Introdução as Redes de Comunicação - Aula 1

Prof Leandro Dihl

Sumário

- Objectivos, funcionamento e avaliação da componente prática.
- Introdução à tecnologia TCP/IP
- Noção de aplicações distribuída
- Conceito de pilha protocolar e encapsulamento
- Noções de endereçamento IP, máscara de subrede e porto
- Referência a alguns exemplos de protocolos do nível de aplicação
- Exercício 1 da Lista de Exercicios

Objectivos, funcionamento e avaliação da componente prática.

Componente prática

- Sockets BSD e **Sockets Microsoft Windows**
- Desenvolvimento de aplicações distribuídas para redes TCP/IP em ambiente Microsoft Windows
- Configuração e teste de conectividade em redes locais elementares

Objectivos, funcionamento e avaliação da componente prática.

Componente prática

• Exposição de matéria apoiada em demonstrações práticas

• Desenvolvimento, teste e depuração de aplicações distribuídas em ambiente laboratorial

Avaliação da componente prática.

Testes

- Teste UDP 07/11
- Teste TCP 19/12

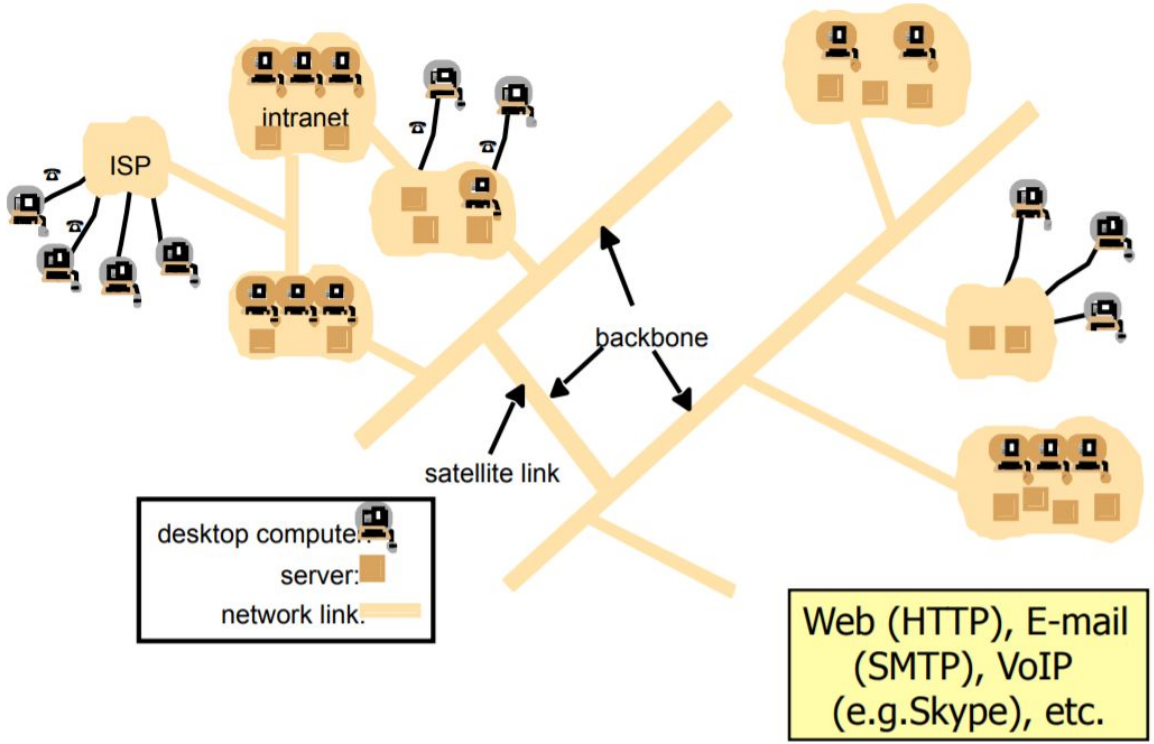
Bibliografia.

- Fernando Boavida, Mário Bernardes, TCP/IP – TEORIA E PRÁTICA, FCA, 2012
- James Kurose, Keith Ross, COMPUTER NETWORK: A TOP-DOWN APPROACH FEATURING THE INTERNET, 6th edition, Pearson/Addison Wesley
- W. Richard Stevens, Bill Fenner, Andrew M. Rudoff, UNIX NETWORK PROGRAMMING, VOLUME 1: THE SOCKETS NETWORKING API, 3rd Edition, Addison-Wesley Professional Computing Series, 2003
- Acetatos e recursos coligidos
- Edmundo Monteiro, Fernando Boavida, ENGENHARIA DE REDES INFORMÁTICAS, 10ª edição, FCA
- W. Richard Stevens, TCP/IP ILLUSTRATED – VOLUME 1 (THE PROTOCOLS), Adison-Wesley, 1994
- Andrew S. Tanenbaum, COMPUTER NETWORKS, 4th edition, Prentice Hall, 2002
- Fred Halsall, Data communication, computer networks and open systems, Adison-Wesley, 1996

Sistemas Distribuídos

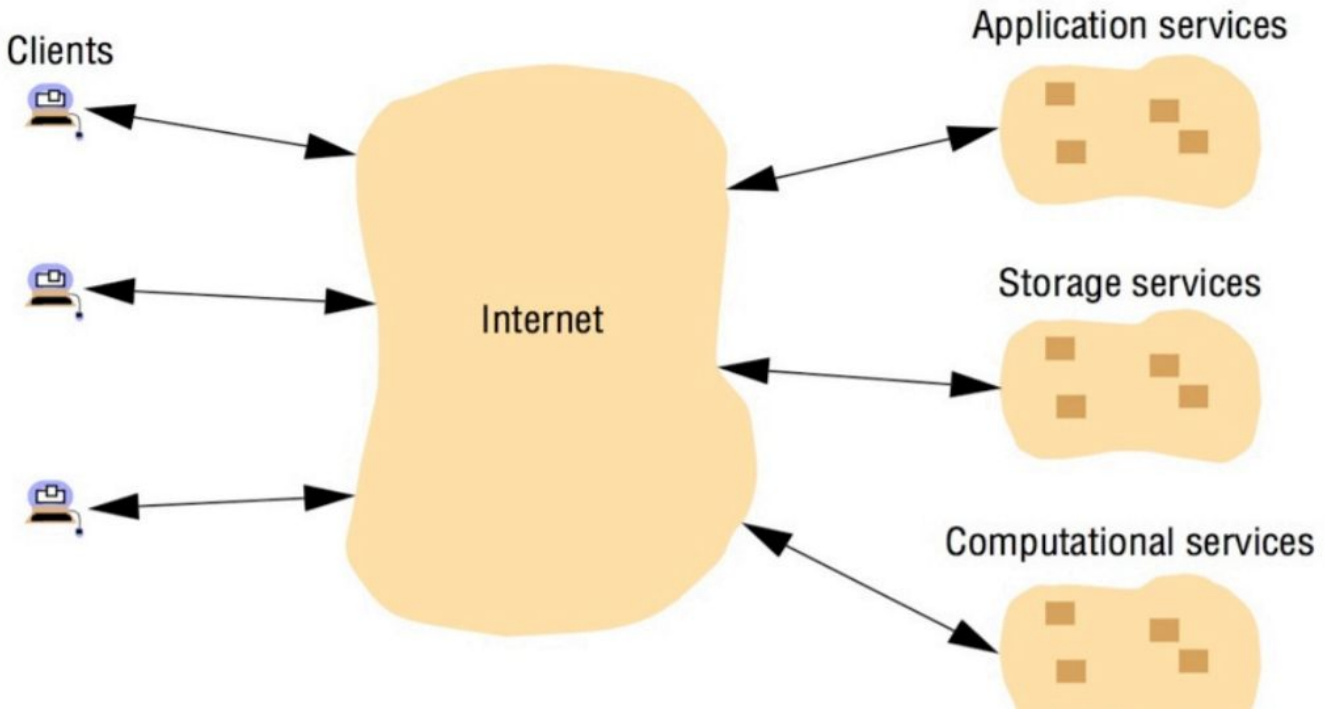
Um sistema distribuído é um conjunto de componentes hardware e software interligados através de uma infraestrutura de comunicações, que cooperam e se coordenam entre si apenas pela troca de mensagens, para execução de aplicações distribuídas

Sistemas Distribuídos



Sistemas Distribuídos

Cloud Computing



Sistemas Distribuídos - Vantagens

Acesso generalizado sem restrições de localização

Acessibilidade ubíqua (suporte para utilizadores fixos, móveis)

Partilha dos recursos distribuídos pelos diferentes utilizadores

- Exemplos: impressores, ficheiros

Distribuição da carga – melhoria do desempenho

Tolerância a falhas – melhoria da disponibilidade

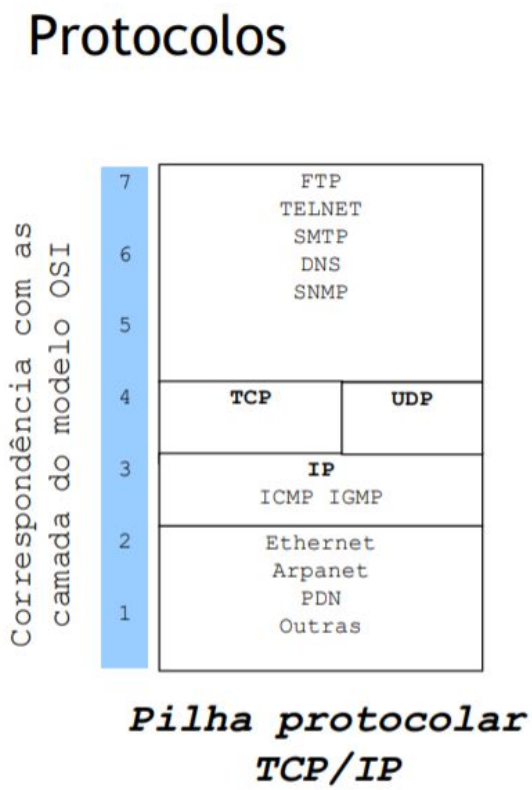
Flexibilidade e adaptabilidade

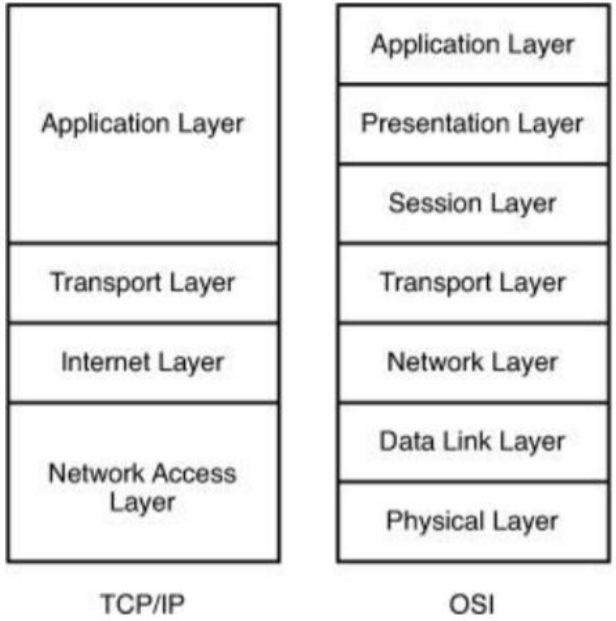
Decomposição de um sistema complexo num conjunto de sistemas mais simples

Pilha Protocolar TCP/IP.

- Na Internet é utilizada a pilha protocolar TCP/IP
- A DARPA (Defense Advanced Research Project Agency) desenvolveu o TCP/IP com o objectivo de interligar as bases militares e os departamentos de pesquisa do governo americano
- Na prática, tornou-se na norma mais utilizada no mundo para sistemas abertos, acabando por dar origem à designada Internet
- Norma de facto
- RFC (Request for Comments)
- Por exemplo: RFC 793, Transmision Control Protocol, Setembro de 1981

Pilha Protocolar TCP/IP.





Application Layer - Exemplos

FTP: File Transfer Protocol = Protocolo de Transferência de Arquivos.

Ele é basicamente um tipo de conexão que permite a troca de arquivos entre dois computadores conectados à internet.

Exemplos:

FileZilla, SmartFTP e gFTP (Linux)

Application Layer - Exemplos

Telnet é um protocolo de rede na Internet ou redes locais para proporcionar uma facilidade de comunicação baseada em texto interativo bidirecional usando uma conexão de terminal virtual. Os dados do usuário são intercalados em banda com informações de controle Telnet em um byte de conexão 8-bit de dados orientado sobre o Transmission Control Protocol (TCP).

Application Layer

Exemplos:

DNS: Os servidores DNS (Domain Name System, ou sistema de nomes de domínios) são os responsáveis por localizar e traduzir para números IP os endereços dos sites que digitamos nos navegadores.

Application Layer

Protocolo de Transferência de Correio Simples (do inglês: Simple Mail Transfer Protocol, abreviado SMTP) é o protocolo padrão de envio de mensagens de correio eletrônico através da Internet entre dois dispositivos computacionais (emissor e receptor), definido na RFC 821.

É um protocolo simples, em texto plano, de somente de envio onde um ou vários destinatários de uma mensagem são especificados sendo, depois, a mensagem transferida, por padrão via porta TCP 25 (ou 465 para conexão criptografada com SSL), podendo usar a porta alternativa 587.

Transport Layer

A camada de transporte, Modelo TCP/IP, é a camada responsável pela transferência de dados entre duas máquinas, independente da aplicação usada e do tipo, topologia ou configuração das redes físicas existentes entre elas. A camada de transporte reúne protocolos de transporte end-to-end entre máquinas, isto é, uma entidade (hardware/software) que utilize os protocolos desta camada só se comunica com a sua entidade destino, sem comunicação com máquinas intermediárias na rede, como pode ocorrer com as camadas inferiores.[1] Dois dos principais protocolos desta camada são o UDP e o TCP.

Transport Layer - UDP Protocol

O User Datagram Protocol (UDP) é um protocolo simples da camada de transporte. Ele é descrito na RFC 768 e permite que a aplicação envie um datagrama encapsulado num pacote IPv4 ou IPv6 a um destino, porém sem qualquer tipo de garantia que o pacote chegue corretamente.

O protocolo UDP não é confiável.

Cada datagrama UDP tem um tamanho e pode ser considerado como um registro indivisível

Também dizemos que o UDP é um serviço sem conexão

O UDP também fornece os serviços de broadcast e multicast, permitindo que um único cliente envie pacotes para vários outros na rede.

Transport Layer - TCP Protocol

- *Orientado à conexão*
- *Handshake*
- *Ponto a ponto* - uma conexão TCP é estabelecida entre dois pontos.
- *Confiabilidade* - O TCP usa várias técnicas para proporcionar uma entrega confiável dos pacotes de dados que, dependendo da aplicação, gera uma grande vantagem que tem em relação ao UDP.
- *Full duplex* - É possível a transferência simultânea em ambas direções (cliente-servidor) durante toda a sessão.
- *Entrega ordenada* - A aplicação faz a entrega ao TCP de blocos de dados com um tamanho arbitrário num fluxo (ou *stream*) de dados, tipicamente em octetos.
- *Controle de fluxo* - O TCP usa o campo janela ou *window* para controlar o fluxo. O receptor, à medida que recebe os dados, envia mensagens ACK (=Acknowledgement), confirmando a recepção de um segmento;
- *Controle de congestionamento*

Exercício 1 - UDP

1. Desenvolva uma aplicação cliente-servidor em que os clientes enviam, via protocolo UDP, uma mensagem de texto, passada como argumento através da linha de comando, para um servidor específico com localização dada pelas constantes SERV_HOST_ADDR (endereço IP) e SERV_UDP_PORT (porto). O servidor deve ir apresentado na saída standard as mensagens que vai recebendo.

Exemplo cliente:

Define variáveis

Inicia Winsock

Cria o socket

Preenche end serv

Envia msg servidor

Obtém o porto

Aguarda resposta

Fecha o socket

Exercício 1 - UDP

```
// INCLUDES E DEFINIÇõES


#include <winsock.h>
#include <stdio.h>

#define SERV_HOST_ADDR "127.0.0.1"
#define SERV_UDP_PORT  6000

#define BUFFERSIZE     4096

void Abort(char *msg);
```

Exercício 1 - UDP

```
int main( int argc , char *argv[] )
{
	int sockfd,msg_len;
	struct sockaddr_in serv_addr;
	char buffer[BUFFERSIZE];
	/*========================= TESTA A SINTAXE =========================*/
	if(argc != 2){
		fprintf(stderr,"Sintaxe: %s frase_a_enviar\n",argv[0]);
		exit(EXIT_FAILURE);
	}

	/*=============== CRIA SOCKET PARA ENVIO/RECEPCAO DE DATAGRAMAS ==============*/

	sockfd = socket( PF_INET , SOCK_DGRAM , 0 );
	if(sockfd < 0)
		Abort("Impossibilidade de criar socket");
```

SOCKADDR_IN structure (winsock.h)

The sockaddr structure varies depending on the protocol selected. Except for the sin*_family parameter, sockaddr contents are expressed in network byte order.

https://docs.microsoft.com/en-us/windows/win32/api/winsock/ns-winsock-sockaddr_in

Syntax

C++

```
typedef struct sockaddr_in {
  short          sin_family;
  u_short        sin_port;
  struct in_addr sin_addr;
  char           sin_zero[8];
} SOCKADDR_IN, *PSOCKADDR_IN, *LPSOCKADDR_IN;
```

socket function

The socket function creates a socket that is bound to a specific transport service provider.

https://docs.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-socket

Syntax

C++

```
SOCKET WSAAPI socket(
  int af,
  int type,
  int protocol
);
```

socket function: Parameters

- **af -** The address family specification.

AF_INET 2	**AF_INET6 23**	**AF_NETBIOS 17**

- **type -** The type specification for the new socket.

SOCK_STREAM 1	**SOCK_DGRAM 2**	**SOCK_RAW 3**

- **protocol -** The protocol to be used. The possible options for the protocol parameter are specific to the address family and socket type specified.

IPPROTO_TCP 6	**IPPROTO_UDP 17**	

Exercício 1 - UDP

```
/*================= PREENCHE ENDERECO DO SERVIDOR ====================*/

        bzero( (char*)&serv_addr , sizeof(serv_addr) ); /*Coloca a zero todos os bytes*/
        serv_addr.sin_family = AF_INET; /*Address Family: Internet*/

        serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR); /*IP no formato "dotted decimal" => 32 bits*/
        serv_addr.sin_port = htons(SERV_UDP_PORT); /*Host TO Netowork Short*/


        /*====================== ENVIA MENSAGEM AO SERVIDOR ==================*/

        msg_len = strlen(argv[1]);

        if(sendto( sockfd , argv[1] , msg_len , 0 , (struct sockaddr*)&serv_addr , sizeof(serv_addr) ) != msg_len)
                Abort("SO nao conseguiu aceitar o datagram");


        printf("<CLI1>Mensagem enviada ... a entrega nao e' confirmada.\n");
```

sento function

The sendto function sends data to a specific destination.

https://docs.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-sendto

Syntax

C++

```
int WSAAPI sendto(
  SOCKET         s,
  const char     *buf,
  int            len,
  int            flags,
  const sockaddr *to,
  int            tolen
);
```

sento function: Parameters

- **s -** A descriptor identifying a (possibly connected) socket.

- **buf -** A pointer to a buffer containing the data to be transmitted.

- **len -** The length, in bytes, of the data pointed to by the buf parameter.

sento function: Parameters

- **flags - A set of flags that specify the way in which the call is made.**

- **to -** An optional pointer to a sockaddr structure that contains the address of the target socket.

- **tolen -** The size, in bytes, of the address pointed to by the to parameter.

sento function: Return Value

- **If no error occurs, sendto returns the total number of bytes sent, which can be less than the number indicated by len.**

WSAENETDOWN	WSAEFAULT	WSAESHUTDOWN

Exercício 1 - UDP

```
        /*========================= FECHA O SOCKET ============================*/

        close(sockfd);

        printf("\n");
        exit(EXIT_SUCCESS);
}
/*_____________________________________ Abort_____________________________________
  Mostra a mensagem de erro associada ao ultimo erro no sistema operativo
  e termina a aplicacao com "exit status" a 1 (constante EXIT_FAILURE)
_____________________________________________________________________________*/

void Abort(char *msg){

        fprintf(stderr,"<CLI1>Erro fatal: <%s>\n",msg);
        perror("Erro do sistema");
        exit(EXIT_FAILURE);

}
```

Exemplo servidor:

Define variáveis

Inicia Winsock

Cria o socket

Associa socket ao endereço de escuta

Loop (recebe msgs)

Exercício 1 - UDP

```
// INCLUDES E DEFINIÇõES

/*============================== Servidor basico UDP ==========================
  Este servidor UDP destina-se a mostrar os conteudos dos datagramas recebidos.
  O porto de escuta encontra-se definido pela constante SERV_UDP_PORT.
  Assume-se que as mensagens recebida sao cadeias de caracteres (ou seja,
  "strings").
=============================================================================*/

#include <stdio.h>
#include <winsock.h>

#define SERV_UDP_PORT 6000
#define BUFFERSIZE     4096

void Abort(char *msg);
```

Exercício 1 - UDP

```
int main( int argc , char *argv[] )
{
	SOCKET sockfd;
	int iResult, nbytes;
	struct sockaddr_in serv_addr;
	char buffer[BUFFERSIZE];
	WSADATA wsaData;
	/*=============== INICIA OS WINSOCKS ==============*/

	iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
	if (iResult != 0) {
		printf("WSAStartup failed: %d\n", iResult);
		getchar();
		exit(1);
	}
	/*============ CRIA O SOCKET PARA RECEPCAO/ENVIO DE DATAGRAMAS UDP ============*/

	if((sockfd = socket( PF_INET , SOCK_DGRAM , 0)) == INVALID_SOCKET)
		Abort("Impossibilidade de abrir socket");
```

WSADATA wsaData;

The WSADATA structure contains information about the Windows Sockets implementation.

https://docs.microsoft.com/en-us/windows/win32/api/winsock/ns-winsock-wsadata

WSAStartup function

The WSAStartup function initiates use of the Winsock DLL by a process.

https://docs.microsoft.com/pt-br/windows/win32/api/winsock/nf-winsock-wsastartup

Syntax

C++

```
int WSAStartup(
  WORD      wVersionRequired,
  LPWSADATA lpWSAData
);
```

WSAStartup function: Parameters

- **wVersionRequired -** A descriptor identifying a (possibly connected) socket.

- **lpWSAData -** A pointer to the WSADATA data structure that is to receive details of the Windows Sockets implementation.

Exercício 1 - UDP

```
/*=============== ASSOCIA O SOCKET AO  ENDERECO DE ESCUTA ===============*/

/*Define que pretende receber datagramas vindos de qualquer interface de
rede, no porto pretendido*/

memset( (char*)&serv_addr , 0, sizeof(serv_addr) );
serv_addr.sin_family = AF_INET; /*Address Family: Internet*/
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);  /*Host TO Network Long*/
serv_addr.sin_port = htons(SERV_UDP_PORT);  /*Host TO Network Short*/

/*Associa o socket ao porto pretendido*/

if(bind( sockfd , (struct sockaddr *)&serv_addr , sizeof(serv_addr)) == SOCKET_ERROR)
        Abort("Impossibilidade de registar-se para escuta");
```

bind function

The bind function associates a local address with a socket.

https://docs.microsoft.com/pt-br/windows/win32/api/winsock/nf-winsock-bind

Syntax

C++

```
int bind(
  SOCKET         s,
  const sockaddr *addr,
  int            namelen
);
```

bind function: Parameters

- **s -** A descriptor identifying an unbound socket.
- **addr -** A pointer to a sockaddr structure of the local address to assign to the bound socket.
- **namelen -** The length, in bytes, of the value pointed to by the name parameter.

- **Return value**

  If no error occurs, bind returns zero. Otherwise, it returns SOCKET_ERROR,

Exercício 1 - UDP

```
	/*================ PASSA A ATENDER CLIENTES INTERACTIVAMENTE =============*/

	while(1){

		fprintf(stderr,"<SER1>Esperando datagram...\n");

		nbytes=recvfrom(sockfd , buffer , sizeof(buffer) , 0 , NULL , NULL);

		if(nbytes == SOCKET_ERROR)
			Abort("Erro na recepcao de datagrams");

		buffer[nbytes]='\0'; /*Termina a cadeia de caracteres recebidos com '\0'*/

		printf("\n<SER1>Mensagem recebida {%s}\n",buffer);

	}

}
```

recvfrom function

The recvfrom function receives a datagram and stores the source address

https://docs.microsoft.com/pt-br/windows/win32/api/winsock/nf-winsock-recvfrom

Syntax

C++

```
int recvfrom(
  SOCKET   s,
  char     *buf,
  int      len,
  int      flags,
  sockaddr *from,
  int      *fromlen
);
```

recvfrom function: Parameters

- **s -** A descriptor identifying a bound socket.

- **buf -** A buffer for incoming data.

- **len -** The length, in bytes, of the buffer pointed to by the buf parameter.

recvfrom function: Parameters

- **flags -** A set of options that modify the behavior of the function call beyond the options specified for the associated socket. See the Remarks below for more details.
- **from -** An optional pointer to a buffer in a sockaddr structure that will hold the source address upon return.
- **fromlen -** An optional pointer to the size, in bytes, of the buffer pointed to by the from parameter.

Return value

- If no error occurs, recvfrom returns zero. Otherwise, it returns SOCKET_ERROR.

Exercício 1 - UDP

```
/*_____ Abort_____
  Mostra uma mensagem de erro e o código associado ao ultimo erro com Winsocks.
  Termina a aplicacao com "exit status" a 1 (constante EXIT_FAILURE)
_____*/

void Abort(char *msg)
{

        fprintf(stderr,"<SERV>Erro fatal: <%s> (%d)\n",msg, WSAGetLastError());
        exit(EXIT_FAILURE);

}
```

Exercício 2

Altere a aplicação anterior de modo a que o servidor reenvie as mensagens recebidas aos respectivos clientes. Estes devem aguardar pelas respostas e apresentá-las na saída standard.

Servidor:

- sendto

Cliente:

- recvfrom