

Comunicação com Sockets e UDP no C#

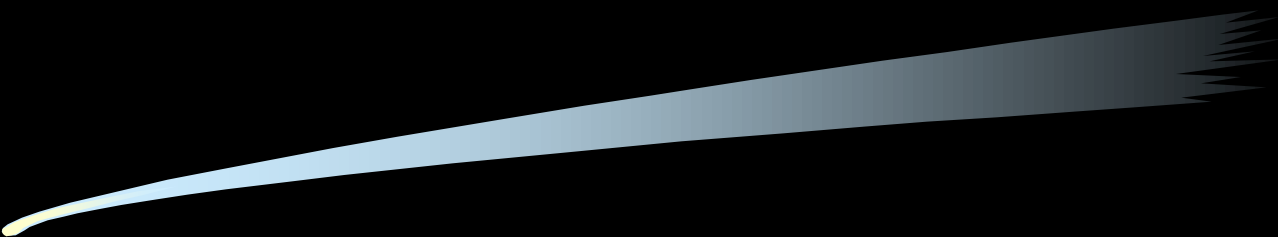


Sistemas Distribuídos

Prof. Dr. Sérgio Carlos Portari Jr

O que são sockets?

- Podemos entender que sockets são uma abstração para que possamos comunicar processos na rede.
- Estes processos podem estar no mesmo computador ou em computadores diferentes conectados a uma rede e enviarão e receberão mensagens durante os seus processamentos. A porta para a troca de mensagens entre esses processos são os sockets.



O que são sockets?

- RESUMINDO:
- O socket é um elemento de software que possui uma interface de rede para troca de mensagens através de protocolos.

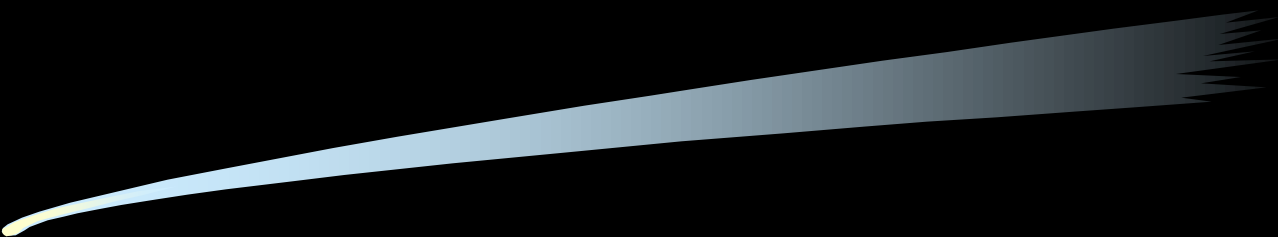
Quais protocolos são utilizados?

- Os principais protocolos utilizados para comunicação de sockets no .Net são TCP e UDP.

O protocolo TCP é orientado à conexão e considerado confiável. Utilizamos este protocolo quando os dados a serem transmitidos precisam de confirmação do seu recebimento e que eles foram transmitidos sem perda de informação.

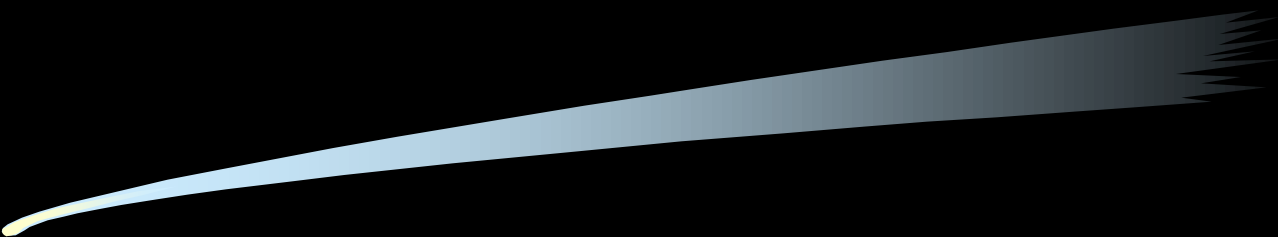
Quais protocolos são utilizados?

- O protocolo UDP não é orientado a conexão. Este protocolo não garante o recebimento da mensagem e nem a integridade dos dados, mas por outro lado é mais rápido do que o TCP.
- Em software onde a garantia da entrega das mensagens e a integridade dos dados não é crucial, mas a velocidade de comunicação é importante o UDP pode ser utilizado.



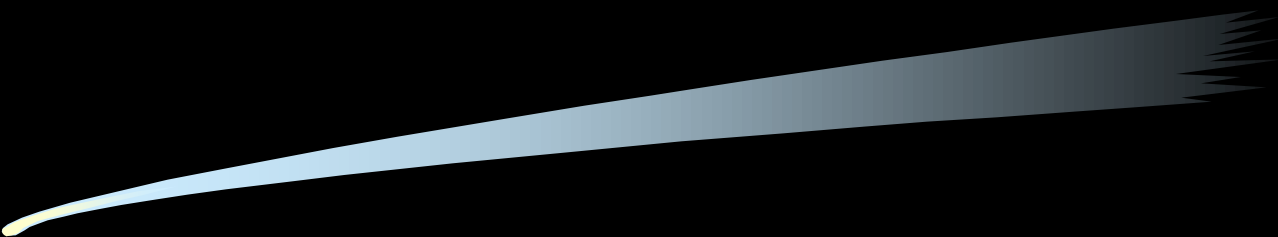
Sockets no .Net:

- O .Net possui dois namespaces (pacotes) para trabalhar com comunicação de rede e sockets.
- O primeiro namespace utilizado é o **System.Net** que provê acesso as funções de rede do Windows.
- Para termos acesso a interface de sockets do Windows utilizamos um outro namespace chamado **System.Net.Sockets**.



Sockets no .Net:

- As principais classes utilizadas neste namespaces são:
 - **IPAddress**: que é utilizada para representar um único endereço IP;
 - **EndPoint**: representa uma combinação entre o endereço IP e uma porta;
 - **Socket**: representa uma interface de um socket no Windows;



Exemplo Prático

- Vamos criar um protótipo de chat que utilizará sockets em C# para realizar a comunicação entre um cliente e um servidor.
- Utilizaremos inicialmente o modo mais simples, em Terminal

Módulo Cliente

(envia mensagem a um servidor)

Crie um projeto console C#.

Vamos adicionar os pacotes (Namespaces) para acesso ao Socket:

- `using System.Net;`
- `using System.Net.Sockets;`

Módulo Cliente

(envia mensagem a um servidor)

- Iremos na sequência criar dentro da classe Program padrão, uma classe estática (que pode ter seus métodos acessados sem ser instanciados) de UdpClient (cliente UDP):

```
private static UdpClient cliente;
```

Módulo Cliente

(envia mensagem a um servidor)

- Dentro do Main() iremos então criar uma instância do objeto client no ip local na porta 9999:

```
cliente = new UdpClient("127.0.0.1", 9999);  
//vamos escrever uma mensagem na tela para  
//sabermos que este software é o cliente  
Console.WriteLine("Terminal Cliente");
```

Módulo Cliente

(envia mensagem a um servidor)

- Criaremos um Loop Infinito que irá ficar recebendo uma string pelo teclado do usuário e enviando ao módulo servidor através de um método de envio (Enviar) que iremos escrever na sequência:

Módulo Cliente

(envia mensagem a um servidor)

```
while (true)
{
    Console.Write(">> ");
    String mensagem = Console.ReadLine();
    Enviar(mensagem);
}
```

Módulo Cliente

(envia mensagem a um servidor)

- O método enviar que foi acionado irá passar a mensagem digitada pelo usuário para o servidor. Crie-o depois de encerrar o Main()

```
private static void Enviar(string mensagem)
{
    byte[] dados = Encoding.ASCII.GetBytes(mensagem);
    cliente.Send(dados, dados.Length)
}
```

Módulo Servidor

(recebe mensagem de um cliente)

- Lembre-se que precisamos adicionar os pacotes (namespaces) do Socket:

```
using System.Net;
```

```
using System.Net.Sockets;
```

Módulo Servidor

(recebe mensagem de um cliente)

- E também, dentro da classe Program, criaremos uma classe estática do UdpClient

```
private static UdpClient servidor;
```

Módulo Servidor

(recebe mensagem de um cliente)

- No Main(), instanciaremos o UdpClient na mesma porta do mesmo IP que colocamos no módulo cliente e escreveremos na tela que estamos no módulo servidor:

```
servidor = new UdpClient(9999);
```

```
Console.WriteLine("Terminal Servidor\n>> ");
```

Módulo Servidor

(recebe mensagem de um cliente)

- Criaremos um Loop infinito que ficará “escutando” a porta 9999 em busca de mensagens, através de um método estático que ficará escrito após terminarmos o Main(), chamado Receptor:

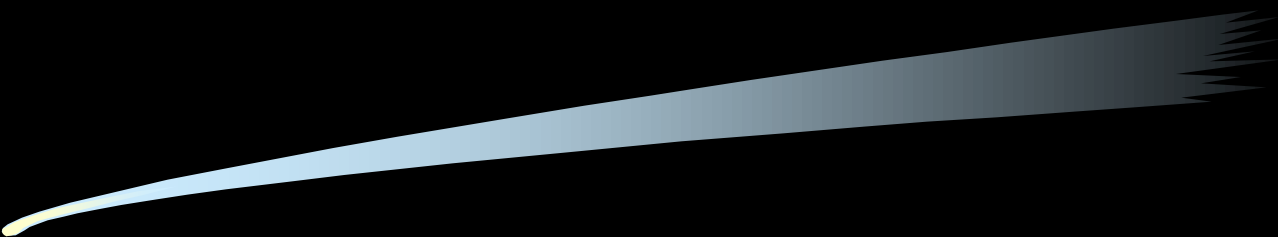
```
while (true)
{
    Receptor();
}
```

Módulo Servidor

(recebe mensagem de um cliente)

- Então vamos ao método estático receptor:

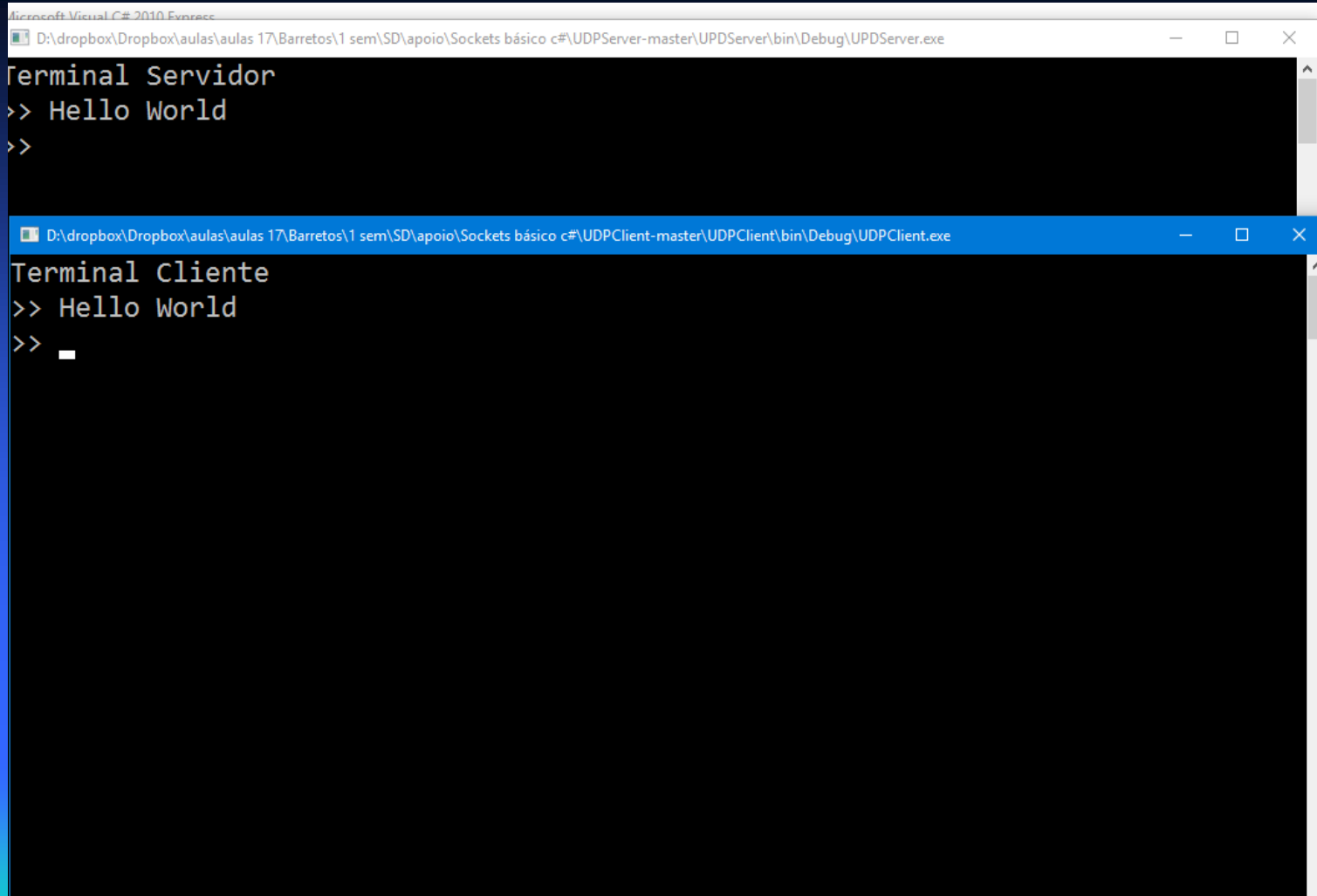
```
private static void Receptor()
{
    IPEndPoint sender = new IPEndPoint(IPAddress.Any, 9999);
    byte[] dados = new byte[1024];
    dados = servidor.Receive(ref sender);
    String info = Encoding.ASCII.GetString(dados, 0, dados.Length);
    Console.WriteLine(info);
    Console.Write(">> ");
}
```



Vamos testar...

- Execute os dois módulos. Poderemos escrever mensagens no módulo cliente e serão recebidos no módulo servidor.

TESTE EXECUÇÃO



The image shows two terminal windows from Visual Studio Code. The top window, titled 'Terminal Servidor', shows the execution of 'UPDServer.exe'. It displays 'Hello World' being received twice. The bottom window, titled 'Terminal Cliente', shows the execution of 'UDPClient.exe'. It displays 'Hello World' being sent once, followed by a cursor on a new line.

```
Microsoft Visual C# 2010 Express
D:\dropbox\Dropbox\aulas\aulas 17\Barretos\1 sem\SD\apoio\Sockets básico c#\UDPServer-master\UDPServer\bin\Debug\UPDServer.exe

Terminal Servidor
>> Hello World
>>

D:\dropbox\Dropbox\aulas\aulas 17\Barretos\1 sem\SD\apoio\Sockets básico c#\UDPClient-master\UDPClient\bin\Debug\UDPClient.exe

Terminal Cliente
>> Hello World
>> _
```