

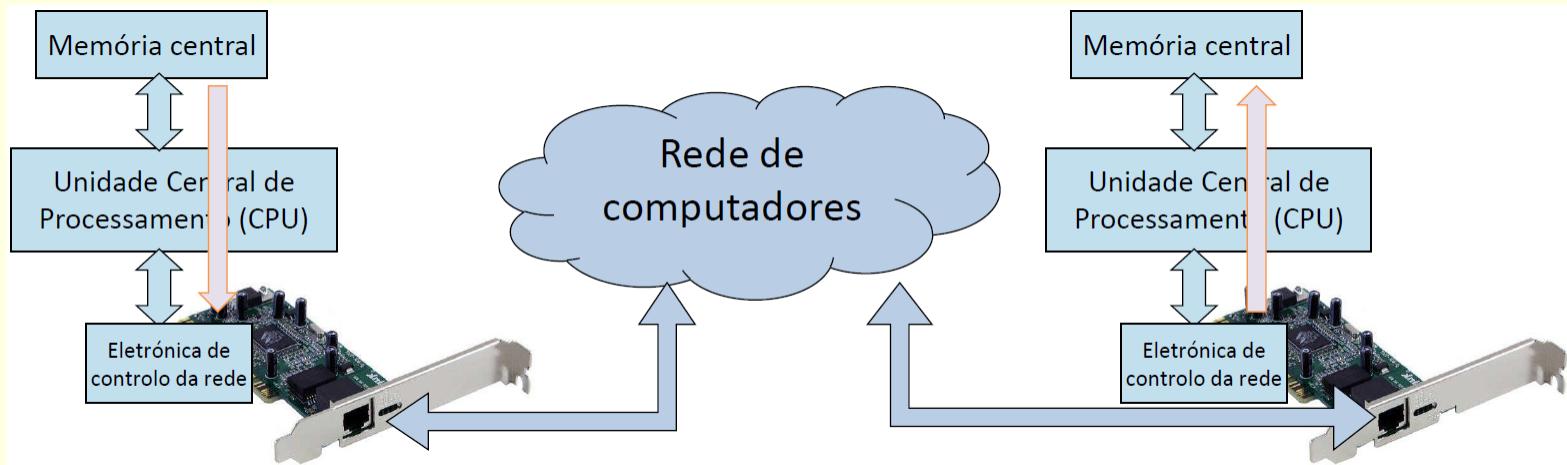
# *Fundamentos de Sistemas de Operação*

Unix Windows NT Netware Mac OS DOS/V/S Vax/VMS  
Linux Solaris HP/UX AIX Mach Chorus

*Sistemas distribuídos:  
Redes de Computadores - introdução*

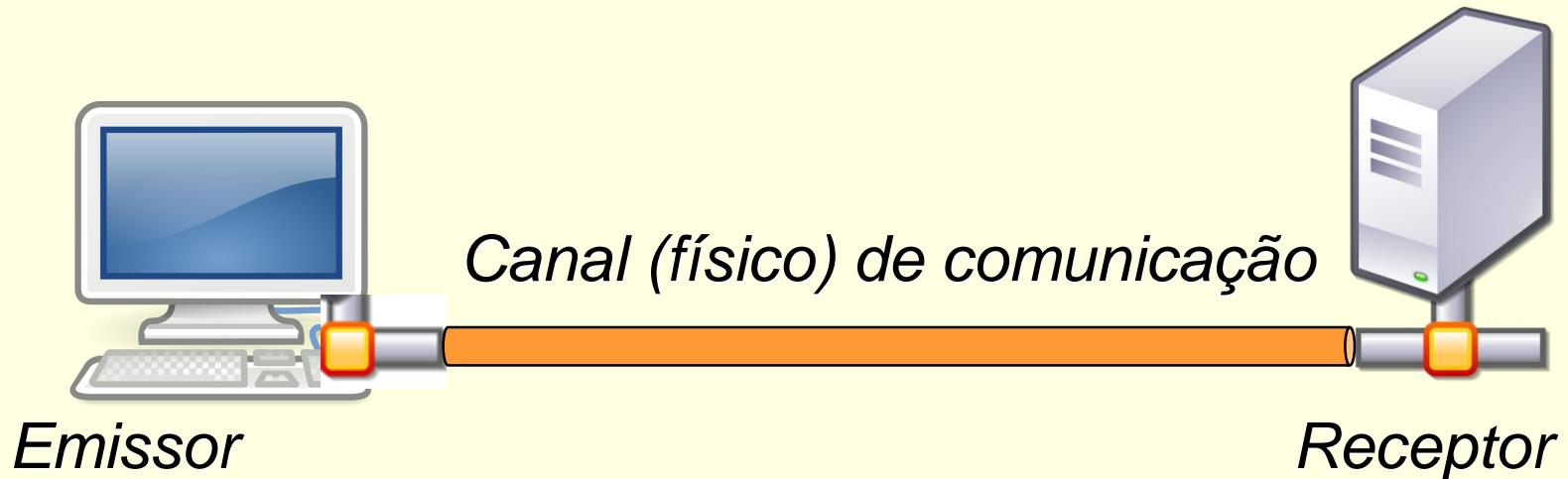
# Comunicação entre computadores (1)

- Envio e Recepção de dados (visão baseada no hardware)



# *Comunicação entre computadores (2)*

- Envio e Recepção de dados (visão esquemática)



# Comunicação entre computadores (3)

- *Canal de comunicação*
  - O que flui no canal? Impulsos eléctricos? Impulsos luminosos? Ondas “radio” (electromagnéticas)?
- *Identificação dos interlocutores:*
  - Como identificar (em sistemas que correm múltiplos processos) quem envia? Como identificar quem deve receber?
- *Utilização do canal*
  - Pode haver vários interlocutores a “dialogarem” ao mesmo tempo, ou só pode “falar” um de cada vez? E se for este ultimo caso, como é que se “passa a vez”? Quem começa?

# Comunicação entre computadores (4)

## □ *Fiabilidade da comunicação*

- *Como saber que o que se recebe corresponde ao que foi enviado, i.e., que não houve perdas, duplicações, alterações (corrupção ou hacking), etc.?*

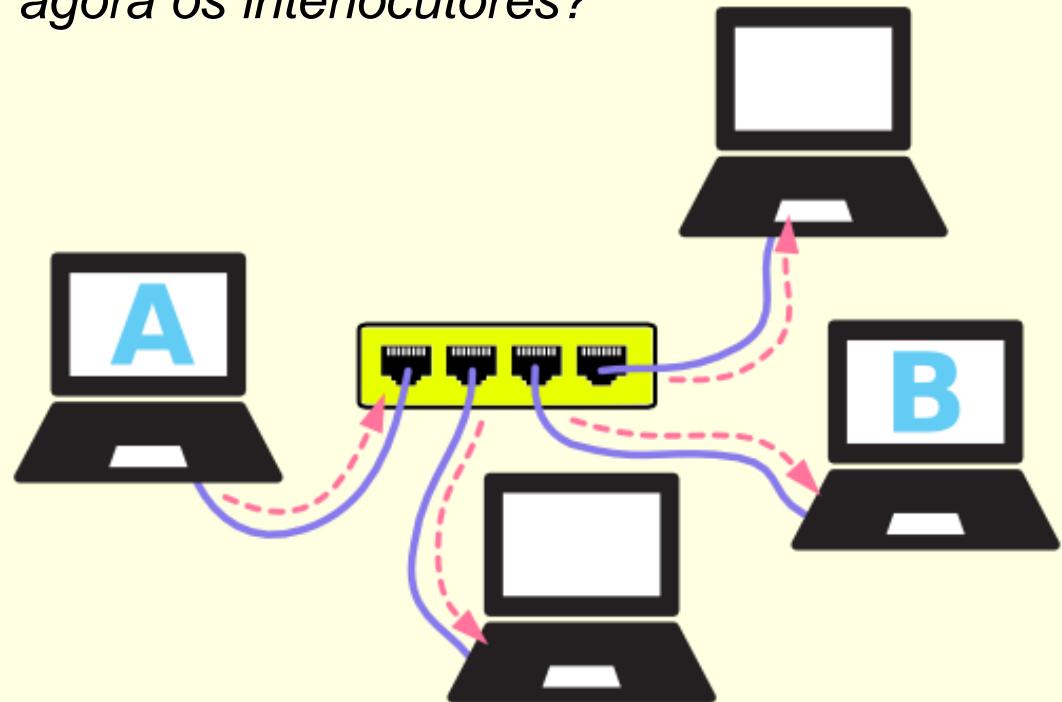
## □ *Velocidade e latência:*

- *Enviados os primeiros “sinais”, quanto tempo demoram estes a ser recebidos? E, uma vez que estão já a ser recebidos, qual a cadência de chegada de “novos sinais”?*
- *A que distância se consegue comunicar?*

□ ...

# Comunicação entre computadores (5)

- Uma rede de computadores...
  - E agora? Qualquer um pode comunicar com qualquer outro?
  - Como identificar agora os interlocutores?



# *Comunicação entre computadores (6)*

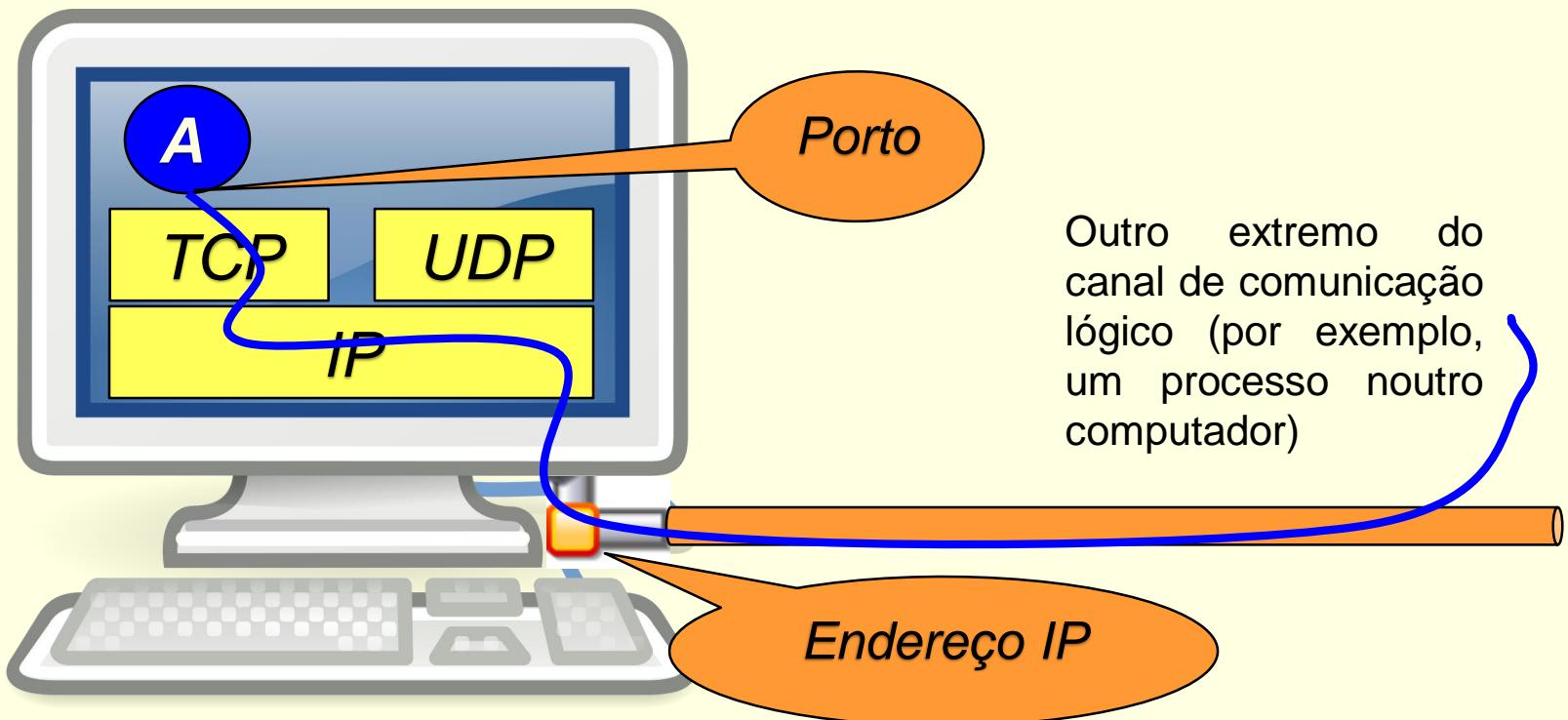
- Os tópicos aflorados são do domínio das
  - Redes de computadores e
  - Protocolos de comunicação
- Serão estudados em outras UCs, nomeadamente
  - Redes de Computadores
  - Sistemas Distribuídos
  - Arquitecturas e Protocolos de Redes de Computadores

# *As nossas simplificações... (1)*

- *Um computador em rede é caracterizado por*
  - *Usar um protocolo de comunicações chamado IP (**Internet Protocol**, aqui na versão 4) e ter atribuído um endereço IP (poderá ter mais, mas não estudaremos isso)*
  - *Ter dois outros protocolos (camadas) “por cima” do IP: o protocolo UDP (User Datagram Protocol) e o protocolo TCP (Transmission Control Protocol)*
- *Dotado destes 3 protocolos: IP, UDP e TCP,*
  - *Uma “entidade” (processo, thread) residente num computador pode comunicar com uma (ou mais) entidades residentes noutras computadores (ou até no mesmo)*

# *As nossas simplificações... (2)*

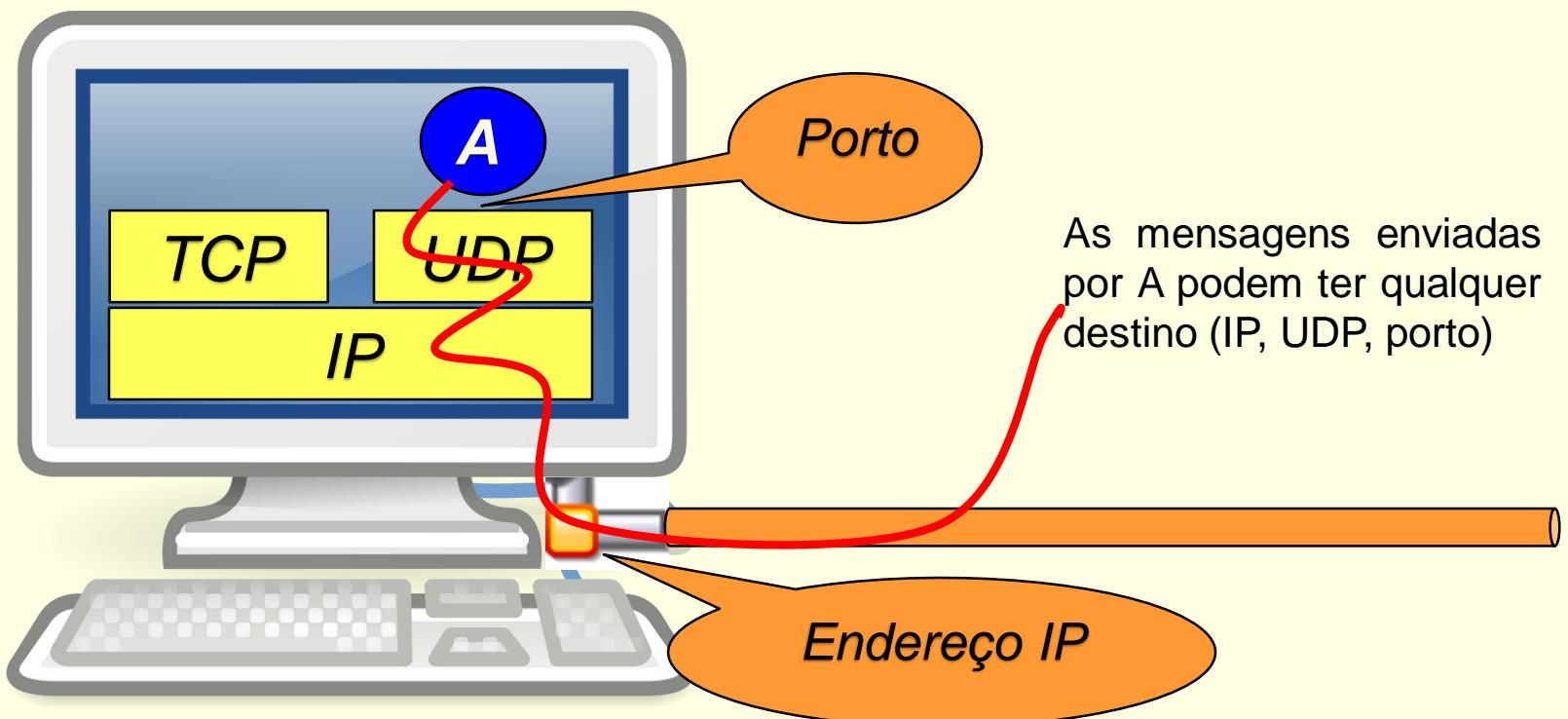
*Extremo de um canal de comunicação definido pelo terno (IP, protocolo, porto)*



# *As nossas simplificações... (3)*

Unix Windows NT Netware Mac OS DOS/V/S Vax/VMS  
Linux Solaris HP/UX AIX Mach Chorus

Ponto de comunicação definido pelo terno (IP, protocolo, porto)



# *Interacções e mensagens... (1)*

## □ *Troca de mensagens*

- Os protocolos de comunicação permitem que dois (ou mais) interlocutores troquem dados enviando mensagens, grupos ou sequências de bytes com dimensão fixa ou variável

## □ *Duração e tipo de interacção*

- A comunicação entre interlocutores pode ser menos frequente e/ou de duração mais curta,
  - caso em que é mais “natural” a utilização do protocolo UDP, que permite o envio de “mensagens independentes” para um ou mais destinatários (e.g., como nos SMS dos telefones móveis)

*(continua)*

# *Interacções e mensagens...(3)*

(continuação)

- *ou mais longa,*
  - caso em que é mais “natural” a utilização do protocolo TCP, que estabelece uma conexão entre dois interlocutores (tal como uma chamada telefónica)

## □ Garantias (UDP)

- “Faz-se o que se pode” (*best effort*) – a rede tenta entregar a mensagem, mas se não conseguir, não avisa da falha; a mesma mensagem pode ser entregue duas (ou mais) vezes; duas mensagens A e B enviadas em sequência podem chegar ao destino por ordem inversa...
- Se o receptor não conseguir processar ao ritmo que recebe, perde mensagens...

(continua)

# *Interacções e mensagens...(3)*

(continuação)

## □ Garantias (TCP)

- Se a rede não conseguir entregar uma mensagem, o emissor repete a mensagem algumas vezes, até que desiste e **assinala um erro**; duas mensagens A e B enviadas em sequência (pelo mesmo emissor) chegam ao “processo” receptor pela mesma ordem (mesmo que tenham chegado ao computador por ordem inversa).
- Se o receptor não conseguir processar ao ritmo que recebe, força o emissor a diminuir a velocidade...

(continua)

# *Interacções e mensagens...(3)*

## □ Que sentido faz o UDP???

- Se as *falhas forem raras, faz sentido*. O facto do protocolo não tratar dos erros, não impede a sua utilização, apenas “dita” que os erros terão de ser tratados na aplicação.
  - Exemplo: o protocolo DNS é usado para permitir interagir com computadores usando nomes em vez de endereços. Quando um utilizador usa um browser para aceder a [www.di.fct.unl.pt](http://www.di.fct.unl.pt) (em vez de usar o endereço) a aplicação envia um pacote UDP para um servidor DNS “perguntando-lhe” qual o endereço IP correspondente (no caso, o endereço IP do servidor web [www.di.fct.unl.pt](http://www.di.fct.unl.pt) é 193.136.126.43).
  - Se não receber resposta (porque o pedido ou a resposta se perderam) tenta de novo, e ao fim de 3 tentativas, faz a “pergunta” a outro servidor DNS...

# *Servidores e clientes...(3)*

## □ *Servidores e clientes*

- O exemplo anterior ilustra o conceito de servidor que aqui queremos deixar: uma aplicação que presta um serviço “escutando” num porto a chegada de mensagens com pedidos, processa esses pedidos, e devolve as respostas aos “remetentes” – que se designam clientes (do serviço)

### *Cliente*

```
pac.Destino={srvIP,UDP,srvPort};  
pac.Mensagem="www.di.fct.unl.pt";  
pac.Remetente={myIP, UDP, myPort};  
if (send(pac) < 0)  
    tratarErro();  
...  
...
```

### *Servidor*

```
while (1) {  
    waitForPac(pak);  
    TrataPedido();  
    pac.Destino=pak.Remetente;  
    pac.Mensagem="193.136.122.43";  
    if (send(pac) < 0) ... ;  
}
```

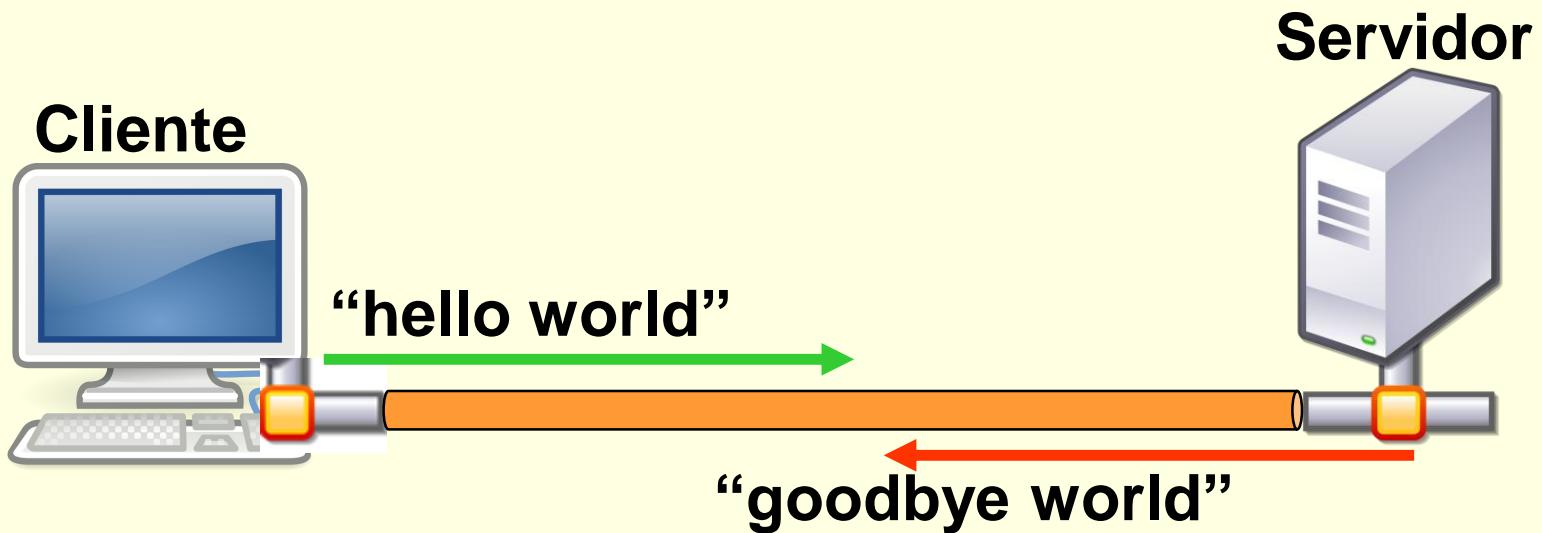
# *A API de sockets (1)*

## □ *Socket*

- Abstracção que permite fazer operações de envio/recepção de dados numa rede de computadores de forma **análoga** às operações de escrita/leitura sobre ficheiros:

Ficheiros	Sockets
<code>creat()</code>	<code>socket()</code>
<code>read()</code>	<code>read()/recv()/. . .</code>
<code>write()</code>	<code>write()/send()/. . .</code>
<code>close()</code>	<code>close()</code>
N . A.	<code>bind()</code>

# Demo: Cliente/Servidor UDP



# *Cliente/Servidor UDP* (1)

## □ *Cliente*

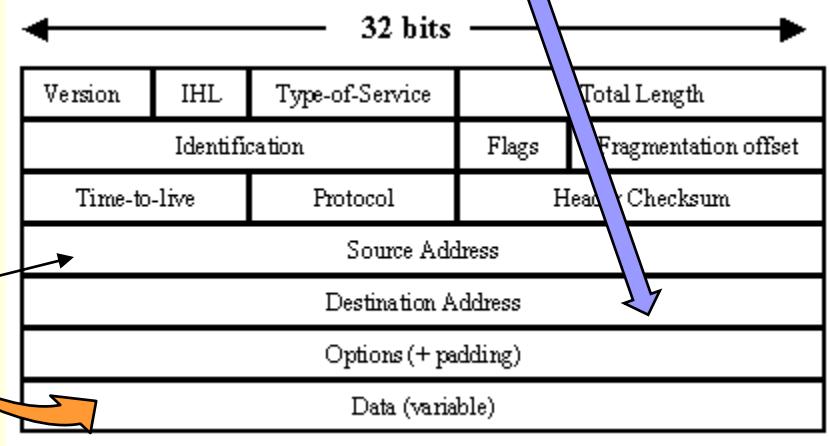
```
int main(int argc, char *argv[]) {  
  
    struct sockaddr_in addrSnd, addrRcv;  
    int sd = UDP_Open(20000);  
    int rc = UDP_FillSockAddr(&addrSnd, "localhost", 10000);  
  
    char message[BUFFER_SIZE]; sprintf(message, "hello world");  
  
    rc = UDP_Write(sd, &addrSnd, message, BUFFER_SIZE);  
    if (rc > 0) {  
        bzero(message, 80);  
        int rc = UDP_Read(sd, &addrRcv, message, BUFFER_SIZE);  
        printf("From server: %s\n", message);  
    }  
    return 0;  
}
```

# Cliente: preenchendo o pacote IP

UDP\_open(20000); UDP\_FillSockAddr(..., "fso-vm64", 10000);

Porto origem: 20000	Porto destino: 10000
Comprimento: 11	Checksum: 0x1234
hello world	

“Convertido” no seu correspondente endereço IP



Obtido da “interface de rede” por onde sai o pacote IP

# *Cliente/Servidor UDP (2)*

## □ Servidor

```
int main(int argc, char *argv[]) {  
  
    int sd = UDP_Open(10000); assert(sd > -1);  
  
    while (1) {  
        struct sockaddr_in addr;  
        char message[BUFFER_SIZE]; bzero(message, BUFFER_SIZE);  
  
        int rc = UDP_Read(sd, &addr, message, BUFFER_SIZE);  
        printf("From client: %s\n", message);  
        if (rc > 0) {  
            char reply[BUFFER_SIZE]; sprintf(reply, "goodbye world");  
            rc = UDP_Write(sd, &addr, reply, BUFFER_SIZE);  
        }  
    }  
    return 0;  
}
```

# *Abstracções em Comunicações (1)*

## □ Troca de mensagens

- É a “mais natural”, e inspirada nas outras áreas das comunicações – correios, rádio, telefone, etc. Assim, tal como no telefone, temos “comunicação ponto-a-ponto”, tal como na rádio temos “comunicação por difusão” (broadcast) datagramas (inspirado nos telegramas), ...

## □ Execução remota de procedimentos (RPC)

- Trata-se de uma forma de comunicação inspirada na execução de funções: o cliente indica qual a função a invocar e quais os parâmetros. Tal informação é transmitida ao servidor, que executa a função e retorna (ao cliente) os resultados...

# *Abstracções em Comunicações (2)*

## □ Memória partilhada distribuída

- *De uma forma simplificada, o espaço de endereçamento de um processo inclui memória que está fisicamente noutras computadores – o que faz que quando se accede a variáveis ou executa funções isso possa estar a acontecer em múltiplos computadores na rede. A ideia é ser transparente para o programador, mas a prática demonstra que isso penaliza demasiado o desempenho, ao ponto de não ser útil em “casos reais”...*

## □ Outras...