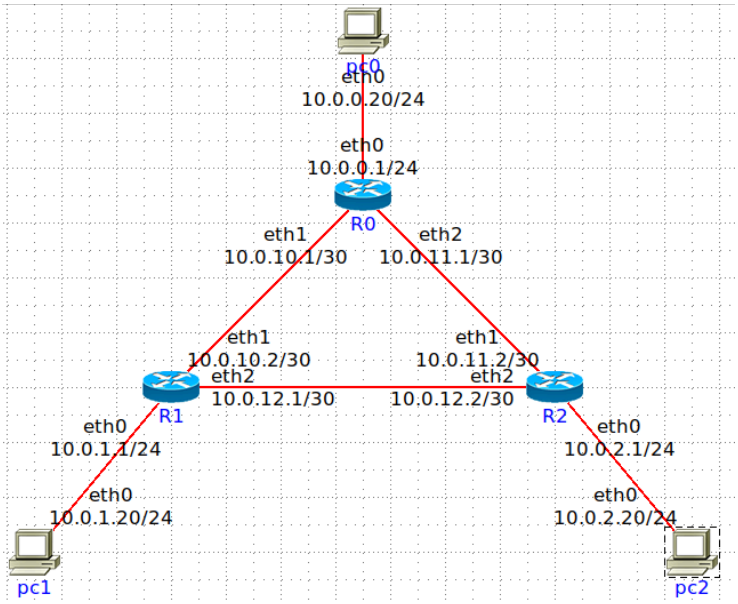
# 12 Tabelas Estáticas de Roteamento

## 12.1 Objetivos

- Analisar o funcionamento de roteadores com tabelas estáticas de roteamento.
- Verificar a entrega direta e indireta de pacotes.
- Analisar *loops* em rede.

## 12.2 Topologia de rede para experimetação

Em todos os experimentos será utilizado como base a seguinte topologia de rede:




## 12.3 Tabelas estáticas de roteamento

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4 http://redes.sj.ifsc.edu.br/3_roteadores_tab_estaticas.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/3_roteadores_tab_estaticas.imn
```

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

- Observe que a rede é composta de 3 PCs (**pc0** - **pc2**) e 3 roteadores roteador (**R0** - **R2**).

5. Testes de conectividade de enlace e configuração do *default gateway*.
    1. Por exemplo, no **pc0** execute o comando:

```
ping 10.0.0.1
```

    Obteve sucesso? Sim ou não e por quê?
    2. Teste a conectividade do **pc0** executando o comando:

```
ping 10.0.10.1
```

    Obteve sucesso? Sim ou não e por quê? Qual foi o erro observado?
    3. Por exemplo, no **pc0** execute o comando:

```
ping 10.0.10.2
```

    Obteve sucesso? Sim ou não e por quê? Qual foi o erro observado?
    4. Configure o roteador padrão em todos os PCs. Adapte o comando exemplo do **pc0** para todos os PCs:

```
route add -net default gw 10.0.0.1
```

    - Com este comando estamos: i) adicionando (*add*) uma rota ii) do tipo rede (*net*) iii) rota padrão (*default*), que é equivalente a 0.0.0.0 iv) com o roteador (gw - *gateway*) v) 10.0.0.1 que identifica a interface do roteador, **R0**, diretamente conectado ao host **pc0**, no caso.

    5. Teste novamente a conectividade, no **pc0** execute o comando:

```
ping 10.0.10.1
```

    e

```
ping 10.0.10.2
```

Obteve sucesso? O comportamento foi o mesmo das tentativas anteriores? Sim ou não e por quê? Qual foi o erro observado?

6. Com os ping do item anterior ativos (um a cada tempo) rode o **Wireshark** no **R0** (clique com o botào direito do mouse sobre o **R0** e em seguida no menu **wireshark eth0**).
    1. Qual a origem e destino dos pacotes? Explique?
    2. Qual a diferença no ping entre os dois itens?

6. Iniciando o roteamento.
    1. Deixe o **ping** do do **pc0** para o **R1** e o **wireshark - eth0** no **R0** rodando e estabeleça uma rota no roteador **R1** com o comando:

    ```
    route add -net 10.0.0.0/24 gw 10.0.10.1
    ```

    O que ocorreu com o **ping** e o **wireshark**? Por quê?

    - Com este comando estamos: i) adicionando (*add*) uma rota ii) do tipo rede (*net*) iii) para a rede 10.0.0.0/24 iv) com o roteador (gw - *gateway*) v) 10.0.10.1 que identifica a interface do roteador, **R0**, diretamente conectado ao roteador **R1**.

    2. Em todos os roteadores crie rotas para todas as redes. Em cada roteador deve-se criar 3 rotas, para as sub-redes "distantes", não diretamente conectadas. Lembre-se que os enlaces diretos já criam automaticamente rotas para as respectivas sub-redes diretamente conectadas ao equipamento, ou seja, entrega direta. Se tudo estiver correto, **todos** os PCs e roteadores devem pingar entre si.

    - Crie rotas sempre pelo caminho mais curto, por exemplo, do **R0** para a rede do **pc1** e **pc2** passando por **R1** e para **R2** respectivamente.

    3. Trace e anote as rotas entre os *hosts* através do **traceroute**.

7. Testando a queda de enlace.
    1. Com todas as rotas em perfeito funcionamento, gere um **ping** do **pc0** para o **pc2** e execute **wireshark eth0** no **R0** , em seguida "derrube" o enlace entre o **R0** e **R2**. Por exemplo, no **R2** execute o comando:

    ```
    ifconfig eth1 down
    ```

    ou

    ```
    ip link set eth1 down
    ```

    O que ocorreu com o **ping** e o **wireshark**? Por quê? Com este enlace comprometido qual seria a solução para a continuidade de funcionamento de toda a rede?

8. Pare e saia do experimento no Imunes:

```
Experiment >> Terminate
File >> Close
```

## 12.4 Testando campo TTL com *loop* na rede

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4 http://redes.sj.ifsc.edu.br/3_roteadores_tab_estaticas_com_loop.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/3_roteadores_tab_estaticas_com_loop.imn
```

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

5. Execute o Wireshark na interface **eth1** do **R0** e **R2** e na **eth2** do **R1**.
6. Gere um tráfego único a partir do **pc0** para o **pc2**:

```
ping -c1 10.0.2.20
```

7. Pare a captura em todos os Wiresharks.
8. Qual mensagem de erro foi recebida no terminal do **pc0**?

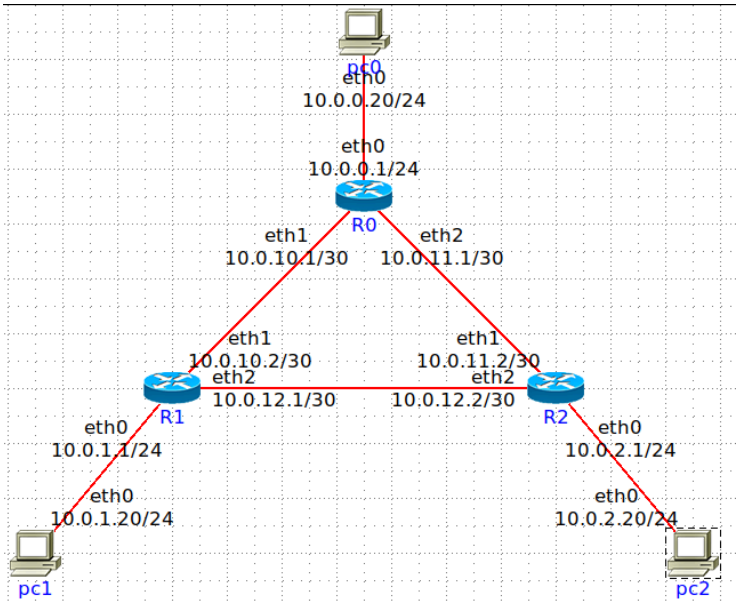
9. Analisando as capturas dos Wireshark responda:
    1. Aproximadamente em qual roteador o pacote foi descartado? Procure pelo menor valor de **ttl**.
    2. Qual o significado da linha, apresentada no terminal do **pc0**, com o seguinte conteúdo parcial: *Time-to-live exceeded (Time to live exceeded in transit)*?
    3. Explique qual o objetivo do campo ttl no cabeçalho IP?

# 13 Protocolos de roteamento dinâmicos - RIP e OSPF

## 13.1 Objetivo

1. Analisar o funcionamento dos protocolos dinâmicos de roteamento RIP e OSPF.
    1. No funcionamento normal.
    2. Na queda de um enlace.
    3. Na recomposição do enlace.
2. Comparar o desempenho de ambos os protocolos.

## 13.2 Topologia de rede para experimetação




## 13.3 Parte 1 - RIP

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4 http://redes.sj.ifsc.edu.br/3_roteadores_RIP.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/3_roteadores_RIP.imn
```

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

    - Observe que a rede é composta de 3 PCs (**pc0** - **pc2**) e 3 roteadores roteador (**R0** - **R2**).
    - A rede será iniciada plenamente funcional, com os roteadores configurados para rodarem o protocolo RIP.
5. Teste a funcionalidade da rede, por exemplo, no **pc0** execute o comando:

```
ping 10.0.2.20
```

6. Anote as rotas do **pc0** para o **pc1** e **pc2**:

```
traceroute 10.0.1.20
traceroute 10.0.2.20
```

7. Anote as tabelas de roteamento de todos os roteadores:

```
route
```

8. Interprete as tabelas de roteamento, diferenciando entrega direta e indireta.
9. Vamos provocar a queda de um enlace, em seguida restabelecer e analisar todo o processo.
    1. Deixe um ping entre o **pc0** e **pc2** rodando.
    2. Deixe o Wireshark rodando na interface eth2 do **R2**.
    3. Desative o enlace **R0-R2**. No **R2** execute:

    ```
    ifconfig eth1 down
    ```

    ou

    ```
     ip link set eth1 down
    ```

    4. Monitorando o ping, aguarde até o retorno das repostas ao mesmo. É comum demorar até uns 2-3 minutos.
    5. Qual o tempo aproximado, medido no relógio, para reativação das repostas do ping?
    6. Anote o número de sequência do último ping com sucesso antes da "derrubada" do enlace e o primeiro após a retorno da funcionamento normal do ping.
    7. Anote a **nova** rota do **pc0** para o **pc2** e a compare com a mesma rota obtida anteriormente:

    ```
    traceroute 10.0.2.20
    ```

    8. Anote novamente as tabelas de roteamento de todos os roteadores:

    ```
    route
    ```

10. Reative o enlace **R0-R2**. No **R2** execute:

```
ifconfig eth1 up
```

ou

```
 ip link set eth1 up
```

    1. Em algum momento o ping deixou de funcionar?
    2. Aguarde por volta de uns 2 minutos e anote novamente a rota do **pc0** para o **pc2**:

    ```
    traceroute 10.0.2.20
    ```

11. Identifique e aponte as diferenças entre as rotas com e sem a queda de enlace. Obs: estão relacionados com a interface desativada.
12. A partir das mensagens do Wireshark responda:
    - É possível usar o filtro rip, para limpar a visualização.
    - Clique sobre a mensagem e expanda o campo *Routing Information Protocol* na janela central, será possível visualizar mensagens do tipo *IP Address: 10.0.12.0, Metric: 16*
    - Os roteadores são identificados por seus IPs.
    - O campo *Metric* indica o número de saltos do roteador em questão até a rede destino.

    1. Tente compreender as mensagens RIPv2 trocadas desde o início explicando-as.
    2. Justifique/explique o valor das métricas (1, 2, 3, ..., 16).
    3. Qual o intervalo aproximado na troca de mensagens?
    4. Qual o número (No.) da mensagem onde a rede apresentou problemas com rotas (obs: retire o filtro rip e procure no número de sequência dos pings (seq) os números de sequência anotados do terminal, item acima).
    5. Quais e quantas mensagens (número) são trocadas entre os roteadores para restabelecer as rotas?
    6. Pesquise o significado do endereço 224.0.0.9.

## 13.4 Parte 2 - OSPF

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4 http://redes.sj.ifsc.edu.br/3_roteadores_OSPF.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/3_roteadores_OSPF.imn
```

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

- Observe que a rede é composta de 3 PCs (**pc0** - **pc2**) e 3 roteadores roteador (**R0** - **R2**).
- A rede será iniciada plenamente funcional, com os roteadores configurados para rodarem o protocolo OSPF.

5. Teste a funcionalidade da rede (pode ocorrer um atraso inicial na formação da rotas), por exemplo, no **pc0** execute o comando:

```
ping 10.0.2.20
```

- Se o *ping* não funcionar imediatamente aguarde até obter respostas, o protocolo está em ação para determinar as melhores rotas.

6. Anote as rotas do **pc0** para o **pc1** e **pc2**:

```
traceroute 10.0.1.20
traceroute 10.0.2.20
```

7. Vamos provocar a queda de um enlace, em seguida restabelecer e analisar todo o processo.
   1. Deixe um ping entre o **pc0** e **pc2** rodando.
   2. Deixe o Wirezhark rodando na interface eth2 do **R2**.
   3. Desative o enlace **R0-R2**. No **R2** execute:

      ```
      ifconfig eth1 down
      ```

      ou

      ```
      ip link set eth1 down
      ```

   4. Monitorando o ping, aguarde até o retorno das repostas ao mesmo. É comum praticamente não percebermos falhas.
   5. Anote novamente a rota do **pc0** para o **pc2**:

      ```
      traceroute 10.0.2.20
      ```

   6. Reative o enlace **R0-R2**. No **R2** execute:

      ```
      ifconfig eth1 up
      ```

      ou

      ```
      ip link set eth1 up
      ```

8. A partir das mensagens do Wireshark responda:
   - É possível usar o filtro ospf, para limpar a visualização.
   - Perceba que com o protocolo OSPF, diferentemente do RIP, não há trocas periódicas de mensagens do protocolo de roteamento.
   - Só haverá trocas quando o protocolo sentir necessidade de alguma mudança de rota, por exemplo, com a queda de um enlace.

   1. Quais as mensagens trocadas pelo protocolo OSPF são observadas no WireShark?
   2. Qual o tempo aproximado para a total recuperação das rotas?
   3. As mensagens trocadas pelos roteadores são distintas quando comparadas ao uso do RIP?
   4. Explique as mensagens "*Hello Packet*", "*LS Update*" e "*LS Acknowledge*".
   5. Houve diferença no tempo de atualização das rotas quando comparado ao RIP? Explique?

# 14 *Neighbor Discovery* e roteamento estático no IPv6

Este roteiro foi baseado no material disponível no Livro - Laboratório de IPv6 (http://ipv6.br/pagina/livro-ipv6/).

Slides de endereçamento IPv6 (http://docente.ifsc.edu.br/odilson/RED29004/IPv6.pdf).

Guia didático de endereçamento IPv6 (http://docente.ifsc.edu.br/odilson/RED29004/enderec-v6.pdf) obtido de http://ipv6.br/.

# 14.1 Objetivos do laboratório

- Um primeiro contato com o protocolo IPv6 (https://pt.wikipedia.org/wiki/IPv6)
- Compreender o funcionando do *Neighbor Discovery*, o equivalente ao ARP (*Address Resolution Protocol*) do IPv4, que em resumo é uma tabela contendo a relação ente IPs e MACs.
- Aprender configurações básicas de interfaces IPv6 no Linux
- Aprender configurações básicas de rotas IPv6

## 14.1.1 Introdução teórica

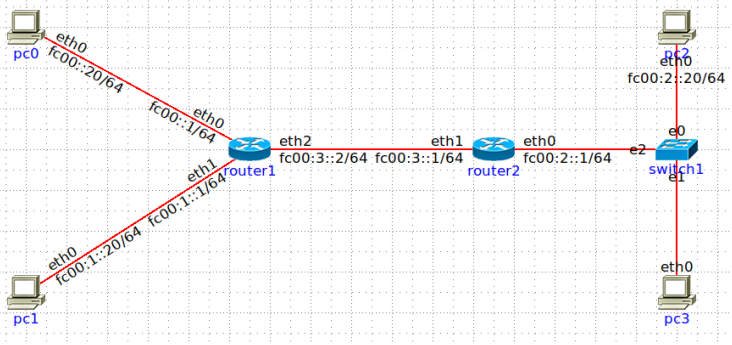
Obs.: texto copiado literalmente de: Laboratório de IPv6 (http://ipv6.br/pagina/livro-ipv6/).

A **descoberta de vizinhança** por meio do protocolo *Neighbor Discovery* no IPv6 é um procedimento realizado pelos nós de uma rede para descobrir endereços físicos dos dispositivos vizinhos presentes no mesmo enlace. A função deste protocolo se assemelha à função do ARP e do RARP no IPv4.

- O procedimento é iniciado quando um dispositivo tenta enviar um pacote cujo endereço físico de destino é desconhecido. O nó solicitante envia uma mensagem *Neighbor Solicitation* (NS) para todos os nós do enlace pertencentes ao grupo *multicast solicited-node* (ff02::1:ffXX:XXXX), de modo que XX:XXXX são os últimos 24 bits do endereço IPv6 em que está interessado.
- É possível notar que, por uma coincidência dos últimos 24 bits, é bastante provável que apenas o nó de destino faça realmente parte deste grupo. Isto é um *truque* interessante do IPv6 para diminuir o tráfego deste tipo de pacote na rede.
- Na mensagem NS, o endereço IPv6 a ser resolvido é informado no campo *Target*. O campo *Source link-layer address* informa ao nó de destino o endereço MAC do nó de origem, poupando-o de ter que fazer o mesmo procedimento no sentido inverso.
- O nó de destino, dono do IPv6 requisitado, ao receber este pacote, envia uma mensagem *Neighbor Advertisement* (NA) como resposta diretamente ao nó requisitante. O seu endereço físico será informado no campo *Target link-layer address*.
- A informação de mapeamento entre endereços IP e endereços físicos é armazenada em uma tabela chamada *neighbor cache*. Nela também fica registrado o *status* de cada destino, informando se o mesmo é alcançável ou não.

# 14.2 Topologia de rede para experimetação

A figura abaixo apresenta o diagrama esquemático da rede a ser montada/analisada. Observe que todos os IPv6 *Global Unicast* já estão definidos na mesma, são esses IPs que utilizaremos em nosso experimento.




# 14.3 Roteiro de atividades

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4 http://redes.sj.ifsc.edu.br/IPv6.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/IPv6.imn
```

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

   - Ignore a mensagem de erro, o mesmo é proposital.
   - Observe que a rede é composta de 4 PCs (**pc0** - **pc3**) e 2 roteadores (**R0** - **R1**) e 1 switch.

5. Execute o wireshark em R1:

```
clique com o botão direito do mouse sobre seu ícone >>  Wireshark >> eth0...
```

6. Vamos testar o *Neighbor Discovery*, anote a saída do comando. A partir do **pc2** execute:

```
ndisc6 -m fc00:2::1 eth0
```

   - Esse comando descobre e retorna o *MAC address* da interface de rede que possui o endereço IPv6 informado. É equivalente ao protocolo ARP do IPv4.
7. Observe que todas as interfaces de rede já estão pré-configuradas, exceto do **pc3**.
8. Vamos adicionar o endereço IPv6 à interface de rede no **pc3**:

```
ip addr add fc00:2::21/64 dev eth0
```

9. Faça um **ping6** entre o **pc3** ao **pc2**:

```
ping6 fc00:2::20
```

   - Se tudo estiver devidamente configurado, deve-se obter sucesso no ping entre o **pc3** e **pc2**. Entrega direta ou indireta?
10. Faça um **ping6** entre o **pc3** ao **pc0**.
    - Obteve sucesso? Sim ou não e por quê?
11. No **pc3** use o seguinte comando para verificar como ficou a configuração dos endereços da interface de rede. O resultado é similar ao apresentado pelo comando **ifconfig**:

```
ip addr show dev eth0
```

12. No **pc3**, liste a tabela de roteamento com o comando:

```
ip -6 route show
```

13. No **pc3**, acrescente o *default gateway* com o seguinte comando:

```
ip -6 route add default via fc00:2::1 dev eth0
```

    - Confira novamente a tabela de roteamento do **pc3**.
14. Faça novamente um **ping6** entre o **pc3** ao **pc0**.
    - Obteve sucesso? Sim ou não e por quê?
15. Pode-se conferir se as rotas, nos roteadores ou qualquer host, com o comando:

```
ip -6 route show
```

16. A partir do computador **pc0** use o comando **traceroute6** e anote a rota para todos os demais PCs.
17. Pare a captura no Wireshark.
18. Baseado na captura de pacotes do Wireshark explique o processo de descoberta de vizinhança (*Neighbor Solicitation* - **NS** e *Neighbor Advertisement* - **NA**), citando os endereços de **multicast** e **link local** utilizados. Obs.: ao final do roteiro há alguns exemplos de mensagens.
19. Numa mensagem do tipo *Neighbor Solicitation* qual é o endereço IPv6 de origem e destino? Explique/defina ambos.
20. Em todos os hosts rode o comando

```
ip -6 neighbor show
```

    1. Qual é a funcionalidade desse comando?
    2. Qual é o significado do conteúdo dessa tabela?
    3. A tabela mostrada em cada um dos casos é compatível com o diagrama da rede montado?
    4. Por que, por exemplo, na tabela do **pc2** não há uma referência explícita ao **pc0**?
21. Explique sucintamente as diferenças na comunicação baseada em IPv4 e IPv6.

- Alguns exemplos de campos visualizáveis para uma mensagem do tipo *Neighbor Advertisement*, presentes nas camadas destacadas:

1. **Source** (camada Ethernet)
   - A origem é o endereço MAC da interface do dispositivo que enviou a resposta.
2. **Protocol** (camada Ethernet)
   - Indica que a mensagem utiliza IPv6.
3. **Next header** (camada IPv6)
   - Indica qual é o próximo cabeçalho. Neste caso, o valor 58 (0x3a) refere-se a uma mensagem ICMPv6.
4. **Source** (camada IPv6)

- A origem é o endereço IP da interface diretamente ligada ao enlace em que a requisição foi recebida.

5. **Destination** (camada IPv6)
6. **Type** (camada ICMPv6)
   - Indica que a mensagem é do tipo 136 (Neighbor Advertisement).
7. **Flags** (camada ICMPv6)
   - Uma mensagem NA possui três flags:

   1. Indica se quem está enviando é um roteador. Neste caso, o valor marcado é 0, pois não é um roteador.
   2. Indica se a mensagem é uma resposta a um NS. Neste caso, o valor marcado é 1, pois é uma resposta.
   3. Indica se a informação carregada na mensagem é uma atualização de endereço de algum nó da rede. Neste caso, o valor marcado é 1, pois está informando o endereço pela primeira vez.

8. **Target Address** (camada ICMPv6)
   - Indica o endereço IP associado às informações das flags. Neste caso, é o próprio endereço da interface do dispositivo em questão.

# 15 Path MTU Discovery e Tunelamento IPv6 para IPv4

Este roteiro foi baseado no material disponível no Livro - Laboratório de IPv6 (http://ipv6.br/pagina/livro-ipv6/).

## 15.1 Objetivos do laboratório

- O objetivo desta experiência é mostrar o funcionamento do mecanismo de descoberta de MTU no IPv6.

## 15.2 Introdução teórica

Obs.: texto copiado literalmente de: Laboratório de IPv6 (http://ipv6.br/pagina/livro-ipv6/).

O MTU é o tamanho máximo de pacote suportado em um determinado enlace de rede. Caso seja necessário enviar um pacote maior do que o MTU do enlace é necessário fragmentá-lo. No IPv6, a fragmentação dos pacotes é realizada apenas na origem. Este procedimento não é realizado (e nem permitido) em roteadores intermediários, como ocorre no protocolo antigo, o IPv4. Isto tem o intuito de reduzir o custo de processamento nos roteadores, seguindo o princípio de manter a inteligência da Internet nas extremidades da rede. Numa rede IPv6, quem está enviando os pacotes tem então que conhecer o MTU do caminho até o destino. Se o caminho for composto por vários seguimentos com MTUs diferentes, valerá, na prática, o menor deles. Essa informação é obtida por meio do Path MTU Discovery (PMTUD), definido na RFC 1981 (McCann et al., 1996). O processo de PMTUD assume que o MTU de todo o caminho é igual ao do primeiro salto. Caso o tamanho dos pacotes enviados seja maior do que o suportado por algum enlace ao longo do caminho, o roteador irá descartá-lo e enviará uma mensagem ICMPv6 packet too big, contendo tanto a mensagem de erro quanto o valor do MTU do enlace seguinte. Após o recebimento dessa mensagem, o nó de origem passa a limitar o tamanho dos pacotes de acordo com o MTU indicado. Isso é repetido até que o tamanho do pacote seja igual ou inferior ao menor MTU do caminho. Os dispositivos armazenam o MTU para cada destino em uma tabela chamada destination cache, não sendo necessário repetir a descoberta a cada pacote enviado. Caso o pacote seja enviado a um grupo multicast, o tamanho utilizado será o menor MTU de todo o conjunto de destinos. Implementações minimalistas de IPv6 podem não realizar a descoberta de MTU e utilizar 1280 bytes como tamanho máximo para os pacotes.

## 15.3 Parte 1: Path MTU Discovery

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4 http://redes.sj.ifsc.edu.br/PathMTU.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/PathMTU.imn
```

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

   - Observe que a rede é composta de 2 PCs e 1 roteador.

5. Altere o valor de MTU no dispositivo **router1**:

```
ip link set eth1 mtu 1400
```

6. Verifique o valor de MTU no dispositivo **router1**:

```
ip addr show
```

- Na interface eth1 deverá constar mtu 1400

7. Altere o valor de MTU no dispositivo **pc2**:

```
ip link set eth0 mtu 1400
```

8. Verifique o valor de MTU no dispositivo **pc2**:

```
ip addr show
```

- Na interface eth0 deverá constar mtu 1400

9. Execute o wireshark no **pc1**:

```
clique com o botão direito do mouse sobre seu ícone >>  Wireshark >> eth0...
```

10. Abra um terminal no **pc1** e verifique a conectividade IPv6 com o **pc2**:

```
ping6 -s 1500 -M want -c 4 fc00:1::20
```

- Veja que o comando ping6, com os parâmetros apresentados, configura os pacotes enviados para conterem 1500 bytes de tamanho, por meio da opção -s 1500, e a interface para permitir a fragmentação de pacotes, utilizando a opção -M want.

11. Pare a captura de pacotes e aplique, no Wireshark, o filtro icmpv6.
12. Procure pelo pacote **Packet Too Big**.
    1. Qual é o tipo (type) e código (code) do pacote? Está de acordo com a teoria? (iana.org (https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml))
    2. Procure o campo **MTU: 1400**, qual o significado desta informação?
13. Procure pelos pacotes **Echo (ping) request**.
    1. Analise os mesmos na camada 3 (IPv6):
       1. quantos fragmentos os mesmos possuem?
       2. Qual o tamanho de cada fragmento?
    2. Onde (máquina) foi realizada a fragmentação?
       1. Como você chegou a esta conclusão?
    3. Procure pelos pacotes **Echo (ping) reply**.
       1. Eles também estão fragmentados? Prove.
       2. Qual o sentido dessa fragmentação?

- O resultado mostra que, para a topologia apresentada, o menor valor de MTU ao longo do caminho foi descoberto corretamente a partir do primeiro pacote. Este foi descartado ao chegar em uma rede cujo o limite do MTU era de 1400 bytes. Na sequência, os pacotes foram enviados fragmentados de acordo com o valor de MTU descoberto e passaram a transitar corretamente pela rede.

## 15.4 Parte 2: Tunelamento IPv6 para IPv4

1. Baixe o arquivo de configuração da rede, no terminal digite:

```
wget -4 http://redes.sj.ifsc.edu.br/IPv4_IPv6_tunel.imn
```

2. Execute o Imunes.
3. Carregue o arquivo de configuração:

```
File >> Open >> /home/aluno/IPv4_IPv6_tunel.imn
```

4. Inicie a simulação no Imunes:

```
Experiment >> Execute
```

- Observe que a rede é composta de 2 PCs e 1 roteador.

5. Verifique as interfaces de rede de todos os hosts, **pc1**, **pc2** e **router1**:

```
ifconfig
```

- Perceba que o roteador não tem configuração de IPv6.

6. Teste a conectividade da rede. No terminal do **pc1** execute:

```
ping 10.0.1.20
ping6 fc00:1::20
```

7. Ambos os pings tiveram sucesso? Sim ou não e por quê?
8. Vamos configurar o túnel IPv6 para IPv4. No terminal do **pc1** execute

```
ip tunnel add toPC2 mode sit ttl 64 remote 10.0.1.20 local 10.0.0.20
ip link set dev toPC2 up
ip -6 route add fc00:1::20 dev toPC2
ip addr show
ip -6 route show
```

   1. Descreva e interprete a funcionalidade de cada um dos comandos acima.
9. Vamos configurar o túnel IPv6 para IPv4. No terminal do **pc2** execute

```
ip tunnel add toPC1 mode sit ttl 64 remote 10.0.0.20 local 10.0.1.20
ip link set dev toPC1 up
ip -6 route add fc00::20 dev toPC1
ip addr show
ip -6 route show
```

10. Execute o wireshark na eth0 **router1**:

```
clique com o botão direito do mouse sobre seu ícone >>  Wireshark >> eth0...
```

11. Abra um terminal no **pc1** e verifique a conectividade IPv6 com o **pc2**:

```
ping6 -c3 fc00:1::20
```

12. Pare a captura de pacotes e aplique, no Wireshark, o filtro icmpv6.
13. Procure por pacotes **Echo (ping) request** e **Echo (ping) reply** e clique sobre um deles.
    1. Veja se os dados contidos nos pacotes conferem com a teoria (há túnel?).
    2. Qual é o tipo (type), na camada *Ethernet 2*? Está de acordo com a teoria? ([1] (https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml))
    3. Analise um pacote na camada 3:
       1. Quantas camadas 3 ele possui?
       2. Quais protocolos?
       3. Quais endereços apresentados em cada versão do IP? São condizentes com os endereços configurados nos hosts?
       4. No *Internet Protocol Version 4*, procure o campo do cabeçalho **Protocol**, qual seu conteúdo?
       5. No *Internet Protocol Version 6*, procure o campo do cabeçalho **Next Header**, qual seu conteúdo?
    4. Baseado em suas respostas anteriores explique qual versão do protocolo está "tunelada" em qual outra. Justifique sua resposta.

---