top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

СЕТЕВОЕ ПРОГРАММИРОВАНИЕ

СЕТЕВОЕ ПРОГРАММИРОВАНИЕ В NET FRAMEWORK

TCP И UDP СОКЕТЫ, UNICAST, BROADCAST, MULTICAST

ИСПОЛЬЗОВАНИЕ СЕТЕВЫХ ПРОТОКОЛОВ HTTP, SMTP, FTP

Урок №5

Unicast, broadcast, multicast

Содержание

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

1. Что такое Unicast

Схема маршрутизации *unicast* предполагает, что пакет отправляется только одному сетевому устройству от другого сетевого устройства напрямую.

Эта схема маршрутизации может быть использована как через TCP, так и через UDP-протокол.

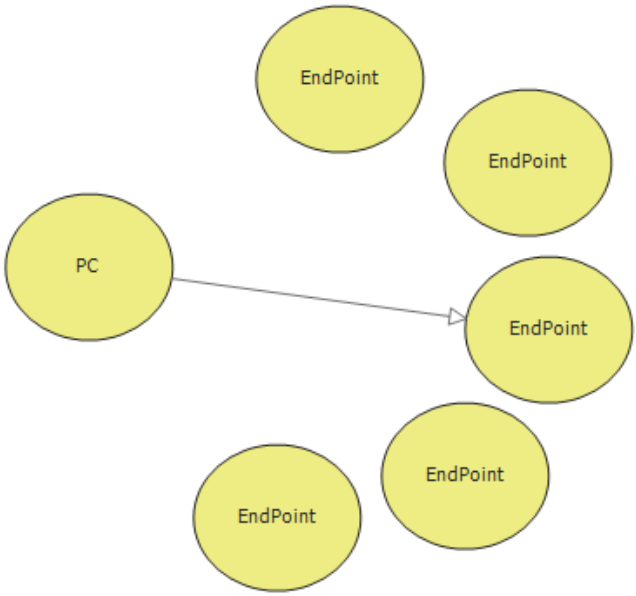

Рисунок 1

Смысл состоит в том, что данные передаются одному реципиенту. Но это, в свою очередь не означает, что они передаются всегда на один и тот же компьютер, поскольку, один компьютер может иметь несколько сетевых устройств, а значит в один момент времени обладать несколькими

IP-адресами, и даже находиться в различных подсетях. И, также, может в различный момент времени менять IP-адрес.

Также необходимо понимать, что используя схему маршрутизации unicast при отправке одного и того же пакета многим получателям, Вам придётся вручную отправлять данные каждому реципиенту отдельно.

2. Что такое Broadcast

Broadcast (разновидность широковещательного канала) схема маршрутизации — это специальный подвид сетевого взаимодействия, который предполагает что после получения пакета, роутер передаёт этот пакет всем компьютерам, которые соответствуют некоторому адресному пространству (подмножеству). Например, все компьютеры подсети, или вообще всем доступным компьютерам.

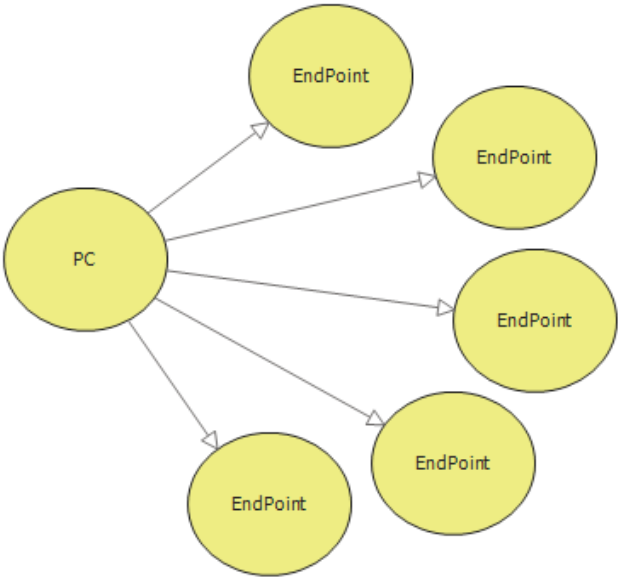

Рисунок 2

Однако, несмотря на то, что *broadcast* схема маршрутизации, может показаться очень привлекательной, она сильно расходует ресурсы сети, а значит использовать её нужно как можно уже, ведь дейтаграмма отправляется не

только тем компьютерам, которые ожидают сообщения, а вообще всем. Так же многие виды сетевых атак, например, такая, как широковещательный шторм, используют широковещательный канал. Поэтому на современном этапе многие роутеры блокируют широковещательные запросы в целях безопасности. И, если вы решите разработать чат, используя широковещательный канал, то в рамках локальной подсети он работать будет, но из неё, скорее всего пакеты выходить не будут.

Для того, чтобы отправить broadcast-пакет нужно создать UDP-сокет и отправить дейтаграмму используя broadcast-адрес. Для определения broadcast-адреса нужно применить побитовую операцию конъюнкции к первому адресу подсети (например, в подсети с маской 255.255.0.0 первым адресом будет 192.168.0.0) и дополнению маски подсети (0.0.255.255). Таким образом получается, что 192.168.0.0 | 0.0.255.255 = 192.168.255.255.

Или в бинарном виде это будет выглядеть следующим образом:

```
11000000101010000000000000000000
OR
    00000000000000001111111111111111
    --------------------------------
= 11000000101010001111111111111111
```

После того, как был определён широковещательный адрес мы создаём сокет и отправляем запрос.

```
Socket soc = new Socket(
AddressFamily.InterNetwork, SocketType.Dgram,
       ProtocolType.Udp);
```

```
IPEndPoint ipep = new IPEndPoint(IPAddress.
      Parse("192.168.255.255"), 1234);
soc.Connect(ipep);
string message = "Hello network!!!";
soc.Send(Encoding.Default.GetBytes(message));
```

3. Что такое Multicast

Multicast — это такая технология сетевой адресации, при которой сообщение доставляются сразу группе получателей. При этом используется стратегия наиболее эффективного способа доставки данных: сообщение передаётся через одни сегмент сети только один раз; данные копируются только тогда, когда направление к получателям разделяется.

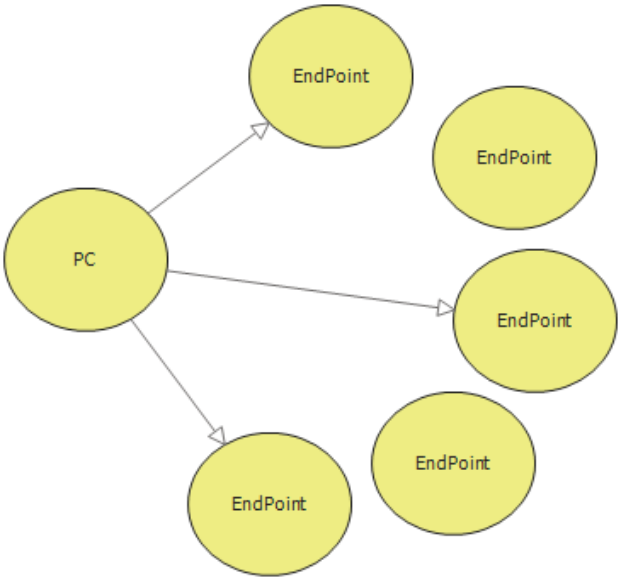

Рисунок 3

Слово *multicast*, как правило, используется как ссылка на **IP Multicast** — метод отправки IP дейтаграмм («в народе» называемых «пакетами») группе получателей за один сеанс отправки данных.

Согласно стандарта RFC 3171, адреса в диапазоне от 224.0.0.0 до 239.255.255.255 используются протоколом *IPv4* как multicast-адреса и формируют класс адресов «D». Как только отправитель посылает дейтаграмму на некоторый multicast-адрес, роутер немедленно перенаправляет её всем получателям, которые зарегистрировались на получение информации с этого multicast-адреса.

Для того, чтобы осуществить отправку данных на multicast-адрес средствами .NET Framework необходимо сперва создать обычный UDP-сокет.

```
Socket someMulticastSocket = new Socket(
    AddressFamily.InterNetwork,
    SocketType.Dgram,
    ProtocolType.Udp);
```

Затем необходимо установить опцию MulticastTimeTolive, которая влияет на время жизни пакета. Если установить её в значение 1, то пакет не выйдет за пределы локальной сети. Если же установить её в значение отличное от 1, то дейтаграмма будет проходить через несколько роутеров.

Установка этой опции осуществляется посредством вызова компонентного метода SetSocketOption класса Socket.

```
someMulticastSocket.SetSocketOption(
    SocketOptionLevel.IP,
    SocketOptionName.MulticastTimeToLive, 2);
```

Следующим действием мы создаём объект класса IPAddress, описывающий некоторый, выбранный нами, multicast-адрес. Регистрируем этот адрес, для созданного

нами сокета посредством вызова метода SetSocketOption, создаём на базе этого адреса конечную точку соединения и соединяем наш сокет с этой конечной точкой.

```
IPAddress dest = IPAddress.Parse("224.5.5.5");
someMulticastSocket.SetSocketOption(
    SocketOptionLevel.IP,
    SocketOptionName.AddMembership,
new MulticastOption(dest));
IPEndPoint ipep = new IPEndPoint(dest, 4567);
someMulticastSocket.Connect(ipep);
```

Теперь мы можем осуществлять передачу multicast-сообщений. Поэтому мы можем перейти к вызову метода Send нашего сокета.

```
string message = "Hello network!!!";
someMulticastSocket.Send(Encoding.Default.
    GetBytes(message));
```

4. Пример реализации musticast приложения

Мы рассмотрим использование схемы маршрутизации *multicast* на примере оконного приложения, осуществляющего рассылку введённой пользователем информации всем зарегистрированным пользователям (т.е. тем пользователям, у которых запущено клиентское приложение) и представляет собой доску объявлений. Мы реализуем клиент-серверную модель с целью унификации интерфейса пользователя.

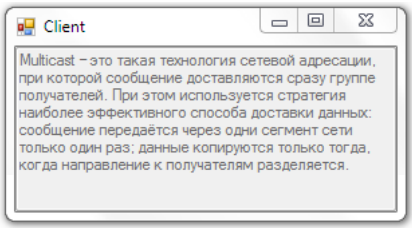

Рисунок 4

Интерфейс клиентского приложения будет представлен окном, в котором размещено текстовое поле в режиме multiline и свойство enabled которого установлено в значение false, чтобы пользователь не мог редактировать приходящие к нему сообщения.

Интерфейс серверной части будет выглядеть похожим образом, с одним исключением — поле должно позволять редактирование, чтобы пользователь мог менять сообщения, на «доске сообщений».

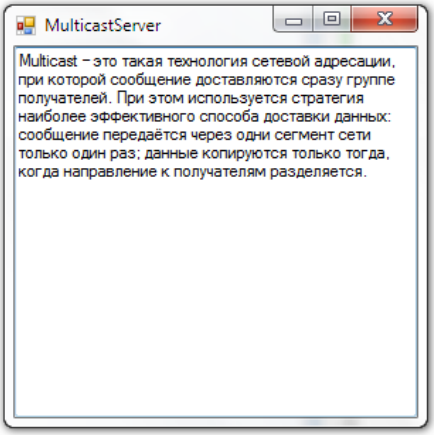

Рисунок 5

Создайте два оконных проекта в одном решении и отредактируйте их интерфейс так, как показано выше.

Пусть текстовое поле в серверном приложении называется textBox1 (название по умолчанию), а в клиентском — outputData.

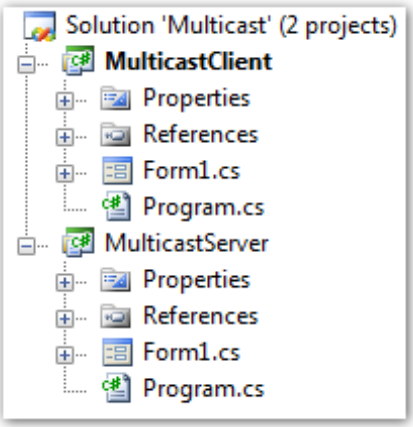

Рисунок 6

Серверная часть приложения

Нам необходимо объявить статическую переменную, которая будет хранить отправляемое сообщение, а так же целочисленную переменную, которая определит интервал, с которым актуальные данные будут отправляться получателям. Далее необходимо описать потоковый метод, который будет «заниматься» отправкой информации.

```
static string message = "Hello network!!!";
static int Interval = 1000;
static void multicastSend()
    {
        while (true)
        {
            Thread.Sleep(Interval);
            Socket soc = new Socket(
                 AddressFamily.InterNetwork,
                 SocketType.Dgram,
                 ProtocolType.Udp);
            soc.SetSocketOption(
                 SocketOptionLevel.IP,
                 SocketOptionName.MulticastTimeToLive, 2);
            IPAddress dest = IPAddress.
                 Parse("224.5.5.5");
            soc.SetSocketOption(SocketOptionLevel.IP,
                 SocketOptionName.AddMembership,
                 new MulticastOption(dest));
            IPEndPoint ipep = new
                 IPEndPoint(dest, 4567);
            soc.Connect(ipep);
            soc.Send(Encoding.Default.
                 GetBytes(message));
            soc.Close();
        }
    }
```

Как Вы видите, в потоковом методе нами открыт бесконечный цикл, в котором мы сначала приостанавливаем выполнение потока на определённый переменной Interval период, затем создаём сокет и регистрируем в нём широковещательный адрес. После этого мы инициируем соединение с широковещательным multicast-каналом, и осуществляем отправку информации. После чего закрываем сокет.

После объявления потокового метода необходимо создать объект потокового класса, и запустить его на исполнение в конструкторе. Не забудьте установить поле IsBackground потокового класса true, чтобы впоследствии не возникло проблем с его закрытием.

```
Thread Sender = new Thread(new
    ThreadStart(multicastSend));
public Form1()
{
    InitializeComponent();
    Sender.IsBackground = true;
    Sender.Start();
}
```

Последним пунктом — необходимо обработать событие TextChanged текстового поля, чтобы обновлять информацию, хранимую переменной message.

```
private void textBox1_TextChanged(object sender,
    EventArgs e)
{
    message = textBox1.Text;
}
```

Клиентская часть

В первую очередь, необходимо объявить делегат, чтобы иметь возможность впоследствии обновлять информацию в текстовом поле без ошибки меж-поточного доступа. Поскольку текстовое поле было создано в потоке отличном от того, в котором оно обновляется, могут возникать ошибки, связанные с блокировкой ресурсов.

```
delegate void AppendText(string text);
void AppendTextProc(string text)
{
    dataOutput.Text = text;
}

void Listner()
{
    while (true)
    {
        Socket soc = new Socket(
            AddressFamily.InterNetwork,
            SocketType.Dgram, ProtocolType.Udp);
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any,
                                         4567);
        soc.Bind(ipep);
        IPAddress ip = IPAddress.Parse("224.5.5.5”);
        soc.SetSocketOption(
            SocketOptionLevel.IP,
            SocketOptionName.AddMembership,
            new MulticastOption(ip, IPAddress.Any));
        byte[] buff = new byte[1024];
        soc.Receive(buff);
        this.Invoke(new AppendText(AppendTextProc),
            Encoding.Default.GetString(buff));
        soc.Close();
    }
}
```

Как Вы видите, в потоковом методе, который будет выполнять получение информации по multicast-каналу, мы открыли бесконечный цикл. В этом цикле мы создаём обычный UDP сокет, для которого регистрируем *multicast* ip-адрес. После этого мы ставим сокет в ожидание получения информации, и как только он её получил, мы вызываем метод, который обновит информацию в текстовом поле.

В классе необходимо объявить объект потокового класса, проинициализировать его, и запустить поток вызовом метода Start

По выполнении описанных выше действий можно запустить оба проекта и посмотреть работающее приложение.

```
Thread listen;
public Form1()
{
    InitializeComponent();
    listen = new Thread(new ThreadStart(Listner));
    listen.IsBackground = true;
    listen.Start();
}
```

5. Домашнее задание

С использование схемы маршрутизации *multicast*, реализуйте без серверный чат с графическим пользовательским интерфейсом. То есть такой чат, которому не нужен сервер, и который отправлял бы сообщения только тем компьютерам, которые его ожидают. Так же предусмотрите возможность мониторинга пользователей, находящихся онлайн.