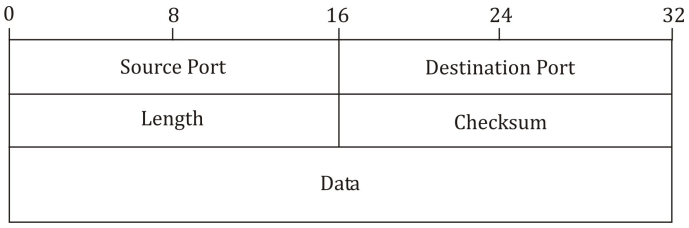
Processando pacotes UDP

O User Datagram Protocol (UDP) é um protocolo simples utilizado em redes de computadores. A transmissão de dados de acordo com o protocolo UDP utiliza a noção de datagrama, que pode ser ilustrada visualmente como:



De maneira intuitiva, um datagrama pode ser entendido como um registro contendo os seguintes campos representados visualmente na figura anterior:

- Porta de origem (source port): número de 16 bits que representa o endereço que enviou o datagrama.
- Porta de destino (destination port): número de 16 bits que representa o endereço que receberá o datagrama.
- Tamanho (length): número de 16 bits que determina o número de bits que forma o datagrama.
- Checksum: número de 16 bits utilizado para validar a consistência dos dados no datagrama.
- Data: Sequências de 32 bits que representam a informação transmitida usando o protocolo UDP.

O objetivo dessa prova é a implementação de um parser de pacotes UDP. Para isso, primeiramente vamos representar um bit, usando o tipo a seguir.

```
data Bit = O | I deriving (Eq, Ord)
```

Os diferentes campos do datagrama UDP podem ser representados pelo tipo Field a seguir.

```
data Field
  = Field {
    size :: Int
    , content :: [Bit]
    } deriving (Eq, Ord)
```

Field é representado como um registro que armazena o seu valor em bits (representado como [Bit]) e um inteiro que representa número de bits armazenado nessa lista. Como exemplo, considere o seguinte campo de dois bits:

```
ex :: Field
ex = Field 2 [O, I]
```

A utilidade do tipo Field é representar as restrições de tamanho presentes na especificação de datagramas UDP. Representaremos um datagrama pelo seguinte tipo de dados:

```
data UDP
  = UDP {
    source :: Field      - 16 bits
    , destination :: Field- 16 bits
    , plength :: Field   - 16 bits
    , checksum :: Field  - 16 bits
    , pdata :: [Field]   - list of 32 bit fields
    } deriving (Eq, Ord)
```

Cada campo do tipo UDP representa um componente do datagrama, utilizando o tipo de dados Field que representa sequências de bits de um certo tamanho. Como exemplo, o campo source é representado por um valor de tipo Field contendo uma lista de 16 bits.

Com base no apresentado, faça o que se pede.

1. Implemente um parser para o tipo Bit, de forma que o dígito 1 seja representado pelo construtor I e o dígito 0 por O.

   ```
   bitParser :: Parser Char Bit
   bitParser = ⊥
   ```

2. Usando o parser para bits, podemos construir um parser que processa uma sequência de n bits. Implemente o parser:

   ```
   bitList :: Int → Parser Char [Bit]
   bitList n = ⊥
   ```

   Um parser bitList n processa uma sequência de n bits retornando-os como resultado.

3. Usando parser bitList implemente um parser para o tipo Field.

   ```
   fieldParser :: Int → Parser Char Field
   fieldParser n = ⊥
   ```

   Note que fieldParser n retorna um valor de tipo Field contendo uma lista de n bits e o inteiro n como campos deste registro.

4. Finalmente, usando os tamanhos de sequências de bits de cada campo de um datagrama UDP, construa um parser para o tipo UDP.

   ```
   udpParser :: Parser Char UDP
   udpParser = ⊥
   ```