УДК 621.396.2

К.Н. Яковишин

Национальный авиационный университет, Киев

ОБУЧЕНИЕ ПРОФЕССИОНАЛОВ ДЛЯ ТЕЛЕКОММУНИКАЦИЙ НА ПРИМЕРАХ МОДЕЛИРОВАНИЯ ПРОТОКОЛОВ И СЕТЕВЫХ ПРОГРАММ

Профессионал для телекоммуникаций должен знать и уметь моделировать телекоммуникационные системы и протоколы в средах быстрого проектирования методами объектно-ориентированного программирования. При этом обучение должно вестись исключительно на примерах. Как создавать такие примеры и как исследовать работу таких примеров – цель и предмет исследования данной статьи. Детально рассмотрен процесс разработки сетевых приложений на основе UDP-протокола. Промоделировано взаимодействие двух сетевых приложений – отправителя и получателя сообщений.

Ключевые слова: *обучение, примеры, моделирование, телекоммуникационные, протоколы.*

Введение

Постановка проблемы. По своей природе современные телекоммуникационные системы – это компьютерные сети со сложной иерархической организацией телекоммуникационных протоколов. Поэтому в системе знаний и навыков специалистов по телекоммуникациям на первое место по приоритету становится знание телекоммуникационных протоколов, умение моделировать телекоммуникационные системы и сети. Другими словами, особое значение приобретают знания компьютерных сетей изнутри, то есть на уровне процессов, алгоритмов и протоколов. Возникают две проблемы, каким способом студентов с фактически нулевым уровнем программирования довести до уровня самостоятельного создания приложений, а в дальнейшем – научить программировать телекоммуникационные алгоритмы.

Анализ последних достижений. В [1 – 3] дан анализ последних достижений и публикаций по проблеме подготовки VIP-специалистов для телекоммуникаций. Отмечено, что большинство книг по программированию рассчитано на подготовку профессиональных программистов. А вот как обучить создавать качественные программные продукты профессионалов-непрограммистов, например, профессионалов для телекоммуникаций, рекомендаций практически нет.

Формулирование цели статьи. При написании данной статьи ставилась цель – поделиться опытом подготовки специалистов по телекоммуникациям, сформулировать принципы и описать технику ускоренного обучения программированию на примерах в среде Borland C++ Builder 6.

Новизна статьи и новизна идей публикации. Материалы по данной тематике публикуются автором впервые. Данная статья является продолжением и конкретизацией идей, изложенных автором в предыдущих публикациях [1 – 3]. Работа впитала в себя 5-летний опыт преподавания дисциплины «Информатика» для студентов направления «Телекоммуникации». Как кажется автору, им изложен новый взгляд на технологию обучения специалистов для телекоммуникаций высокого профессионального уровня.

Разработка примера моделирования

Профессионал для телекоммуникаций должен глубоко понимать, как взаимодействуют сетевые приложения. Лучше всего познакомиться с этим можно, написав собственный пример.

Выбор адресации процессов в сети Интернет

Для обмена сообщениями между приложениями в сети Интернет необходима адресация процессов в той или иной форме. В настоящее время доминирует следующая система адресации.

Адрес процесса в сети Интернет состоит из **двух частей**:

– IP-адреса физического порта компьютера, с которого сообщение отправлено (IP-адрес порта отправителя) или на который сообщение будет принято (IP-адрес порта получателя),

– **порта** (номера программного порта в виде десятичного числа), с которого ожидает прием сообщения процесс-получатель или с которого сообщение отправил процесс-отправитель.

Например: **176.36.203.4** (IP- адрес) и **7777** (номер порта).

Параметр **порт** используется процессами в обоих протоколах - и TCP, и UDP. Порты TCP/UDP являются не только абстрактными адресами служб.

Для каждого порта операционная система поддерживает буфер в системной памяти, куда помещаются отправляемые или получаемые сообщения.

Порт задается в протоколах TCP или UDP двухбайтным адресом, поэтому ОС может поддерживать до 2^{16} портов.

Обмен сообщениями по протоколу UDP

Прикладные процессы обмениваются сообщениями-пакетами с модулями UDP через **UDP-порты.**

Формат UDP-сообщений (часто называемых дейтаграммами) описывается на рис. 1.

Рис. 1. Формат UDP-дейтаграмм

Порты в диапазоне от 0 до 1023 имеют специальное применение, а остальные порты имеют свободное применение.

Протокол **UDP** (RFC-768) обеспечивает доставку дейтаграмм без подтверждения их получения.

Протокол UDP не требует установления сеанса соединения модулей UDP. К заголовку IP-пакета протокол UDP добавляет поля **UDP-порт отправителя** и **UDP-порт получателя**.

В UDP-дейтаграмме указываются также поля: длина и контрольная сумма. Эти два параметра обеспечивают целостность данных.

Пример обмена сообщениями по протоколу UDP

Рассмотрим обмен сообщениями между двумя сетевыми приложениями по протоколу UDP. Пусть приложение с именем **UDP_Message_Receiver** будет принимать сообщение, а приложение с именем **UDP_Message_Sender** будет посылать сообщение. Пусть также оба эти приложения будут выполняться в одном компьютере, это специальный случай. Так поступают в режиме отладки и демонстрации. В общем случае любое сетевое приложение должно одинаково хорошо устанавливаться и выполняться в любом компьютере сети.

Запустим приложение **UDP_Message_Receiver.** Откроется окно (рис. 2) с формой, содержащей три поля – это: **Local Port, Message, From**. Параметр **Local Port** – это номер программного порта, по которому приложение **UDP_Message_Receiver** будет ожидать поступление сообщения. Поле **Message** пока пусто, но после приема сообщения в нем будет виден текст принятого сообщения. Поле **From** отобразит **IP- адрес** порта-отправителя сообщения, то есть приложения **UDP_Message_Sender.**

После запуска на выполнение приложение **UDP_Message_Receiver** будет находиться в состоянии ожидания прибытия сообщения.

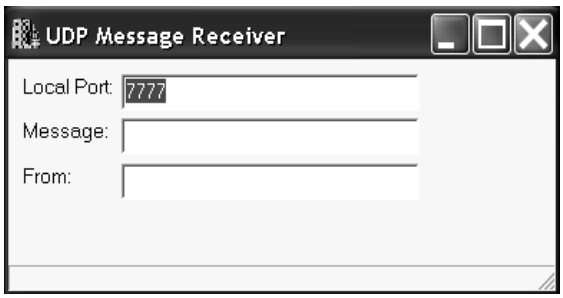

Рис. 2. Окно приложения-получателя сообщений

Запустим приложение **UDP_Message_Sender.** Откроется окно (рис. 3) с формой, содержащей три поля – это: **Remote Host, Remote Port, Message.**

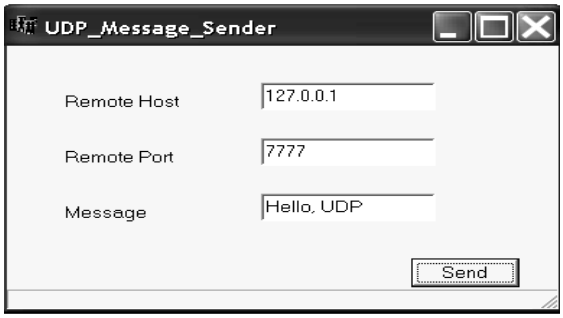

Рис. 3. Окно приложения-отправителя сообщений

Параметр **Remote Host** отображает **IP-адрес** порта-получателя сообщения. В данном конкретном случае **IP- адрес** имеет значение **127.0.0.1.** При таком значении этого **IP- адреса** физический адрес порта-приемника сообщения совпадает с физическим адресом порта-отправителя сообщения.

Это специальный случай, когда оба приложения – **UDP_Message_Receiver** и **UDP_Message** – будут выполняться в одном и том же компьютере. Такой режим, как отмечалось выше, используется программистом для отладки и упрощения демонстрации приложений. Ведь действительно, удобнее видеть взаимодействие приложений на одном мониторе (экране) компьютера. Параметр **Remote Port** – это номер программного **UDP-порта**, по которому приложение **UDP_Message_Sender** будет отсылать сообщение. Наконец, третий параметр **Message** содержит текст отсылаемого сообщения.

После запуска на выполнение приложение **UDP_Message_Sender** будет находиться в состоянии ожидания нажатия кнопки **Send**.

Нажатие кнопки **Send** запускает процесс передачи сообщения. На приемной стороне (рис. 4) вид окна приложения **UDP_Message_Receiver** изменится. В поле **Message** появится текст принятого сообщения (**Hello, UDP**). В поле **From** появится значение **IP-адреса** физического порта-отправителя сообщения.

Дополнительно в нижней части формы приложения **UDP_Message_Receiver** появится текст – информация о длине принятого сообщения (**10 bytes received –** 10байт принято).

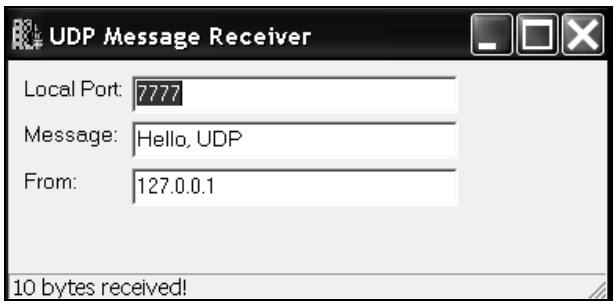

Рис. 4. Окно приложения-получателя сообщений

Создание приложения для обмена сообщениями по протоколу UDP

Рассмотрим действия программиста по созданию приложения для обмена сообщениями в сети по протоколу **UDP**. Создадим и откроем папку **\UDP_Message_Sender**. Сначала папка будет пуста. Запустим построитель Borland C++Builder 6 [4]. Будут выведены пять окон, в том числе окно стартовой формы — **Form1** (рис. 5).

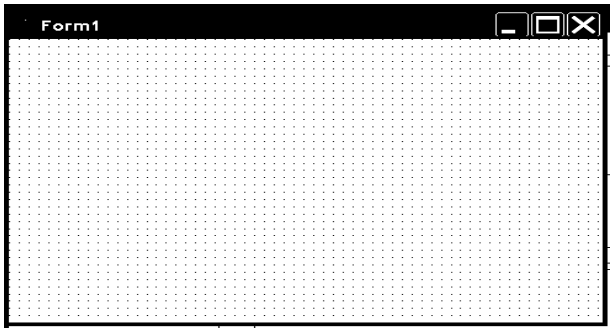

Рис. 5. Окно стартовой формы – **Form1**

В меню **File** введем команду **New | Application,** а затем сохраним проект в папке **\UDP_Message_Sender**. Как это сделать, подробно описано в [4]. Построитель самостоятельно сформирует и запишет в папку **UDP_Message_Sender** шесть файлов.

В библиотеке построителя (**BCB VCL**) есть компонента, которая может обеспечить посылку и прием сообщений по протоколу **UDP** – это компонента **TNMUDP**.

Компонента **TNMUDP** используется для отправки сообщений через Интернет по протоколу **UDP** (User Datagram Protocol). Это протокол пользовательских дейтаграмм. Стандарт протокола описан в RFC 768.

Формальное описание компоненты **TNMUDP** следующее:

- **Properties:** Name, LocalPort, RemoteHost, RemotePort
- **Methods:** SendBuffer, ReadBuffer, SendStream, ReadStream
- **Events:** OnDataSend, OnDataReceived.

Где найти компоненту **TNMUDP** и как получить ее описание? Откроем папку:

E:\Program Files\Borland\CBuilder6\Lib\Obj

и найдем файл **NMUDP** типа **DCU** (смотреть следующий рис. 6).

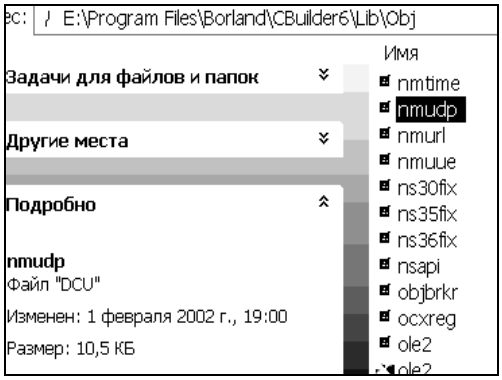

Рис. 6. Файл с компонентой **TNMUDP**

Что такое ***.dcu** - файлы? Это **Delphi Compiled Unit** (объектные файлы проекта). Эти файлы получены компилятором Delphi из исходных текстовых программных модулей.

Давайте найдем компоненту **TNMUDP** в построителе и исследуем ее. Для этого в меню **FastNet** щелкнем по компоненте **NMUDP**. Перетащим эту компоненту на форму приложения. Получим такое вот окно (рис. 7):

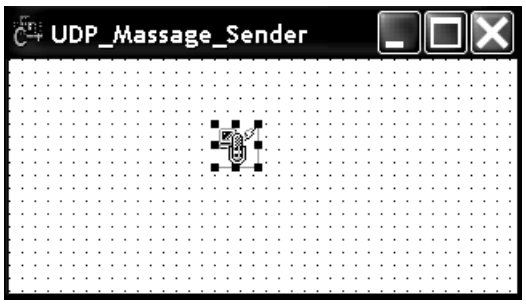

Рис. 7. Окно с компонентой **TNMUDP**

Компонента выделена на форме черными маленькими квадратиками, а в окне **Object TreeView** – серым тоном. При этом в окне **Object Inspector** можно выделить и видеть свойства компоненты (рис. 8), а можно выделить и видеть события компоненты (рис. 9).

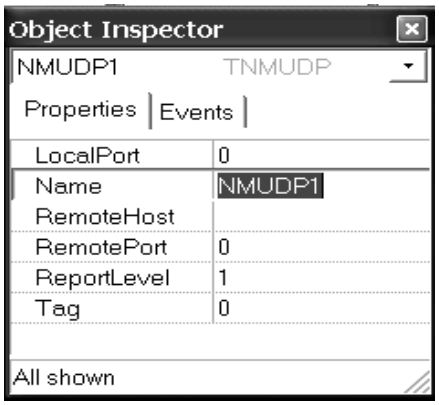

Рис. 8. Свойства компоненты **TNMUDP**

Таким образом, видно, что до отправки сообщения приложение-отправитель **UDP_Message_Sender** должно получить каким-либо способом два параметра – это:

Remote Host и **Remote Port**.

Параметр **Remote Host –** это **IP**-адрес физического порта-получателя сообщения, а параметр **Remote Port –** это программный UDP-порт приложения-получателя сообщения.

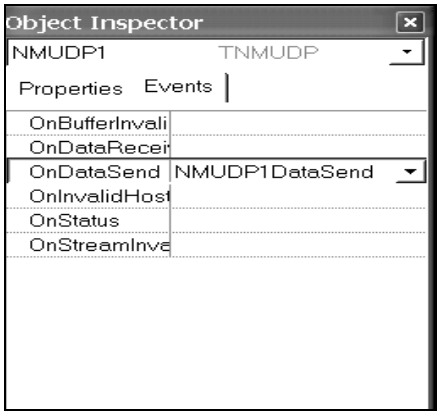

Рис. 9. События компоненты **TNMUDP**

Кроме этих двух параметров приложение-отправитель **UDP_Message_Sender** должно также знать текст отправляемого сообщения.

Должна быть также предусмотрена кнопка запуска отправки сообщения, например, с названием **SEND**.

Украшением будет вывод информации о статусе процесса отправки сообщения. Для реализации такого механизма в приложении-отправителе **UDP_Message_Sender** можно использовать три компоненты типа **Label**, три компоненты типа **Edit**, компоненту **Button** (кнопка), компоненту **StatusBar**.

Перетащим мышкой из библиотеки **BCB VCL** все названные компоненты. Кроме того, дадим новое название форме **Form1**. Для этого в окне просмотра списка объектов (**Object TreeView**) выделим объект **Form1**, а окне редактора свойств объектов (**Object Inspector**) выделим свойство **Caption** (Текст заголовка). Назначим этому свойству значение: **UDP Message Sender**. Нажмем кнопку **Сохранить**.

Вот что получится (рис. 10).

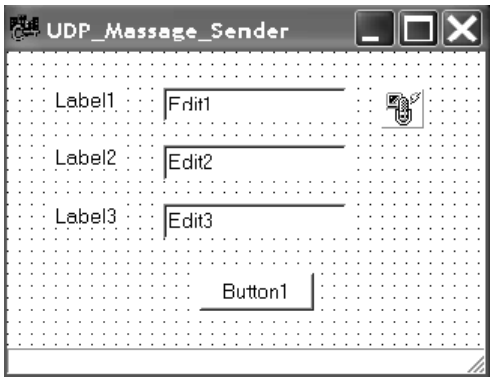

Рис.10. Весь набор компонент на форме

Теперь выполним настройку всех девяти компонент, задав необходимые свойства и события.

Выделим компоненту **Label1** и свойству **Caption** присвоим значение **Remote Host**. Аналогично компоненту **Label2** назовем **Remote Port**, а компоненту **Label3** - именем **Message**.

Свойству **Text** компоненты **Edit1** присвоим значение **127.0.0.1** (это IP-адрес физического порта). Такое специфическое значение указывает на то, что приложение-получатель сообщения установлено и работает в операционной системе того же компьютера, на котором установлено и работает приложение-отправитель сообщения. Такой механизм используется программистом в режиме отладки и демонстрационных целях.

Свойству **Text** компоненты **Edit2** присвоим значение **7777** (это номер UDP-порта) приложения-получателя сообщения.

Свойству **Text** компоненты **Edit3** присвоим значение **Hello, UDP**. Это само сообщение.

Свойству **Caption** компоненты **Button** присвоим значение **Send**.

Событию **OnClick** компоненты **Button** присвоим значение **Button1Click**.

Приложение **UDP_Message_Sender** почти готово. Однако в созданном приложении не хватает главного – в приложении нет самой процедуры отправки сообщения.

Создадим процедуру (механизм) отправки сообщения по нажатию кнопки **Send**. Другими словами, создадим реакцию на однократное нажатие кнопки **Send**, которое приведет к отправке сообщения по заданному адресу.

В тело функции **TForm1:Button1Click** модуля **Unit1.cpp** вставим следующий текст:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
//Задать IP-адрес физического порта-получателя:

NMUDP1->RemoteHost=Edit1->Text;

//Задать UDP-номер порта приложения-получателя:

NMUDP1->RemotePort=StrToInt(Edit2->Text);

//Задать буфер сообщения и его длину:

char buffer[256]; int len;

//Скопировать сообщение из Edit3 в буфер buffer:

strcpy(buffer,Edit3->Text.c_str());

len=strlen(buffer);

//Отправить сообщение методом  SendBuffer;
//компоненты NMUDP1 по адресу  127.0.0.1;

//UDP - порт с десятичным адресом = 7777:
```

```
NMUDP1->SendBuffer(buffer,256,len);

//Вывести сведения о длине сообщения;

StatusBar1->SimpleText=IntToStr(len)+" bytes sent!";
}
```

Выполним сохранение, компиляцию и построение приложения (детали смотреть в [4]).

Теперь выполним команду **RUN**. Будет выведено окно работающего приложения-отправителя сообщения **UDP_Message_Sender** (рис. 11).

Аналогично по описанной выше процедуре можно создать приложение **UDP_Message_Receiver**.

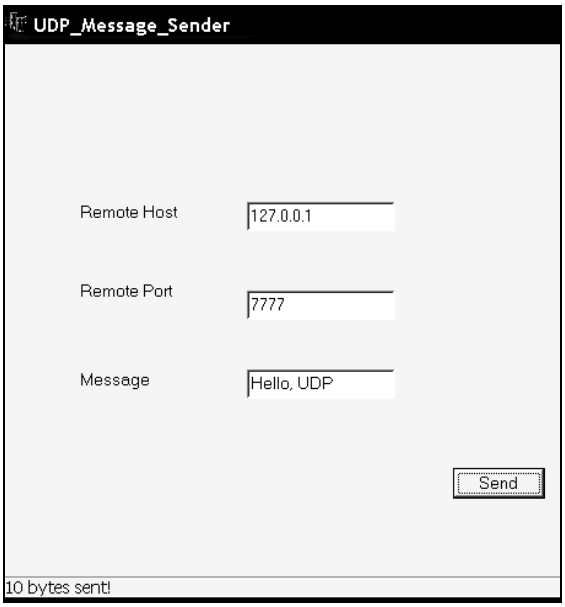

Рис. 11. Окно приложения-отправителя

Выводы

В статье рассмотрен конкретный пример создания и работы сетевых приложений по протоколу UDP на основе библиотеки визуальных компонент Borland C++ Builder 6.

Детально описана адресация сообщений в сети Интернет, приведены сведения о протоколе UDP, формате UDP-дейтаграмм.

Главное внимание уделено алгоритму взаимодействия сетевых приложений, подробно описана процедура создания сетевых приложений в системе быстрого проектирования Borland C++ Builder 6.

Материалы статьи могут быть использованы для написания курсовых работ, постановки и выполнения лабораторных работ, а также для иллюстрации лекций.

Список литературы

1. Яковишин К.Н. VIP-специалисты для телекоммуникаций и IT-индустриии / К.Н. Яковишин // Проблеми і перспективи розвитку IT-індустрії // Системи обробки інформації. – Х.: ХУ ПС, 2013. – Вип. 3(110), том 2. – С. 164-168.

2. Яковишин К.Н. Новое в учебной практике моделирования телекоммуникационных систем / К.Н. Яковишин, Э.В. Скрипчинская // Тези науково-практичної конференції „Захист інформації в інформаційно-крмунікаційних системах" – К., 2013. – С. 83-86.

3. Яковишин К.Н. Ускоренное обучение программированию в C++BUILDER профессионалов для телекоммуникаций / К.Н. Яковишин // Системи обробки інформації. – Х.: ХУ ПС, 2015. – Вип. 4(129). – С. 115-121.

4. Иллюстрированный самоучитель по C++ Builder. [Электронный ресурс]. – Режим доступа к ресурсу: http://samoucka.ru/document26927.html.

Поступила в редколлегию 19.02.2016

Рецензент: д-р техн. наук, проф. Г.Ф. Конахович, Национальный авиационный университет, Киев.

НАВЧАННЯ ПРОФЕСІОНАЛІВ ДЛЯ ТЕЛЕКОМУНІКАЦІЙ НА ПРИКЛАДАХ МОДЕЛЮВАННЯ ПРОТОКОЛІВ І МЕРЕЖЕВИХ ПРОГРАМ

К.М. Яковишин

Професіонал з телекомунікацій повинен знати і вміти моделювати телекомунікаційні системи і протоколи в середовищах швидкого проектування методами об'єктно-орієнтованого програмування. При цьому навчання повинне вестися виключно на прикладах. Як створювати такі приклади і як досліджувати роботу таких прикладів – мета і предмет дослідження цієї статті. Детально розглянутий процес розробки мережевих програм на основі UDP-протоколу. Промодельовано взаємодію двох мережевих програм – посилача і отримувача повідомлень.

***Ключові слова**: навчання, приклади, моделювання, телекомунікаційні протоколи, мережеві програми.*

HOW TO PREPARE PROFESSIONALS FOR TELECOMMUNICATIONS – BY MODELING PROTOCOLS AND NETWORK PROGRAMS

K.N. Yakovishin

Professional for telecommunications must know and able to modeling the telecommunication systems and protocols in the RAD-systems by the methods of the object-oriented programming. Thus educating must be conducted exceptionally on examples. How to create such models and how to investigate work of such models - aim and object of this article. In detail a network program development process is considered on the basis of UDP-protocol. The design of two network programs is executed - sender and recipient of reports.

***Keywords**: professionals, telecommunications, examples, modeling, protocols.*