# Laboratório de Redes de Computadores - Trabalho Final

**Monitor de Tráfego de Rede em Tempo Real**

## Objetivo

O objetivo geral deste trabalho é desenvolver uma ferramenta para monitoramento de tráfego de rede em tempo real utilizando *raw sockets*. A ferramenta deve ser capaz de capturar, interpretar e classificar pacotes de rede, além de fornecer uma interface do usuário simples para visualizar contadores e estatísticas de tráfego de rede e ao mesmo tempo escrever um histórico dos pacotes recebidos em um arquivo de log.

Os objetivos específicos incluem:

- Desenvolvimento de uma aplicação usando *sockets raw*;
- Estudo do funcionamento dos protocolos de rede e do relacionamento entre as camadas;
- Entender como os pacotes de dados são estruturados e como eles podem ser interpretados para extrair informações úteis;
- Entender o tráfego de uma rede local e os tipos de protocolos normalmente trafegados;
- Utilizar uma estrutura de rede já definida como parte do problema.

## Descrição

Você foi contratado para desenvolver uma aplicação que deverá analisar o tráfego de uma rede, a qual possui uma estrutura onde um conjunto de clientes acessam a internet por meio de um servidor proxy. Tanto os clientes, quanto o proxy, residem na mesma LAN. A única forma dos clientes terem

acesso à rede externa é por meio de um programa que atua como um cliente do proxy, o qual encapsula o tráfego dos clientes em pacotes IP com informações forjadas, para maior segurança. O servidor proxy irá desencapsular o tráfego de cada cliente e encaminhar o mesmo, atuando como um roteador. Infelizmente, você não possui controle sobre a estrutura da rede, portanto para analisar o tráfego será necessário desenvolver uma aplicação para monitoramento em tempo real, que executará na máquina que atua como servidor proxy.

O monitor de tráfego de rede em tempo real a ser desenvolvido deve ser capaz de identificar e classificar diferentes tipos de pacotes de dados que passam pela rede. Isso inclui identificar protocolos de rede como IP, TCP, UDP, ICMP, DHCP etc., bem como origem, destino e tamanho dos pacotes. O programa a ser desenvolvido deve possuir uma interface modo texto que apresenta contadores para cada tipo de pacote recebido. Para cada protocolo, seu programa deve apresentar um conjunto de informações que serão mantidos em arquivos de log do tipo *.csv*, para as camadas da pilha de procotolos TCP/IP. A interface a ser monitorada é a interface virtual *tun0* no servidor proxy[1], onde o tráfego dos pacotes de todos os clientes pode ser interceptado.

**Arquivos de *log* e monitor de tráfego**

O arquivo de log para o protocolo IP na camada de rede (camada_internet.csv) deve contemplar os protocolos IPv4, IPv6 e ICMP e ter as seguintes colunas:

- Data e hora que o pacote foi capturado;
- Nome do protocolo (IPv4, IPv6, ICMP, outro);
- Endereço IP de origem (ex: 100.114.7.75, fe80::ad3e:46fc:abf7:55c9);
- Endereço IP de destino;
- Número identificador do protocolo que está sendo carregado no pacote;
- Outras informações do protocolo (para ICMP);
- Tamanho total do pacote em bytes.

O arquivo de log para camada de transporte (camada_transporte.csv) deve contemplar os protocolos TCP e UDP e ter as seguintes colunas:

- Data e hora que o pacote foi capturado;

[1]Mais detalhes sobre a arquitetura da rede são apresentados adiante.

- Nome do protocolo (TCP, UDP, outro);
- Endereço IP de origem;
- Porta de origem (ex: 8080);
- Endereço IP de destino;
- Porta de destino;
- Tamanho total do pacote em bytes.

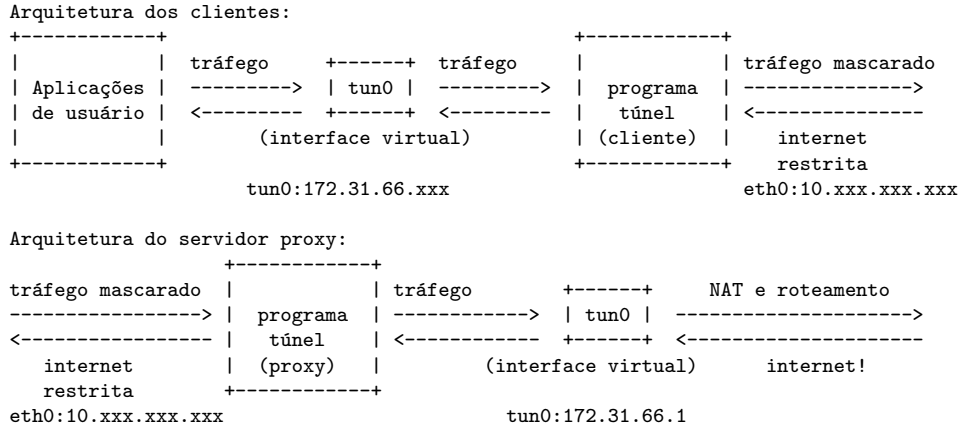
O arquivo de log para camada de aplicação (camada_aplicacao.csv) deve contemplar os protocolos HTTP, DHCP, DNS e NTP e ter as seguintes colunas:

- Data e hora que o pacote foi capturado;
- Nome do protocolo (HTTP, DHCP, DNS, NTP, outro);
- Informações do protocolo (obtidas de seu cabeçalho).

Os arquivos de log devem ser atualizados em tempo real e a qualquer momento deve ser possível dar um "cat"para visualizar o que foi capturado até o momento. A ferramenta de monitoramento deve apresentar para cada cliente (representado por um IP da rede túnel) informações sobre máquinas remotas acessadas e para cada uma as portas, protocolos, conexões, pacotes trocados e o volume de tráfego associado.

### Túnel de tráfego

O túnel já utilizado na empresa que solicitou os seus serviços possui uma arquitetura ilustrada no diagrama:



Como apresentado, o tráfego das aplicações dos clientes passa por um programa túnel por meio de uma interface virtual. O programa túnel encapsula e mascara o tráfego e injeta o mesmo na interface física, a qual está conectada a uma LAN. O servidor proxy (túnel) recebe o tráfego encapsulado, desencapsula o mesmo e o injeta em uma interface virtual. A partir desse ponto, a máquina onde está executando o proxy utiliza uma regra de *iptables* (com *masquerading*) para implementar uma NAT, e roteia o tráfego para a internet.

### Configuração do túnel

Para compilar o programa que implementa o túnel, é necessário ter instalado em um ambiente Linux os pacotes *build-essentials* e *iptables*. No container da disciplina não é necessário instalar nenhuma dependência. Após descompactar os fontes, execute o comando *make*. É importante que o ambiente onde o túnel irá executar seja uma LAN, composta por diversos computadores Linux conectados à mesma rede (como o LabRedes). A LAN pode ser implementada por computadores físicos, VMs ou containers.

### Executando o túnel no modo servidor proxy

Para carregar o túnel no modo proxy, execute o comando abaixo. Não é necessário usar *sudo* caso a máquina seja um container. Para saber o nome da interface e seu IP, utilize o comando *sudo ifconfig* ou *ip addr*.

```
$ sudo ./traffic_tunnel <interface do servidor> -s
```

Após a carga do programa túnel, verifique se a interface *tun0* foi criada e se o roteamento foi habilitado no kernel.

### Executando o túnel no modo cliente

Para carregar o túnel no modo cliente, utilize um script diferente para cada cliente, e execute o comando abaixo. São fornecidos 2 scripts como referência. Assim como no servidor proxy, é necessário saber o nome da interface.

```
$ sudo ./traffic_tunnel <interface do cliente> -c client1.sh
```

### Container Linux (opcional)

O container pode ser instalado com o comando abaixo (substitua *podman* por *docker* caso esteja utilizando Docker). Isso pode ser feito em máquinas Linux ou Windows.

```
$ podman pull ghcr.io/sjohann81/labredes:latest
```

Instruções sobre o uso de containers são apresentadas em `https://github.com/sjohann81/linux_container`.

### Resultados e entrega

Este trabalho deverá ser realizado em grupos de 3 ou 4 integrantes, sendo a data de entrega 24/11 e apresentação será nos dias 24/11 e 01/12. É importante que todos os grupos estejam aptos a apresentarem o trabalho a partir do início da aula. Qualquer linguagem de programação poderá ser utilizada para o desenvolvimento. Para a entrega, é esperado que apenas um dos integrantes envie pelo Moodle, até a data e hora especificadas, um arquivo *.tar.gz* ou *.zip* com os nomes dos integrantes, contendo o código fonte completo do projeto e um relatório descrevendo a implementação. É importante que no relatório seja apresentada uma análise, incluindo o uso de *screenshots* que demonstrem o funcionamento do projeto no ambiente de rede (físico ou virtual).

**Importante:** Não serão aceitos trabalhos entregues fora do prazo. Trabalhos que não compilam ou não executam não serão avaliados. Trabalhos que estiverem rodando no mesmo host ou que não utilizem *sockets raw* serão desconsiderados.