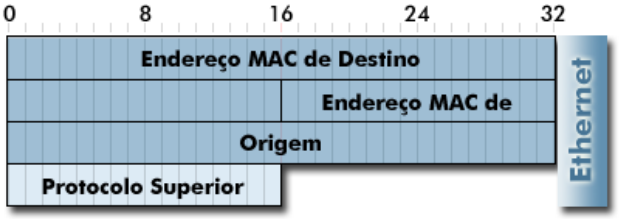
OBJETIVO: Exercitar a manipulação de arquivos/dados binários através da decodificação do TCP/IP.

QUESTÃO ÚNICA

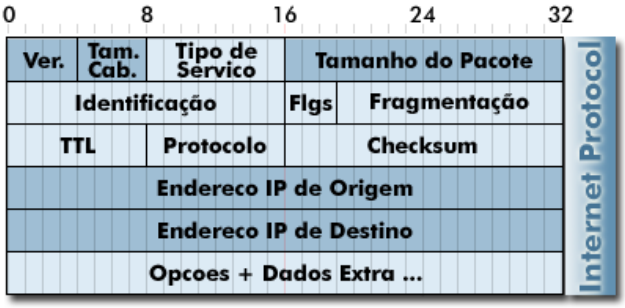
Cada pacote de dados que trafega na Internet é composto por uma série de cabeçalhos de protocolos. Cada um desses protocolos pertence a uma camada de rede. Cada camada faz uma tarefa específica e necessária para possibilitar o envio do pacote de um computador na Internet para outro. A figura "TCP/IP" mostra as principais camadas da Internet (conhecido como pilha "TCP/IP") bem como um dos protocolos mais usados em cada uma dessas camadas. Cada protocolo possui seu "cabeçalho" que contém os dados necessários para o seu funcionamento. O cabeçalho de um protocolo é imediatamente seguido pelo cabeçalho do próximo. Desta forma, um pacote típico de uma conexão HTTP (usando a rede cabeada) terá o cabeçalho do protocolo Ethernet seguido imediatamente pelo cabeçalho do IP, TCP e, finalmente, os dados do HTTP.

Um pacote típico de uma conexão HTTP (com tamanhos mais ou menos proporcionais)			
Cab. Ethernet (14 bytes)	**Cabeçalho IP** (≥20 bytes)	**Cabeçalho TCP** (≥32 bytes)	**HTTP+Dados** (até 1500 bytes)

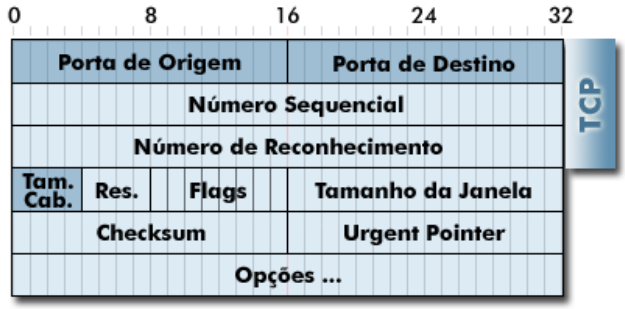
Estes cabeçalhos nada mais são do que "estruturas", e podem ser facilmente acessados como tais. As figuras a seguir ilustram os cabeçalhos do Ethernet, do IP e do TCP, bem como as suas respectivas estruturas. Note que nas estruturas estamos usando os tipos `uint8_t` para inteiros de 8 bits (mesmo que `char`), `uint16_t` para os inteiros de 16 bits (mesmo que `short int`) e `uint32_t` para os inteiros de 32 bits (mesmo que `int`). Para usar estes tipos, é preciso incluir (`include`) a biblioteca `stdint.h`. Note também que os campos de tamanho variável (`Opções + Dados Extra` do IP e `Opções` do TCP) não foram incluídos nas estruturas, mas seus tamanhos podem ser calculados com base nos campos de tamanho dos cabeçalhos (`Tam. Cab.`).




```
typedef struct {
    uint8_t  daddr[6];  // Endereco MAC de destino
    uint8_t  saddr[6];  // Endereco MAC de origem (source)
    uint16_t protocol;  // Protocolo da próxima camada (IP!)
} ethernet_hdr_t;
```




```
typedef struct {
    uint8_t  hdr_len:4, // Tamanho do Cabeçalho,
             version:4; // Versão do IP
    uint8_t  tos;       // Tipo de serviço
    uint16_t tot_len;   // Tamanho total do IP
    uint16_t id;        // Id do pacote
    uint16_t frag_off;  // Fragmentado?
    uint8_t  ttl;       // Tempo de vida
    uint8_t  protocol;  // Protocolo próxima camada (TCP!)
    uint16_t check;     // Checksum
    uint8_t  saddr[4];   // Endereço IP de origem
    uint8_t  daddr[4];   // Endereço IP de destino
} ip_hdr_t;
```




```
typedef struct {
    uint16_t sport;       // Porta TCP de origem
    uint16_t dport;       // Porta TCP de destino
    uint32_t seq;         // Sequência
    uint32_t ack_seq;     // Acknowledgement
    uint8_t  reservado:4, // Não usado
             hdr_len:4;   // Tamanho do cabecalho
    uint8_t  flags;       // Flags do TCP
    uint16_t window;      // Tamanho da janela
    uint16_t check;       // Checksum
    uint16_t urg_ptr;     // Urgente
} tcp_hdr_t;
```

Curiosidade:

Você pode ver estas estruturas que o Linux implementa abrindo os seguintes arquivos:

```
/usr/include/linux/if_ether.h
/usr/include/linux/ip.h
/usr/include/linux/tcp.h
```

Neste trabalho, sua missão será abrir um arquivo de entrada (`argv[1]`) que conterá exatamente um pacote inteiro retirado da Internet. Este arquivo começa imediatamente no cabeçalho do Ethernet, seguido do cabeçalho do IP, depois do TCP e, por último, os dados do HTTP. Um arquivo de exemplo pode ser baixado em:

`http://bit.ly/http-rawcap`

Você deverá imprimir, deste arquivo, todos os dados destacados (mais escuros) nas figuras dos cabeçalhos mostrados anteriormente. Deverá imprimir, também, todo o conteúdo da camada de aplicação (HTTP).

Exemplo de Execução:

```
$ ./read_tcp_ip http.rawcap
Lendo Ethernet ..
  --> MAC de Origem:  00:23:12:c2:17:b6
  --> MAC de Destino: 00:50:56:a7:79:5a
Lendo IP ..
  --> Versão do IP: 4
  --> Tamanho do cabeçalho: 20 bytes
  --> Tamanho do pacote: 440 bytes
  --> Endereço IP de Origem:  10.208.2.69
  --> Endereço IP de Destino: 200.143.247.51
Lendo TCP ..
  --> Porta de Origem:  56056
  --> Porta de Destino: 80
  --> Tamanho do cabeçalho: 32 bytes
Lendo Dados (HTTP) ..
  --> Tamanho dos dados: 388 bytes
  --> Dados:
GET /r/Fallout HTTP/1.1
Host: www.reddit.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent:  Mozilla/5.0  (X11;  Linux  x86_64)  AppleWebKit/537.36  (KHTML,  like  Gecko)  Ubuntu
Chromium/28.0.1500.71 Chrome/28.0.1500.71 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
```

Dicas:

- Camada MAC:
  - Para imprimir um número hexadecimal (usado no endereço ethernet), use o formato `%.2x` do `printf`.
- Camada IP:
  - No `Tamanho do cabeçalho` do IP, este tamanho é em múltiplos de 32 bits (4 bytes) e, portanto, para mostrar este valor em bytes, você precisa multiplicá-lo por 4.
  - No `Tamanho do pacote` do IP, como ele ocupa mais de um byte, você precisa usar a função `ntohs` para convertê-lo para a "ordem de byte" usado pela sua máquina.
  - Para pular as `Opções` do IP, que possuem tamanho variável, você precisará pular (`fseek`) `ip_hdr.hdr_len*4 – sizeof(ip_hdr_t)` bytes.
- Camada TCP:
  - Nas `Porta de Origem` e `Porta de Destino` do TCP, como eles ocupam mais de um byte, você precisa usar a função `ntohs` para convertê-los para a "ordem de byte" usado pela sua máquina.
  - No `Tamanho do cabeçalho` do TCP, este tamanho é em múltiplos de 32 bits (4 bytes) e, portanto, para mostrar este valor em bytes, você precisa multiplicá-lo por 4.
  - Para pular as `Opções` do TCP, que possuem tamanho variável, você precisará pular (`fseek`) `tcp_hdr.hdr_len*4 - sizeof(tcp_hdr_t)` bytes.
- Camada HTTP:
  - Para saber o tamanho total de bytes na camada de aplicação (HTTP), use
    `int tam_dados = ntohs(ip_hdr.tot_len) - (ip_hdr.hdr_len*4) – (tcp_hdr.hdr_len*4);`
  - Daí, é só imprimir os caracteres (`fgetc`).

Nota aos Geeks:

- Para capturar dados diretamente da placa de rede, usamos a biblioteca `libpcap` (`http://www.tcpdump.org/`). Esta biblioteca possui funções (e.g., `pcap_loop`) que capturam o pacote e executam uma função sua passando como parâmetro o ponteiro para o início da camada de enlace (Ethernet, se for rede cabeada; ou WiFi, se for rede sem fio). A partir daí, os passos são parecidos com os feitos neste laboratório.

ENTREGA DO LABORATÓRIO

O laboratório deve ser entregue até o dia 05/06/2025. Para entregar, envie o código-fonte para horacio@icomp.ufam.edu.br com o assunto "Entrega do 9o Laboratório de LPA".