**7.5. Практическая работа № 4. Применение широковещательных**

**IP-адресов**

**7.5.1. Цель и задачи работы**

Основнойцелью практической работы является приобретение навыков использования широковещательных адресов распределенными в локальной сети приложениями.

Результатом практической работы являются разработанное распределенное приложение, использующее широковещательные адреса.

**7.5.2. Теоретические сведения**

Теоретические сведения необходимые для выполнения практической работы изложены в разделах 3.2-3.12, 3.15 пособия.

**7.5.3. Разработка серверной части распределенного приложения**

**Задание 1.** Разработайте функцию **GetRequestFromClient**, описание которой представлено на рисунке 7.5.1. Функция предназначена для ожидания запроса клиентской программы. Предполагается, что правильный запрос (позывной сервера) состоит из набора символов, который указывается функции в качестве параметра **name**. Ожидание запроса в функции **GetRequestFromClient** осуществляется с помощью функции **recvfrom**. Если поступившее сообщение является позывным сервера, то функция, заполняет возвращаемую структуру **SOCKADDR\_IN** (параметры **from** и **flen** функции) и завершается с кодом возврата **true**. Если поступившее сообщение не является позывным сервера, то оно игнорируется, и функция вновь переходит в состояние ожидания. Если функция **recvfrom** завершается аварийно с кодом **WSAETIMEDOUT** (таблица 3.3.1), то функция **GetRequestFromClient** должна завершиться с кодом возврата **false.** Любойдругой аварийный код завершения должен приводить к исключительной ситуации (оператор **throw**), соответствующей функциям обработки ошибок, разработанных в практическом занятии № 2.

Создайте новое приложение **ServerB**, вызывающее функцию **GetRequestFromClient**.Пусть позывной сервера будет ***Hello***. Запустите приложение **ServerB** иубедитесь, что программа перешла в состояние ожидания. Запустите приложение **ClientU**, разработанное в практической работе № 3. Убедитесь, что **ServerB** не реагирует на ошибочный позывной. Исправьте в приложении **ClientU** посылаемую строку на ***Hello*** и убедитесь, что **ServerB** реагирует на правильный позывной.

// **-- обработать запрос клиента**

// **Назначение:** функция предназначена для обработки запроса

// клиентской программы

**bool GetRequestFromClient(**

**char\* name,** //[in] позывной сервера

**short port,** //[in] номер просушиваемого порта

**struct sockaddr\* from,** //[out] указатель на **SOCKADDR\_IN**

**int\* flen** //[out] указатель на размер **from**

**)**

// **Код возврата:** в случае если пришел запрос клиента, то

// функция возвращает значение **true**, иначе возвращается

// значение **false**

// **Примечание:** параметр **name –** строка, заканчивающаяся **0x00** и

// содержащая **позывной сервера** (набор символов,

// получаемый сервером от клиента и интерпретируемый,

// как запрос на установку соединения);

// параметр **from** – содержит указатель структуры,

// содержащей параметры сокета клиента приславшего

// запрос

Рисунок 7.5.1. Описание функции GetRequestFromClient

**Задание 2.** Разработайте функцию **PutAnswerToClient**, описание которой приводится на рисунке 7.5.2. Функция предназначена для подтверждения сервером запроса клиента на установку соединения. Функция оправляет в адрес клиента (параметры сокета клиента указываются в параметре **to**) свой позывной, что предполагает готовность сервера к дальнейшей работе с клиентом. Предполагается, что функция будет использоваться после завершения функции **GetRequestFromClient**. Внесите изменения в программу **ServerB**, чтобы сервер смог отвечать с помощью функции **PutAnswerToClient**  на правильный позывной полученный от клиента. Проверьте правильность работы сервера **ServerB**  с помощью программы **ClientU**.

**Задание 3.** Внеситеизмененияв программу **ServerB** таким образом, чтобы сервер отвечал на многократные запросы от разных клиентов (необходимо построить цикл с функциями **GetRequestFromClient** и **PutAnswerToClient**). Проверьте работоспособность сервера.

// **-- ответить на запрос клиента**

// **Назначение:** функция предназначена пересылки позывного

// сервера программе клиента

**bool PutAnswerToClient(**

**char\* name,** //[in] позывной сервера

**struct sockaddr\* to,** //[in] указатель на **SOCKADDR\_IN**

**int\* lto** //[in] указатель на размер **from**

**)**

// **Код возврата:** в случае успешного завершения функция

// возвращает значение **true**, иначе возвращается

// значение **false**

// **Примечание:** параметр **name –** строка, заканчивающаяся **0x00** и

// содержащая **позывной сервера** (набор символов,

// получаемый сервером от клиента и интерпретируемый,

// как запрос на установку соединения);

// параметр **to** – содержит указатель структуры,

// содержащей параметры сокета клиента

Рисунок 7.5.2. Описание функции PutAnswerToClient

**7.5.4. Разработка клиентской части распределенного приложения**

**Задание 4.** Разработайте функцию **GetServer**, описание которой приводится на рисунке 7.5.3. Функция предназначена для отправки широковещательного запроса в локальную сеть (всем компьютерам сегмента локальной сети) с позывным сервера. Предполагается, что на одном (или на нескольких) компьютере сети есть сервер **ServerB**,которыйпросушивает порт с номером, указанным в параметре **port**  функции **GetServer** . Для отправки широковещательного запроса, функция **GetServer**,должна использовать широковещательный IP-адрес (раздел 3.15 пособия). Использование широковещательного IP-адреса требует специального режима работы сокета, который устанавливается с помощью функции **setsockopt**, входящей в состав Winsock2. Описание этой функции и пример ее использования приводятся в разделе 3.15 пособия. После отправки широковещательного запроса с помощью функции **sendto**, функция **GetServer** должна вызвать функцию **recvfrom** для ожидания отклика сервера. При правильном отклике (отклик должен совпадать с позывным), функция формирует структуру **SOCKADDR\_IN**  с параметрами сокета сервера, возвращает значение **true** и завешается. Если сообщение в адрес клиента приходит, но отклик не содержит правильный позывной или функция **recvfrom** аварийно завершается с кодом **WSAETIMEDOUT**, функция должна завешаться с кодом возврата **false**. Любойдругой аварийный код завершения должен приводить к исключительной ситуации (оператор **throw**), соответствующей функциям обработки ошибок разработанных в практическом занятии № 2.

// **-- послать запрос серверу и получить ответ**

// **Назначение:** функция предназначена широковещательной

// пересылки позывного серверу и приема от

// сервера ответа.

**bool GetServer(**

**char\* call,** //[in] позывной сервера

**short port,** //[in] номер порта сервера

**struct sockaddr\* from,** //[out] указатель на **SOCKADDR\_IN**

**int\* flen** //[out] указатель на размер **from**

**)**

// **Код возврата:** в случае успешного завершения функция

// возвращает значение **true**, иначе возвращается

// значение **false**

// **Примечание:** - параметр **call –** строка, заканчивающаяся **0x00**

// и содержащая **позывной сервера;**

// - параметр **from** – содержит указатель структуры,

// которая содержит параметры сокета откликнувшегося

// сервера

Рисунок 7.5.3. Описание функции GetServer

Создайте новое приложение **ClientB**, вызывающее функцию  **GetServer**. Запустите сервер  **ServerU,** разработанной в практической работе № 3. Убедитесь с помощью отладчика, что происходит обмен данными и функция **GetServer**  завершается с возвратом **false**  после получения неверного отклика.

**7.5.5. Взаимодействие сервера и клиента**

**Задание 5.** Запустите на разных компьютерах программы **ClientB** и **ServerB**. Внесите изменения в программу **ClientB** для того, чтобы она выводила на экран консоли параметры сокета сервера откликнувшегося на позывной. Внесите изменения в программу **ServerB** для того, чтобы она выводила на экран консоли параметры клиента, отправившего правильный позывной в адрес сервера.

**Примечание.** Разработанные функции  **GetRequestFromClient** и **GetServer**  имеют существенный недостаток. После вызова функции **recvfrom**  они переводят поток в режим ожидания. Выход из этого состояния возможен лишь в том случае, если в адрес сокета поступило сообщение или будет исчерпан допустимый интервал ожидания. Такой алгоритм работы делает эти функции мало применимыми. Сохраните тексты этих функций, они будут дорабатываться в следующих практических работах.

**Задание 6.** Измените программу **ServerB** таким образом, чтобы при запуске она проверяла наличие в локальной сети еще одного такого же сервера (точнее сервера с тем же позывным) и выдавала на экран консоли предупредительное сообщение о количестве существующих серверов и их IP-адресах.