# Redes de Computadores – 2018
# Segundo Trabalho
# Analisador de Pacotes

Prof. Ronaldo Alves Ferreira

## 1 Descrição do Trabalho

Analisadores de pacotes são ferramentas de enorme valor para a depuração de protocolos de rede. Neste trabalho, você implementará um programa que lê dados de pacotes de uma interface de rede e imprime esses dados formatados de acordo com os protocolos presentes nos pacotes.

O projeto está dividido em múltiplas partes. Sua nota será baseada na quantidade de opções que você implementar.

Seu analisador de pacotes deve rodar da seguinte maneira:

```
% xnoop -i <interface> [options] [filter]

interface:  nome da interface de onde os pacotes serão lidos.

options:

  -c n (mostra os n primeiros pacotes capturados).

  -n (não efetua tradução de nomes).

  -v (modo verboso - explicado abaixo).

  -V (modo verboso estendido - explicado abaixo).

filter:  filtro de seleção de pacotes (explicado abaixo).
```

# 2 Funcionamento Básico

Se você executar xnoop sem opções ou filtros, ele deve apenas imprimir um resumo dos pacotes lidos da interface especificada após o seu encerramento.

```
% xnoop -i eth0
ethernet frames:        100
ethernet broadcast:     5
ARP                     2
IP                      80
ICMP                    2
UDP                     20
TCP                     50
To this host:           3
```

Para encerrar o programa, você deve pressionar Control+C. Você deve instalar um tratador de sinais para capturar o sinal de encerramento e imprimir as estatísticas.

# 3 Modo Verboso

**-v**

Esta opção imprime uma linha para cada pacote, abaixo estão alguns exemplos.

```
% xnoop -i eth0 -c 4 -v
16:26:06.943243 mail.facom.ufms.br -> (broadcast) ARP Who is 200.129.206.126
16:26:08.927162 mail.facom.ufms.br -> www.facom.ufms.br TCP SMTP sourceport=25 destport=33012
16:26:08.927334 www.facom.ufms.br -> ssh.facom.ufms.br UDP NFS sourceport=2049 destport=121
16:26:11.011644 ssh.facom.ufms.br -> (broadcast) ICMP Destination unreachable (Bad port)
```

```
% xnoop -i eth0 -c 2 -n -v
16:26:19.172885 200.129.206.118 -> 255.255.255.255 ARP Who is 200.129.206.126
16:26:35.294692 200.129.206.118 -> 200.129.206.126 TCP TELNET sourceport=23 destport=33012
```

# 4 Modo Verboso Estendido

### -V

Esta opção estende o modo verboso. No modo estendido, você imprimirá os pacotes com mais detalhes. Abaixo estão alguns exemplos.

```
% xnoop -i eth0 -c 4 -V

 ETHER:  ----- Ether Header -----
 ETHER:
 ETHER:  Packet 1
 ETHER:  Packet size = 210 bytes
 ETHER:  Destination = 8:0:20:1:3d:94,
 ETHER:  Source      = 8:0:69:1:5f:e,
 ETHER:  Ethertype = 0800 (IP)
 ETHER:
 IP:   ----- IP Header -----
 IP:
 IP:   Version = 4, header length = 20 bytes
 IP:   Type of service = 00
 IP:         ..0. .... = routine
 IP:         ...0 .... = normal delay
 IP:         .... 0... = normal throughput
 IP:         .... .0.. = normal reliability
 IP:   Total length = 196 bytes
 IP:   Identification 19846
 IP:   Flags = 0X
 IP:   .0.. .... = may fragment
 IP:   ..0. .... = more fragments
 IP:   Fragment offset = 0 bytes
 IP:   Time to live = 255 seconds/hops
 IP:   Protocol = 17 (UDP)
 IP:   Header checksum = 18DC
 IP:   Source address = 129.144.40.222,
 IP:   Destination address = 129.144.40.200,
 IP:
 UDP:  ----- UDP Header -----
 UDP:
 UDP:  Source port = 1023
 UDP:  Destination port = 2049
 UDP:  Length = 176
 UDP:  Checksum = 0
 UDP:
 UDP: Data: First 64 bytes
 UDP: 5408 0400 2100 0000 ffff ffff ffff c003 "T...!..........."
 UDP: 5408 0400 2100 0000 ffff ffff ffff c003 "T...!..........."
 UDP: 5408 0400 2100 0000 ffff ffff ffff c003 "T...!..........."
```

```
 UDP: 5408 0400 2100 0000 ffff ffff ffff c003 "T...!..........."
 UDP:


ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 0:c0:4f:79:ed:a2,
ETHER:  Source      = 0:0:c:b:47:64,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 40 bytes
IP:   Identification = 25372
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 254 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 4b14
IP:   Source address = 128.211.1.31, lorenzo.cs.purdue.edu
IP:   Destination address = 128.10.3.108, xinu8.cs.purdue.edu
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 41891
TCP:  Destination port = 23 (TELNET)
TCP:  Sequence number = 2009202523
TCP:  Acknowledgement number = 660063620
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
```

```
TCP:        .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0x8080
TCP:  Urgent pointer = 0
TCP:  No options
TCP: Data: (first 64 bytes)
TCP: fffa 1800 5854 4552 4dff f0ff fa23 0069 "....XTERM....#.i"
TCP: 0d0a 0d0a 5375 6e4f 5320 352e 360d 0a0d "....SunOS 5.6..."
TCP: 0000 0000 0000 6e4f 5320 352e 360d 0a0d "......nOS 5.6..."
TCP: fffb 01ff fd01 6c6f 6769 6e3a 200d 0a0d "......login: ..."
TCP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 3
ETHER:  Packet size = 60 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 0:aa:0:a2:ec:fa, Intel
ETHER:  Ethertype = 0806 (ARP)
ETHER:
ARP:  ----- ARP/RARP Frame -----
ARP:
ARP:  Hardware type = 1
ARP:  Protocol type = 0800 (IP)
ARP:  Length of hardware address = 6 bytes
ARP:  Length of protocol address = 4 bytes
ARP:  Opcode 1 (ARP Request)
ARP:  Sender's hardware address = 0:aa:0:a2:ec:fa
ARP:  Sender's protocol address = 128.10.3.201, xinu101.cs.purdue.edu
ARP:  Target hardware address = ?
ARP:  Target protocol address = 128.10.3.31, helga.cs.purdue.edu
ARP:
```

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 4
ETHER:  Packet size = 98 bytes
ETHER:  Destination = 0:c0:4f:79:ed:af,
ETHER:  Source      = 0:0:c:b:47:64, Cisco
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 84 bytes
IP:   Identification = 37847
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 254 seconds/hops
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = 69e3
IP:   Source address = 128.211.1.31, lorenzo.cs.purdue.edu
IP:   Destination address = 128.10.3.107, xinu7.cs.purdue.edu
IP:   No options
IP:
ICMP:  ----- ICMP Header -----
ICMP:
ICMP:  Type = 8 (Echo request)
ICMP:  Code = 0
ICMP:  Checksum = 8dfd
ICMP:
```

# 5 Filtros

Filtros em xnoop devem ser especificados usando a notação pós-fixada. Para que um pacote seja mostrado, a expressão de filtro deve resultar em um valor diferente de 0 no topo da pilha. Um erro durante a avaliação da expressão deve resultar em um 0 no topo da pilha. Operações aritméticas serão feitas sempre usando 4 bytes. Comparações devem usar 8 bytes.

Os operadores são os seguintes:

## Primitivos

- **[0-9][0-9]*** - Empilha um número em decimal. O número deve ser representado em 4 bytes.
- **0x[0-9,A-F,a-f][0-9,A-F,a-f]*** - Empilha um número em hexadecimal. O número deve ser representado em 4 bytes.
- **d.d.d.d** - Empilha um endereço IP . O endereço deve ser representado em 8 bytes, sendo que os quatro bytes menos significativos devem ser preenchidos com o endereço IP e os 4 bytes mais significativos com 0.
- **xx:xx:xx:xx:xx:xx** - Empilha um endereço Ethernet. O endereço Ethernet utiliza os 6 bytes menos significativos e os 2 bytes mais significativos devem ser preenchidos com 0.

## Operadores Lógicos e Aritméticos

- **a b =** ou **a b eq** - Desempilha **a** e **b** da pilha. Se **a** for igual a **b**, empilha 1, senão empilha 0.
- **a b and** - Desempilha **a** e **b** da pilha. Se **a!=0** e **b!=0**, empilha 1, senão empilha 0.
- **a b or** - Desempilha **a** e **b** da pilha. Se **a!=0** ou **b!=0**, empilha 1, senão empilha 0.
- **a not** - Nega o topo da pilha. Se **a==0**, empilha 1, senão empilha 0.
- **a b <operador>** - Desempilha **a** e **b** da pilha. Realiza a operação indicada por **<operador>** e empilha o resultado de volta na pilha. As operações podem ser +,-,*,/ e %. Qualquer erro deve resultar em 0 no topo da pilha.

## Protocolos

- **ip** - Empilha 1 se o pacote for um pacote IP. Caso contrário, empilha 0.
- **udp** - Empilha 1 se o pacote for um pacote UDP. Caso contrário, empilha 0.
- **tcp** - Empilha 1 se o pacote for um pacote TCP. Caso contrário, empilha 0.
- **icmp** - Empilha 1 se o pacote for um pacote ICMP. Caso contrário, empilha 0.
- **arp** - Empilha 1 se o pacote for um pacote ARP. Caso contrário, empilha 0.

## Ethernet

- **etherto** - Empilha o endereço Ethernet de destino.
- **etherfrom** - Empilha o endereço Ethernet de origem.
- **ethertype** - Empilha o tipo do pacote Ethernet.

## IP

- **ipto** - Empilha o endereço IP de destino. Se não for um pacote IP, empilha 0.
- **ipfrom** - Empilha o endereço IP de origem. Se não for um pacote IP, empilha 0.
- **ipproto** - Empilha o campo protocolo do pacote IP. Se não for um pacote IP, empilha 0.

## UDP

- **udptoport** - Empilha a porta UDP de destino. Se não for um pacote UDP, empilha 0.
- **udpfromport** - Empilha a porta UDP de origem. Se não for um pacote UDP, empilha 0.

## TCP

- **tcptoport** - Empilha a porta TCP de destino. Se não for um pacote TCP, empilha 0.
- **tcpfromport** - Empilha a porta TCP de origem. Se não for um pacote TCP, empilha 0.

## ICMP

- **icmptype** - Empilha o tipo de pacote ICMP. Se não for um pacote ICMP, empilha 0.

Exemplos:

Mostra dez pacotes UDP ou TCP vindos de 128.10.3.108.

```
% xnoop -i eth0 -c 10 -V udp tcp or 128.10.3.108 ipfrom eq and
```

Mostra dez pacotes destinados a 0:c0:4f:79:ed:a2.

```
% xnoop -i eth0 -c 10 etherto 0:c0:4f:79:ed:a2 eq
```

Mostra apenas os pacotes TCP.

```
% xnoop -i eth0 tcp
```

Mostra um pacote ARP.

```
% xnoop -i eth0 -c 1 ethertype 0x806 eq
```

IMPORTANTE: Você também deve implementar um programa calc:

```
% calc <filter>
```

Este programa deve avaliar um filtro passado como parâmetro. Ele serve para testar o seu avaliador de filtros. Os operadores de pacote, tais como ip, devem retornar 0 quando usados no programa calc.

# 6 Interface de Entrada

A interface de entrada deve ser acessada usando raw socket e colocada em modo promíscuo para capturar todos os pacotes. A impressão ou não de um pacote será controlada pelo filtro.

# 7 Fontes para Consultas

No campo tipo dos pacotes Ethernet, você deve considerar apenas os valores 0x0800 (IP) e 0x0806 (ARP). Além desse campo, nos pacotes Ethernet você precisa considerar apenas os campos de endereço de origem e endereço de destino. Esses endereços são números de 48 bits e aparecem no início do pacote, antes do campo tipo. Os endereços Ethernet são normalmente mostrados como uma cadeia no formato xx:xx:xx:xx:xx:xx, em que xx representa um número de 8 bits em hexadecimal.

Os valores do campo protocolo de pacotes IP podem ser encontrados no arquivo /etc/protocols em estações Linux.

Os valores das portas (origem e destino) de pacotes TCP e UDP podem ser encontrados no arquivo /etc/services em estações Linux.

Descrições dos campos dos protocolos ARP, IP, ICMP, UDP e TCP podem ser encontradas nas RFCs que definem os protocolos. Para encontrar essas RFCs, basta digitar o nome do protocolo seguido por rfc no Google. As RFCs de interesse são 791 (IP), 793 (TCP), 768 (UDP), 792 (ICMP), 826 (ARP).

# 8 Entrega do Trabalho

O trabalho deverá ser submetido eletronicamente via Moodle. O prazo de entrega se encerra no dia **14 de outubro às 23h59min**. O sistema de submissão será bloqueado nesse horário e você não poderá submeter por outro meio.

Além do código documentado em C, você deve entregar um relatório descrevendo o seu trabalho. Neste relatório, você deve incluir uma breve introdução, decisões de implementação, funcionalidades não implementadas, problemas enfrentados na implementação, etc. O relatório deve ser entregue em um arquivo PDF.

Para a submissão do trabalho, crie um diretório chamado **t2** e inclua o seu código fonte e o arquivo com o relatório. Submeta todo o diretório. Se o seu programa for composto de vários arquivos em C, é recomendado que voce crie também um arquivo Makefile. Remova os arquivos temporários (.o, .bak, etc) antes da submissão.

O trabalho pode ser feito em grupos de no máximo dois alunos. Casos de plágio serão tratados com rigor. Caso você faça o trabalho em grupo, submeta apenas um trabalho e identifique os componentes do grupo no relatório e no código fonte.

# 9 Avaliação

Além da correção do programa, o professor fará uma entrevista com os membros do grupo. Na entrevista, o grupo deverá explicar o funcionamento do programa e responder a perguntas relativas ao projeto.