Глава 1

Защита информации в операционных системах

1.1 Сетевое программирование на C

Сегодня миллионы компьютеров и устройств связаны в глобальную сеть интернет, либо в отдельные локальные подсети. В связи с этим возникает необходимость создания приложений, которые бы использовали все преимущества передачи данных по сети. Например, одним из распространенных приложений, которое использует передачу по сети, является веб-браузер. И платформа .NET и язык программирования C предоставляют все необходимые возможности для создания приложений, которые могут взаимодействовать по сети и использовать различные сетевые протоколы.

Но прежде чем переходить непосредственно к созданию приложений, надо пару слов сказать, что вообще представляет собой коммуникация в сети.

Вся сеть состоит из отдельных элементов - хостов, которые представляют собой компьютеры и другие подключенные устройства. Между собой они соединены каналами связи (кабели Ethernet, Wi-Fi и т.д.) и маршрутизаторами. Маршрутизаторы объединяют компьютеры в подсети и контролируют передачу данных между ними.

Но компьютеры-хосты не взаимодействуют абы как между собой. Они применяют протоколы. Протокол представляет собой соглашения о том, как пакеты данных будут передаваться по каналам коммуникации. Таким образом, протокол упорядочивает взаимодействие.

Существует множество различных протоколов. Протоколы, которые используются для передачи данных по сети, составляют семейство про-

токолов TCP/IP. Основные из них: Internet Protocol (IP), Transmission Control Protocol (TCP) и User Datagram Protocol (UDP). Причем эти протоколы организованы в уровневую систему:

IP представляет сетевой уровень. Он использует нижележащие уровни, которые представляют физические каналы коммуникации - кабели Ethernet и т.д., для передачи пакетов с данными другому хосту.

Выше IP располагается транспортный уровень, который образуют протоколы TCP и UDP. Эти протоколы используют определенные порты для передачи данных. TCP позволяет отследить потерю пакетов и их дублирование при передаче. UDP подобного не позволяет сделать и нацелен на простую передачу данных.

Однако приложение взаимодействует с уровнем TCP / UDP не напрямую, а через специальный API, который предоставляют сокеты. Сокеты - это не какой-либо протокол, это просто интерфейс для создания сетевых приложений, который опирается на встроенные возможности операционной системы.

В зависимости от используемого протокола различают два вида сокетов: потоковые сокеты (stream socket) и дейтаграммные сокеты (datagram socket). Потоковые сокеты используют протокол TCP, дейтаграммные - протокол UDP.

В итоге, когда приложение посылает сообщение приложению, запущенному на другом хосте, то приложение обращается к сокетам для передачи данных на уровень TCP / UDP. Далее с этого транспортного уровня данные передаются сетевому уровню - уровню протокола IP. И этот протокол передает данные далее физическим уровням, и после этого данные уходят по сети.

Чтобы уникально определять хосты в сети каждый хост имеет адрес. Существует несколько различных протоколов адресов. В настоящее время наиболее распространен протокол IPv4, который предполагает представление адреса в виде 32-битного числа, например, 37.120.16.63. Такой адрес содержит четыре числа, разделенных точками, и каждое число находится в диапазоне от 0 до 255. Однако также в последнее время набирает оборот использование адресов протокола IPv6, которые представляют собой 128-битное значение.

Однако такие адреса очень сложно запомнить, поэтому в реальности чаще оперируют доменами. Домены представляют специальные названия, используемые для интернет-адресов. Например, есть доменное имя "www.microsoft.com ему соответствует адрес в формате IPv4 2.23.143.150. Но для протокола IP, через который идет взаимодействие, доменные адреса не существуют. Поэтому при отправке или передаче данных по доменному имени, компьютер еще обращается к службам Domain Name

System (DNS), который выполняют сопоставление между интернет-адресами в формате IPv4 или IPv6 и доменными названиями.

Кроме адреса при сетевых взаимодействиях используются порты. Порт представляет 16-битное число в диапазоне от 1 до 65 535. Использование портов позволяет разграничить несколько запущенных приложений на одном хосте.

Ключевыми компонентами сетевого взаимодействия являются клиент и сервер. Клиент посылает запрос, а сервер получает запрос, обрабатывает его и посылает обратно клиенту некоторый ответ. Простейший пример - веб-браузер, который служит в качестве клиента, отправляя запрос на некоторый сайт. А сайт выступает в качестве сервера, отправляя браузеру некоторый ответ, который браузер затем отображает пользователю. Однако в реальности нередко одно приложение может выступать и в качестве сервера, и в качестве клиента.

Собственно, это все базовые принципы взаимодействия по сети, которые надо знать. В реальности, как правило, при создании приложений не потребуется глубокого знания всех протоколов и нюансов их работы. Если в редких случаях возникнет необходимость более детального знания протоколов, то в этом случае можно обратиться к специализированной литературе, в данном же руководстве мы сосредоточимся непосредственно на тех возможностях, которые предоставляет фреймворк .NET для работы с сетью.

Основная функциональность фреймворка .NET по работе с сетями содержится в пакете System.Net. Также есть дополнительные пакеты:

System.Net.Http: содержит функциональность по работе с протоколом HTTP

System.Net.NetworkInformation: редоставляет доступ к данным о сетевом трафике и сетевых адресах, а также к прочей информации о хостах сети. Также предоставляет функциональность ping

System.Net.Security: предоставляет сетевые потоки для безопасной связи между хостами

System.Net.Sockets: предоставляет доступ к функциональности сокетов операционной системы

System.Net.WebSockets: предоставляет доступ к реализации инфтерфейса WebSocket

System.Net.Quic: содержит типы, которые реализуют протокол QUIC в соответствии со спецификацией RFC 9000.(**??**)

Глава 2

Отправка и получение данных в TCP. Однонаправленная связь между сокетами

В листинге 2.1 Пример сетевого программирования:

Листинг 2.1: Поиск основных состояний

```
1     using System;
2     using System.Collections.Generic;
3     using System.Linq;
4     using System.Text;
5     using System.Threading.Tasks;
6     using System.Net.Sockets;
7
8     namespace Krestiki_and_zeriki
9     {
10      class Program
11      {
12        static void Main(string[] args)
13        {
14          Socket socket = new Socket(AddressFamily.
                InterNetwork, SocketType.Stream, ProtocolType.
                Tcp);
15          String a;
16          Socket cl;
17          byte[] mas = new byte[20];
18          a = Console.ReadLine();
19          if (a == "s")
20          {
```

```
21              socket.Bind(new System.Net.IPEndPoint(System.
                    Net.IPAddress.Parse("127.0.0.1"), 8888));
22              socket.Listen(0);
23              cl = socket.Accept();
24              String str = "zxzxzxzx";
25
26              mas=Encoding.UTF8.GetBytes(str);
27              cl.Send(mas);
28            }
29            else
30            {
31              try
32              {
33                socket.Connect(new System.Net.IPEndPoint(
                      System.Net.IPAddress.Parse("127.0.0.1"),
                      8888));
34              }
35              catch
36              {
37                Console.WriteLine("                          .");
38                return;
39              }
40              int xcv = socket.Receive(mas);
41              string str;
42              str=Encoding.UTF8.GetString(mas,0,xcv);
43              Console.WriteLine(str);
44            }
45          }
46        }
47      }
```