Hướng Dẫn Ví Dụ Trên Chương Trình Thực Tế

Chương 2

Lập trình TCP Socket

Nội dung

1.Mô hình socket hướng kết nối

2. Chương trình Client đơn giản

3. Xử lý một số vấn đề trong lập trình hướng kết nối

4. Sử dụng stream với TCP

57

Địa chỉ IP

- Tất cả các máy tính với địa chỉ dành riêng phải kết nối với ít nhất 1 máy tính hoặc 1 router với địa chỉ dùng chung để truy cập Internet
- Địa chỉ IP của một máy tính có thể thay đổi → vai trò DHCP server
- Một địa chỉ duy nhất không thay đổi gắn với card mạng là địa chỉ MAC (còn gọi là địa chỉ phần cứng)

58

Network stack

- Quá trình lưu thông trên mạng của các tín hiệu là cực kỳ phức tạp, nếu không có khái niệm đóng gói (encapsulation) thì người lập trình sẽ phải chú ý nhiều những chi tiết nhỏ
- Người lập trình chỉ cần tập trung vào điều gì xảy ra ở tầng cao trong OSI

59

Network stack cổ điển

Số thứ tự	Tên tầng	Giao thức
7	Application	FTP
6	Presentation	XNS
5	Session	RPC
4	Transport	TCP
3	Network	IP
2	Data link	Ethernet frames
1	Physical	Điện áp

60

Network stack hiện đại

Số thứ tự	Tên tầng	Giao thức
4	Structured Information	SOAP
3	Message	HTTP
2	Stream	TCP
1	Packet	IP

61

Network stack

- Không cần quan tâm thông tin được “lan truyền như thế nào”, mà chỉ quan tâm “gửi cái gì”
- Chúng ta không khảo sát các giao thức ở tầng vật lý

62

Ports

- Mỗi máy tính có thể có nhiều ứng dụng mạng chạy đồng thời
- Dữ liệu phải đính kèm thông tin cho biết ứng dụng nào dùng nó → port number
- Ví dụ: 80 cho ứng dụng Web, 110 cho ứng dụng email
- Thông tin port được chứa trong header của TCP, UDP packet

63

Ports

Số thứ tự port	Giao thức
20	FTP (data)
21	FTP (control)
25	SMTP (email, outgoing)
53	DNS (domain names)
80	HTTP (Web)
110	POP3 (email, incoming)
119	NNTP (news)
143	IMAP (email, incoming)

64

Socket là gì?

- Lập trình mức socket là nền tảng của lập trình mạng
- Socket là một đối tượng thể hiện điểm truy cập mức thấp vào IP stack.
- Socket có thể ở chế độ mở, đóng hoặc một số trạng thái trung gian khác
- Socket có thể gửi, nhận dữ liệu
- Dữ liệu tổng quát được gửi theo từng khối (thường gọi là packet), khoảng vài KB/lần để tăng hiệu suất

65

Mô hình socket hướng kết nối

- Mô hình ứng dụng Client – Server hướng kết nối
- Các thao tác phía server để xây dựng ứng dụng
- Các thao tác phía client để xây dựng ứng dụng
- Quá trình truyền tin giữa client và server
- Đóng kết nối

66

Mô hình ứng dụng Client – Server hướng kết nối

1. Các thao tác để xây dựng ứng dụng client – server hướng kết nối
   - Các thao tác phía server
   - Các thao tác phía client
   - Quá trình truyền nhận dữ liệu
   - Đóng kết nối
2. Mô hình client – server hướng kết nối

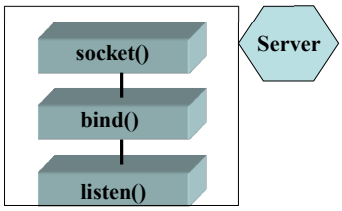
67

Các thao tác để xây dựng ứng dụng client – server hướng kết nối

- Phía server:
  - Tạo ra một Sockets
  - Gắn Sockets đó với một địa chỉ cụ thể (binding)
  - Lắng nghe kết nối tới
  - Chấp nhận kết nối
- Phía Client:
  - Tạo ra một Sockets
  - Kết nối đến Server
- Quá trình truyền nhận dữ liệu
- Đóng kết nối

68

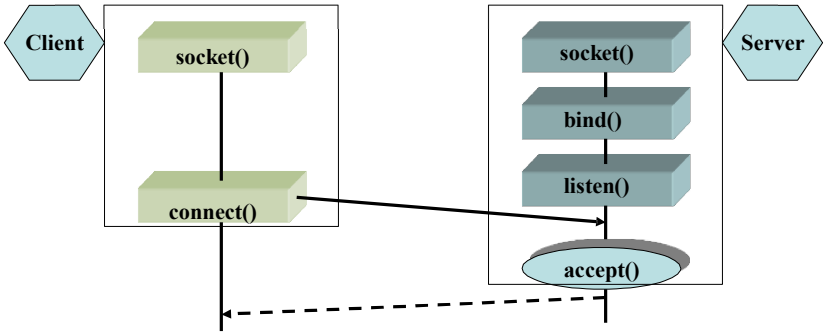
Xây dựng ứng dụng client – server hướng kết nối



- ***socket()***: Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- ***bind():*** Server yêu cầu gán số hiệu port cho socket.
- ***listen()***: Server lắng nghe các ***yêu cầu nối kết*** từ các client trên cổng đã được gán.
- Server sẵn sàng phục vụ client.

69

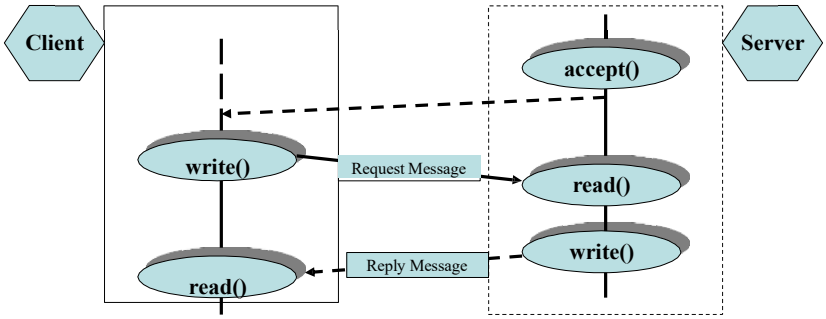
Xây dựng ứng dụng client – server hướng kết nối



- ***socket()***: Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- ***connect()***: Client gởi yêu cầu nối kết đến Server có địa chỉ IP và port xác định.
- ***accept()***: Server chấp nhận nối kết của Client, kênh giao tiếp ảo được hình thành, Client và Server có thể trao đổi thông tin với nhau.

70

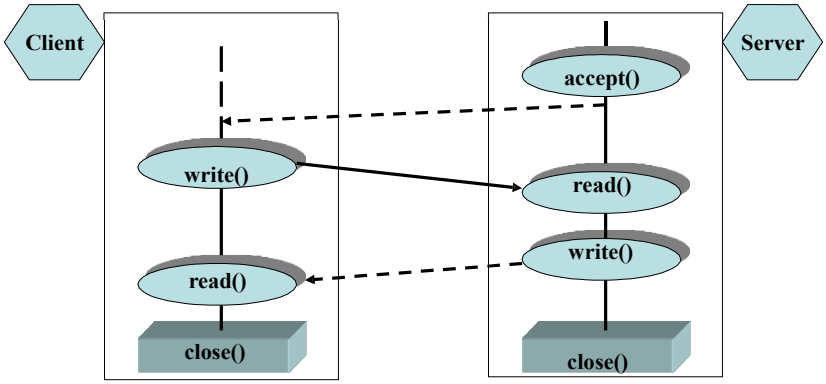
Xây dựng ứng dụng client – server hướng kết nối



- Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh ***read()*** và chờ cho đến khi có thông điệp yêu cầu (Request Message) từ client.
- Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh ***write()***.
- Sau khi gởi yêu cầu bằng lệnh ***write()***, client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh ***read()***.

71

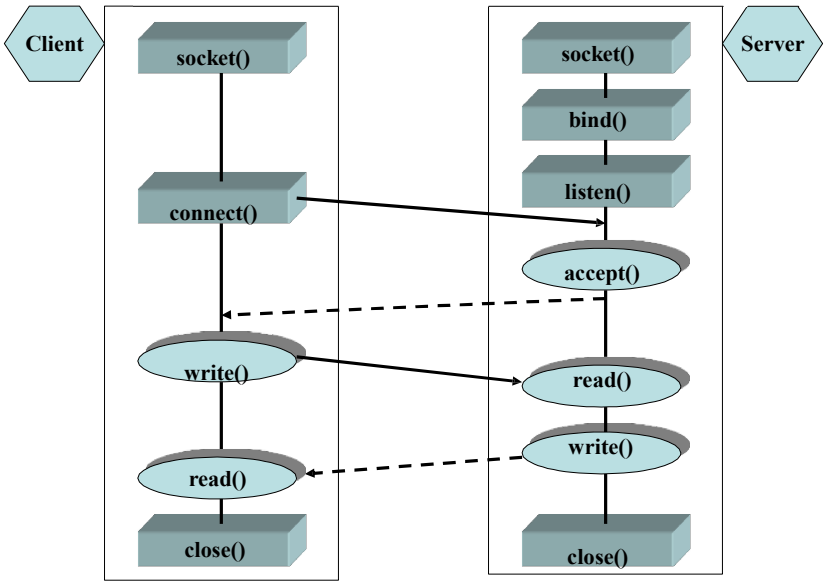
Xây dựng ứng dụng client – server hướng kết nối



- Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).
- Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh ***close()***.

72

Mô hình ứng dụng client – server hướng kết nối



73

Các thao tác phía Server

1. Tạo một socket
2. Gắn socket với một địa chỉ cụ thể (binding)
3. Đặt socket ở trạng thái lắng nghe kết nối tới từ client
4. Chấp nhận kết nối từ client

74

1.Tạo một Socket

- Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

```
//Server Socket
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
```

75

2. Gắn socket với một địa chỉ cụ thể (binding)

- Sau khi một socket đã được tạo, nó phải được gắn với một địa chỉ cụ thể trong hệ thống.
- Phương thức Bind() thực hiện chức năng này
- Cú pháp:
  - Bind(EndPoint *address*)

```
//EndPoint cục bộ
IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 5000);
//Server Socket
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
//Kết nối server với 1 EndPoint
server.Bind(ipep);
```

76

Gắn socket với một địa chỉ cụ thể (binding)

- Ví dụ về gọi hàm Bind()

```
IPHostEntry local =
Dns.GetHostByName(Dns.GetHostName());
IPEndPoint iep = new
IPEndPoint(local.AddressList[0], 5000);
Socket server = new
Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
server.Bind(iep);
```

77

3. Đặt socket ở trạng thái lắng nghe kết nối tới từ client

- Sau khi socket đã được gắn với một địa chỉ cụ thể, sử dụng phương thức Listen() để đặt socket ở trạng thái lắng nghe kết nối từ Client tới.
- Cú pháp:
  - Listen(int ***conn***)
  - Trong đó: **conn** là số kết nối tối đa cho phép đợi ở hàng đợi

78

4. Chấp nhận kết nối từ client

- Sau khi gọi phương thức Listen(), Socket đã bắt đầu sẵn sàng chấp nhận kết nối từ Client.
- Gọi hàm Accept() để chấp nhận các kết nối từ Client.
- Hàm Accept() sẽ trả lại một đối tượng socket mới được sử dụng cho quá trình truyền và nhận dữ liệu sau này

79

Chấp nhận kết nối từ client

- Ví dụ về chấp nhận kết nối: server.Bind(iep);
  server.Listen(10);
  Socket client = server.Accept();

```
//EndPoint cục bộ
IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 5000);
//Server Socket
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
//Kết nối server với 1 EndPoint
server.Bind(ipep);
//Server lắng nghe tối đa 10 kết nối
server.Listen(10);
Console.WriteLine("Dang cho Client ket noi den...");
//Hàm Accept() sẽ block server lại cho đến khi có Client kết nối đến
Socket client = server.Accept();
```

80

Các thao tác phía Client

1. Tạo một socket
2. Kết nối đến server

81

Kết nối đến server

- Dùng hàm Connect() để kết nối đến Server.
- Hàm Connect yêu cầu một đối tượng IPEndPoint của server ở xa mà Client sẽ kết nối tới.

82

Kết nối đến server

- Ví dụ về kết nối đến Server

```
IPAddress host = IPAddress.Parse("127.0.0.1");
IPEndPoint hostep = new IPEndPoint(host, 5000);
Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
client.Connect(hostep);
```

83

Quá trình truyền và nhận dữ liệu

- Sau khi Client kết nối đến Server và đã được chấp nhận, Client và Server có thể bắt đầu quá trình truyền và nhận dữ liệu
- Sử dụng hàm Send() và Receive() để thực hiện các công việc này.

84

Các phiên bản của hàm Send() và Receive()

Method	Description
Receive(byte[] data)	Receives data and places it in the specified byte array
Receive(byte[] data, SocketFlags sf)	Sets socket attributes, receives data, and places it in the specified byte array
Receive(byte[] data, int size, SocketFlags sf)	Sets socket attributes, receives the specified size of data, and places it in the specified byte array
Receive(byte[] data, int offset, int size, SocketFlags sf)	Sets socket attributes, receives the size bytes of data, and stores it at offset offset in the data byte array
Send(byte[] data)	Sends the data specified in the byte array
Send(byte[] data, SocketFlags sf)	Sets socket attributes and sends the data specified in the bytes array
Send(byte[] data, int size, SocketFlags sf)	Sets socket attributes and sends the specified size of data in the specified byte array
Send(byte[] data, int offset, int size, SocketFlags sf)	Sets socket attributes and sends size bytes of data starting at offset offset in the data byte array

85

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

SocketFlags sf có các giá trị sau

Value	Description
DontRoute	Sends data without using the internal routing tables
MaxIOVectorLength	Provides a standard value for the number of WSABUF structures used to send and receive data
None	Uses no flags for this call
OutOfBand	Processes out-of-band data
Partial	Partially sends or receives message
Peek	Only peeks at the incoming message

86

Đóng kết nối

- Sau khi quá trình truyền và nhận dữ liệu kết thúc, Socket cần phải được đóng lại.
- Ta có thể sử dụng hàm Shutdown() để tạm dừng phiên làm việc và dùng hàm Close() để đóng phiên làm việc đó.
- Ví dụ:

```
sock.Shutdown(SocketShutdown.Both);
sock.Close();
```

87

Ngoại lệ Socket

- Một đặc điểm của lập trình socket trong .Net đó là sử dụng *socket exceptions*
- C# sử dụng the try-catch để xử lý các ngoại lệ

88

Ví dụ về chương trình có sử dụng xử lý ngoại lệ

```
IPAddress host = IPAddress.Parse("192.168.1.1");
IPEndPoint hostep = new IPEndPoint(host, 8000);
Socket sock = new
Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
try { sock.Connect(hostep); }
catch (SocketException e)
{
Console.WriteLine("Problem connecting to host");
Console.WriteLine(e.ToString());
 sock.Close();
return;
 }
```

89

Chương trình Client đơn giản

- Ví dụ về xây dựng một chương trình Client đơn giản.

90

Xử lý vấn đề trong lập trình hướng kết nối

- Vấn đề với bộ đệm dữ liệu
- Vấn đề với thông điệp TCP
- Giải quyết vấn đề với thông điệp TCP

91

Vấn đề với bộ đệm dữ liệu

- Trong ví dụ trước chúng ta giả sử rằng quá trình truyền và nhận dữ liệu được thực hiện trong một môi trường có kiểm soát.
- Trong đó tất cả các thông điệp có kích thước nhỏ và biết trước. Dữ liệu được truyền và nhận chỉ là dạng Text

92

Vấn đề với bộ đệm dữ liệu

- Tuy nhiên trong các ứng dụng thực tế chúng ta có thể không biết trước kích thước của dữ liệu đến
- Điều gì sẽ xảy ra khi dữ liệu đến bộ đệm có kích thước khác nhau?
- Điều gì sẽ xảy ra khi dữ liệu đến nhiều hơn so với kích thước của bộ đệm?

93

Sử dụng đúng kích thước bộ đệm

- Dữ liệu được nhận bởi TCP được lưu ở bộ đệm trong của hệ thống
- Mỗi lời gọi phương thức Receive() sẽ loại bỏ các dữ liệu trong bộ đệm mà nó nhận được
- Kích thước của dữ liệu được đọc bởi Receive() được xác định bởi 2 yếu tố
  - The size of the data buffer supplied to the Receive() method
  - The buffer size parameter supplied to the Receive() method

94

Vấn đề với thông điệp TCP

- Một vấn đề quan trọng trong lập trình TCP đó là TCP không phân biệt ranh giới giữa các thông điệp
- Vậy làm thế nào để xử lý được vấn đề này?
- Ví dụ về việc thông điệp không được bảo vệ.
  - Chương trình BadClient
  - Chương trình BadServer

95

Lớp TCPClient

- Để đảm bảo độ tin cậy trong các ứng dụng mạng, người ta còn dùng một giao thức khác, gọi là giao thức có kết nối: TCP (Transport Control Protocol). Trên Internet chủ yếu là dùng loại giao thức này, ví dụ như Telnet, HTTP, SMTP, POP3… Để lập trình theo giao thức TCP, .NET cung cấp hai lớp có tên là TCPClient và TCPListener.

96

Lớp TCPClient: các thành viên

Tên phương thức	Mô tả
TcpClient ()	Tạo một đối tượng **TcpClient**.
TcpClient (IPEndPoint)	Tạo một **TcpClient** và gắn cho nó một EndPoint cục bộ. (Gán địa chỉ máy cục bộ và số hiệu cổng để sử dụng trao đổi thông tin về sau)
TcpClient (RemoteHost: String, RemotePort: Int32)	Tạo một đối tượng **TcpClient** và kết nối đến một máy có địa chỉ và số hiệu cổng được truyền vào. RemoteHost có thể là địa chỉ IP chuẩn hoặc tên máy.

97

Lớp TCPClient: các thành viên

Tên thuộc tính	Mô tả
Available	Cho biết số byte đã nhận về từ mạng và có sẵn để đọc.
Client	Trả về Socket ứng với TCPClient hiện hành.
Connected	Trạng thái cho biết đã kết nối được đến Server hay chưa

98

Lớp TCPClient: các thành viên

Tên phương thức	Mô tả
Close	Giải phóng đối tượng **TcpClient** nhưng không đóng kết nối.
Connect (RemoteHost, Port)	Kết nối đến một máy TCP khác có tên và số hiệu cổng.
GetStream	Trả về NetworkStream để từ đó giúp ta gửi hay nhận dữ liệu. (Thường làm tham số khi tạo StreamReader và StreamWriter) . Khi đó gắn vào StreamReader và StreamWriter rồi ta có thể gửi và nhận dữ liệu thông qua các phương thức ReadIn, WriteIn tương ứng của các lớp này.

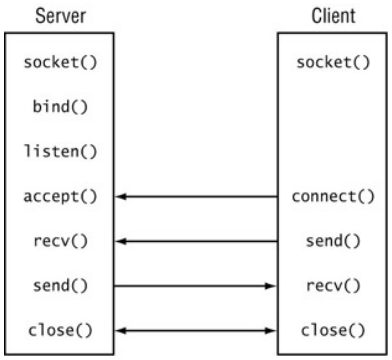
99

Lớp TCPClient: Thực hiện kết nối

- Bước 1: Tạo một đối tượng TCPClient
- Bước 2: Kết nối đến máy chủ (Server) dùng phương thức Connect
- Bước 3: Tạo 2 đối tượng StreamReader (Receive) và StreamWriter (Send) và **"nối"** với GetStream của TCPClient
- Bước 4:
  - Dùng đối tượng StreamWriter.Writeline/Write vừa tạo ở trên để gửi dữ liệu đi.
  - Dùng đối tượng StreamReader.Readline/Read vừa tạo ở trên để đọc dữ liệu về.
- Bước 5: Đóng kết nối.

100

Lớp TCPClient: Thực hiện kết nối



101

Lớp TCPClient

- Nếu muốn gửi/nhận dữ liệu ở mức byte (nhị phân) thì thiết lập NetworkStream (truyền GetStream cho NetworkStream)

102

So sánh TCP & UDP

TCP	UDP
Hoạt động tin cậy	Hoạt động không tin cậy
Phải thiết lập kết nối giữa client & server	Không cần thiết lập kết nối
Dữ liệu gửi không chứa địa chỉ và port của máy nhận	Phải xác định địa chỉ và port của máy nhận trong dữ liệu gửi

103

Xử lý vấn đề với thông điệp TCP

- Một số phương pháp sau có thể dùng để xử lý vấn đề trên
  - Sử dụng thông điệp có kích thước cố định
  - Sử dụng kích thước thông điệp
  - Sử dụng các ký hiệu đánh dấu thông điệp

104

Xử lý vấn đề với thông điệp TCP

- Ví dụ về sử dụng thông điệp có kích thước cố định
- Phân tích:
  - Phải biết trước kích thước của tất cả các thông điệp
  - Tất cả các thông điệp phải có cùng kích thước

105

Ví dụ

❖ Ví dụ về sử dụng kích thước thông điệp

Ví dụ xác định trước kích thước của các thông điệp

Ví dụ các thông điệp phải có cùng kích thước

106

Sử dụng các ký hiệu đánh dấu thông điệp

- Được thực hiện thông qua sử dụng các lớp C# Stream
  - NetworkStream class
  - StreamReader class
  - StreamWriter class

Các lớp C# Stream

- Bởi vì việc điều khiển các thông điệp trong kết nối TCP thường phức tạp
- .NET Framework cung cấp thêm các lớp để trợ giúp cho việc xử lý này.
- Các lớp này bao gồm:
  - NetworkStream class
  - StreamReader class
  - StreamWriter class

Lớp NetworkStream

- Lớp này được sử dụng để cung cấp một giao diện cho Socket
- Tạo ra một đối tượng như sau:

```
Socket newsock = new
Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
NetworkStream ns = new
NetworkStream(newsock);
```


Các thuộc tính của NetworkStream

Property	Description
CanRead	Is true if the NetworkStream supports reading
CanSeek	Is always false for NetworkStreams
CanWrite	Is true if the NetworkStream supports writing
DataAvailable	Is true if there is data available to be read

Một số phương thức của NetworkStream

Method	Description
BeginRead()	Starts an asynchronous NetworkStream read
BeginWrite()	Starts an asynchronous NetworkStream write
Close()	Closes the NetworkStream object
EndRead()	Finishes an asynchronous NetworkStream read
EndWrite()	Finishes an asynchronous NetworkStream write
Equals()	Determines if two NetworkStreams are the same
Read()	Reads data from the NetworkStream
ReadByte()	Reads a single byte of data from the NetworkStream

111

Các lớp StreamReader and StreamWriter

- Các lớp này dùng để đọc và ghi dữ liệu trên luồng vào ra
- Sử dụng các lớp này với NetworkStream cho các thông điệp TCP

112

Hàm tạo

- Hàm tạo được sử dụng phổ biến là:
  - public StreamReader(Stream stream)
  - public StreamWriter(Stream stream)

Một số phương thức của StreamReader

Method	Description
Close()	Closes the StreamReader object
Read()	Reads one or more bytes of data from the StreamReader
ReadBlock()	Reads a group of bytes from the StreamReader stream and places it in a specified buffer location
ReadLine()	Reads data from the StreamReader object up to and including the first line feed character
ReadToEnd()	Reads the data up to the end of the stream
ToString()	Creates a string representation of the StreamReader object

Một số phương thức của StreamWrtiter

Method	Description
Flush()	Sends all StreamWriter buffer data to the underlying stream
Write()	Sends one or more bytes of data to the underlying stream
WriteLine()	Sends the specified data plus a line feed character to the underlying stream

115

Ví dụ về StreamReader

```
OpenFileDialog ofd = new OpenFileDialog();
        ofd.ShowDialog();
        FileStream fs = new FileStream(ofd.FileName,
FileMode.OpenOrCreate);
        StreamReader sr = new StreamReader(fs);
        int lineCount = 0;

        while (sr.ReadLine() != null)
        {
           lineCount++;
        }
        fs.Close();
MessageBox.Show("Có  " + lineCount + " dòng trong " + ofd.FileName);
```

116

Ví dụ về StreamReader

```
try { // tao instance cua StreamReader de doc mot file. // lenh
using cung duoc su dung de dong StreamReader.
using (StreamReader sr = new StreamReader("textfile.txt"))
{
string line; // doc va hien thi cac dong trong file cho toi // khi
tien toi cuoi file.
while ((line = sr.ReadLine()) != null)
        {
        Console.WriteLine(line);
        }
 } Console.ReadKey();
```


Ví dụ về StreamWriter

```
string[] names = new string[] { “Nguyen Van A", “Tran Van  B"
}; using (StreamWriter sw = new StreamWriter("textfile.txt"))
 {
foreach (string s in names)
      {
      sw.WriteLine(s);
      }
} // doc va hien thi du lieu trong textfile.txt
string line = "";
 using (StreamReader sr = new StreamReader("textfile.txt"))
 { while ((line = sr.ReadLine()) != null)

            { Console.WriteLine(line); }
}
```

Bài tập

1. Viết chương trình minh họa các lớp IPEndpoint, IPAddress, Dns như phần hướng dẫn minh họa bài tập trên lớp
2. Viết chương trình TCP đặt ở hai máy thực hiện công việc sau: Khi một ứng dụng gửi chuỗi "OPEN#<Đường dẫn >" thì ứng dụng trên máy kia sẽ mở file nằm trong phần <đường dẫn>. Khi một ứng dụng gửi chuỗi "**SHUTDOWN**" , thì ứng dụng kia sẽ tắt máy tính; chuỗi "**RESTART**" thì ứng dụng kia sẽ khởi động lại máy tính (dùng hàm API ExitWindows).

119

Bài tập

3. Viết chương trình TCP (ứng dụng A) đặt trên một máy thực hiện chức năng sau: Khi ứng dụng (B) gửi một chuỗi chữ tiếng Anh thì ứng dụng A sẽ gửi trả lại nghĩa tiếng Việt tương ứng. Nếu từ tiếng Anh không có trong từ điển (từ điển ở đây chỉ có 4 từ Hello, Student, Teacher, Goodbye) thì ứng dụng A gửi trả lại chuỗi "Không tìm thấy".

120