2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**2.1. Общее проектирование**

Для решения поставленной задачи необходимо реализовать клиент-серверное приложение, обеспечивающее надежную связь между мобильным приложением и компьютером после проведения процедуры авторизации пользователя.

Необходимо выделить главные аспекты, которым оно должно удовлетворять:

1. Безопасность подключения. Удаленный доступ к компьютеру предоставляется только авторизованным пользователям. Посторонние лица не должны иметь доступа к хранимым на устройстве данным.
2. Удобство использования. Интерфейс приложения и для компьютера, и для мобильного устройства, должен быть прост и понятен.
3. Функционал. Приложение должно обладать максимальным функционалом, необходимым для работы.

**Безопасность подключения**

Основным сетевым протоколом в разрабатываемом клиент-серверном приложении был выбран протокол UDP.

UDP – это сетевой протокол передачи данных без установки соединения между сервером и клиентом. Он обеспечивает быстрое, но ненадежное подключение. Таким образом, необходимо реализовать механизм авторизации подключаемого к компьютеру устройства.

Такой механизм должен включать в себя:

1. Получение сервером данных об удаленном устройстве, которое пытается к нему подключиться. К этим данным относятся: имя пользователя, наименование устройства и удаленный IP-адрес.
2. Систему авторизации пользователя удаленного устройства: при запуске серверного приложения генерируется случайный пароль, состоящий из букв латинского алфавита верхнего и нижнего регистра и цифр; пользователь мобильного устройства получает удаленный доступ к компьютеру только после ввода правильного пароля, сгенерированного серверным приложением.

**Удобство использования**

Интерфейс приложения не должен обладать сложными непонятными пользователю элементами. Весь функционал должен быть легкодоступен, и все поля для заполнения пользователем должны быть определены и подписаны.

Серверное приложение, имеющее многооконную структуру, строится средствами языка программирования C# на платформе Microsoft .NET Framework с использованием API Windows Forms.

Windows Forms позволяет разрабатывать интеллектуальные клиенты - приложения с полнофункциональным графическим интерфейсом, простые в развертывании и обновлении, способные работать при наличии или отсутствии подключения к Интернету и использующие более безопасный доступ к ресурсам на локальном компьютере по сравнению с традиционными приложениями Windows. [MSDN]

С помощью модели программирования Windows Forms достаточно просто разработать красочный, удобный и адаптивный интерфейс серверного приложения, с которым будет просто работать с любого компьютера.

Приложение-клиент, также, состоит из нескольких модулей, выполняющих свои функции. У пользователя не должно возникать проблем при необходимости получения доступа к какому-либо модулю, поэтому интерфейс приложения должен включать в себя простые и эффективные средства перехода между ними.

Разработка приложения на платформе Android средствами языка программирования Java позволяет разбивать проектируемое приложение на независимые друг от друга компоненты, каждый из которых может стать точкой входа для программы; выделять уникальные ресурсы для различных конфигураций устройств; создавать уникальные макеты в XML-файлах для устройств с разными размерами экранов; разрабатывать простой, приятный глазу и интуитивно понятный интерфейс.

**Функционал**

Функционал программного продукта можно разделить на две группы: функционал серверного приложения и функционал клиентского приложения.

Весь функционал разрабатываемого приложения доступен пользователям в полном объеме. К нему относятся: установка подключения между клиентом и сервером; удаленное управление компьютером, на котором установлено серверное приложение; осуществление текстовой или голосовой связи между пользователями клиентского и серверного приложения; настройка личных данных пользователя и генерация нового пароля для подключения; передача файлов.

**2.2. Схема и принцип работы приложения**

1. Авторизация подключения

1.1. Серверное приложение генерирует случайный восьмизначный пароль, состоящий из латинских букв верхнего и нижнего регистра и цифр, и переходит в состояние ожидания подключения.

1.2. Пользователь клиентского приложения на мобильном устройстве вводит данные для подключения: имя пользователя и IP-адрес устройства, к которому будет производиться подключение; и нажимает кнопку «Подключиться».

1.3. Если был введен корректный IP-адрес, данные о подключаемом устройстве (имя пользователя, наименование устройства) отправляются серверному приложению.

1.4. Серверное приложение получает запрос на подключение, проверяет списки сохраненных устройств. Если подключаемого устройства в них нет, то приложение отправляет ему запрос пароля. Если устройство состоит в списке сохраненных устройств, то подключение считается установленным и дальнейшие пункты пропускаются.

1.5. Приложение-клиент получает запрос пароля и отображает диалог ввода пароля для пользователя. Пользователь вводит пароль, и приложение отправляет его обратно серверу.

1.6. Сервер получает пароль, введенный пользователем приложение-клиента, и проверяет его. Если пароль верный, подключение устанавливается. Если пароль неверный, клиенту отправляется сообщение-отказ в подключении.

1.7. После установки соединения пользователям приложения-клиента и приложения-сервера становятся доступны новые модули программы, позволяющие более детально управлять установленным соединением.

2. Инициализация данных об экране

2.1. После авторизации подключения серверное приложение отправляет клиенту команду, содержащую данные об экране.

2.2. Клиент получает данные, инициализирует структуры для работы с серверным экраном и отправляет серверу запрос на получение изображения.

2.3. Сервер получает запрос на изображение и отправляет клиенту снимок экрана с компьютера.

3. Обмен данными. В каждый момент времени серверное приложение фиксирует все изменения, происходящие с изображением управляемого компьютера и отправляет их клиентскому приложению для отображения, а приложение-клиент фиксирует все действия, совершаемые пользователем (нажатия, перетаскивания, удержания, нажатие клавиш на клавиатуре и осуществление любых других команд) и отправляет серверу для воспроизведения тех же действий на управляемом компьютере.

4. Чат и голосовая связь

4.1. Структуры для работы с чатом инициализируются сразу после авторизации подключения.

4.2. При запуске окна чата или инициализации звонка в серверном приложении не происходит никаких изменений, а на время открытия модуля чата или звонка в приложении-клиенте процесс обмена данными между клиентом и сервером приостанавливается, пересылаются только команды, ответственные за осуществление голосовой или текстовой связи.

5. Соединение останавливается пользователем одного из приложений. При этом второму приложению отправляется специальная команда, указывающая, что необходимо прекратить работу всех модулей и перейти к пункту авторизации подключения, либо прекратить работу программы.

Общая схема работы программного обеспечения представлена на рисунке 2.1.



Рисунок 2.1 – Обобщенная схема работы ПО

**2.3. Проектирование пользовательского интерфейса**

Пользовательский интерфейс – важная часть любого приложения, ведь именно он является основным средством взаимодействия пользователя с программой. Интерфейс является неотъемлемой, важной и трудоемкой частью разработки приложения.

В рамках данной квалификационной работы необходимо спроектировать интерфейс клиент-серверного приложения, а именно двух его частей:

1. Интерфейс приложения для устройства на платформе Android;

2. Интерфейс приложения для компьютера.

Основные требования к разрабатываемому приложению:

1. Визуальные эффекты и анимация должны быть простыми и ненавязчивыми;

2. Дизайн приложения должен состоять из неконтрастных, не бросающихся в глаза цветов;

3. Цветовые оформления приложений для мобильного устройства и компьютера должны полностью совпадать или иметь минимальные различия;

4. Все поля для ввода данных пользователем должны быть точно обозначены и подписаны;

5. Минимизация ввода одних и тех же данных от пользователя;

6. Использование только коротких простых фраз;

7. Легкодоступность модулей приложения;

Приняв во внимание перечисленные пункты необходимо подготовить макет приложения для демонстрации. С его помощью оценивается функционал и удобство приложения.

С использованием средств языка программирования C# и модели программирования Windows Forms был разработан уникальный дизайн для заголовков и информационных частей окон серверного приложения, диалоговых окон и элементов взаимодействия с пользователем (кнопок, текстовых полей, полей для ввода и т.д.).

Средствами языка разметки XML была написана основа дизайна для мобильного приложения на платформе Android, состоящая из определения основных сочетаний цветов, настройки поведения элементов интерфейса.

Интерфейс серверного приложения представлен несколькими модулями:

1. Окно настроек подключения, которое открывается при запуске программы и остается видимым до тех пор, пока не установится соединение между клиентом и сервером.

2. Главная панель, которая дает полный доступ ко всему функционалу приложения. Может находиться в двух состояниях: свернутом и развернутом. В свернутом состоянии отображаются только иконки всех доступных функций приложения. В развернутом состоянии к иконкам добавляются подписи. Главная панель сворачивается и разворачивается с плавной анимацией.

3. Диалоговые окна.

4. Окно чата.

5. Окно звонка.

На рисунке 2.2. представлен дизайн окна серверного приложения.

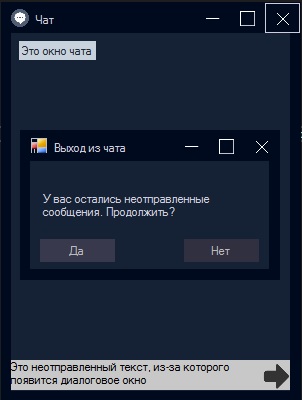


Рисунок 2.2 – Окно чата приложения-сервера

Интерфейс приложения для мобильного устройства также представлен несколькими отдельными модулями:

1. Окно подключения, которое открывается при запуске приложения. Здесь пользователь должен ввести свое имя и IP-адрес удаленного компьютера.

2. Окно управления удаленным компьютером с панелью управления всеми функциями приложения.

3. Окно чата.

4. Окно звонка.

На рисунке 2.3. представлен дизайн окон подключения и управления удаленным компьютером.

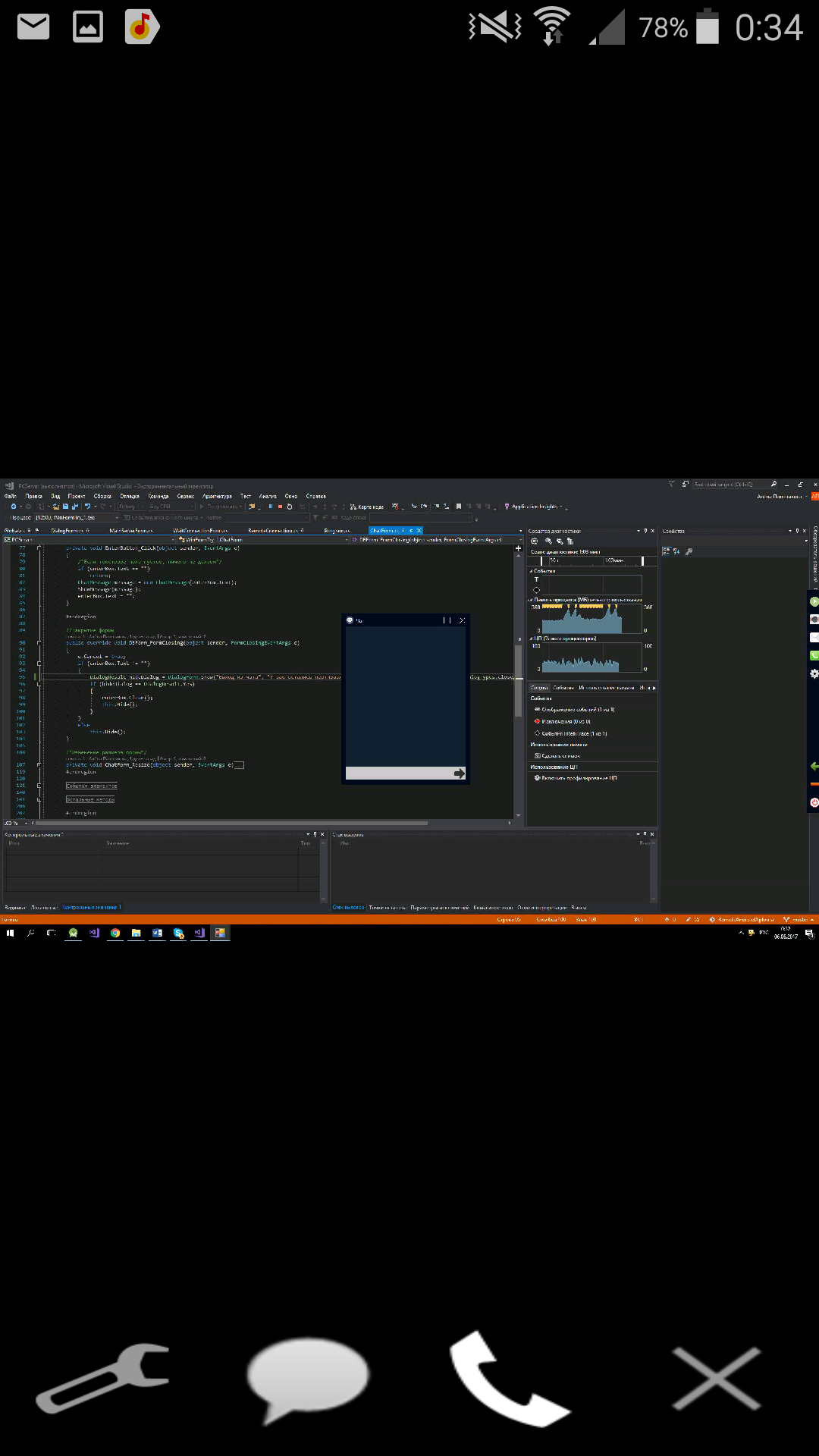
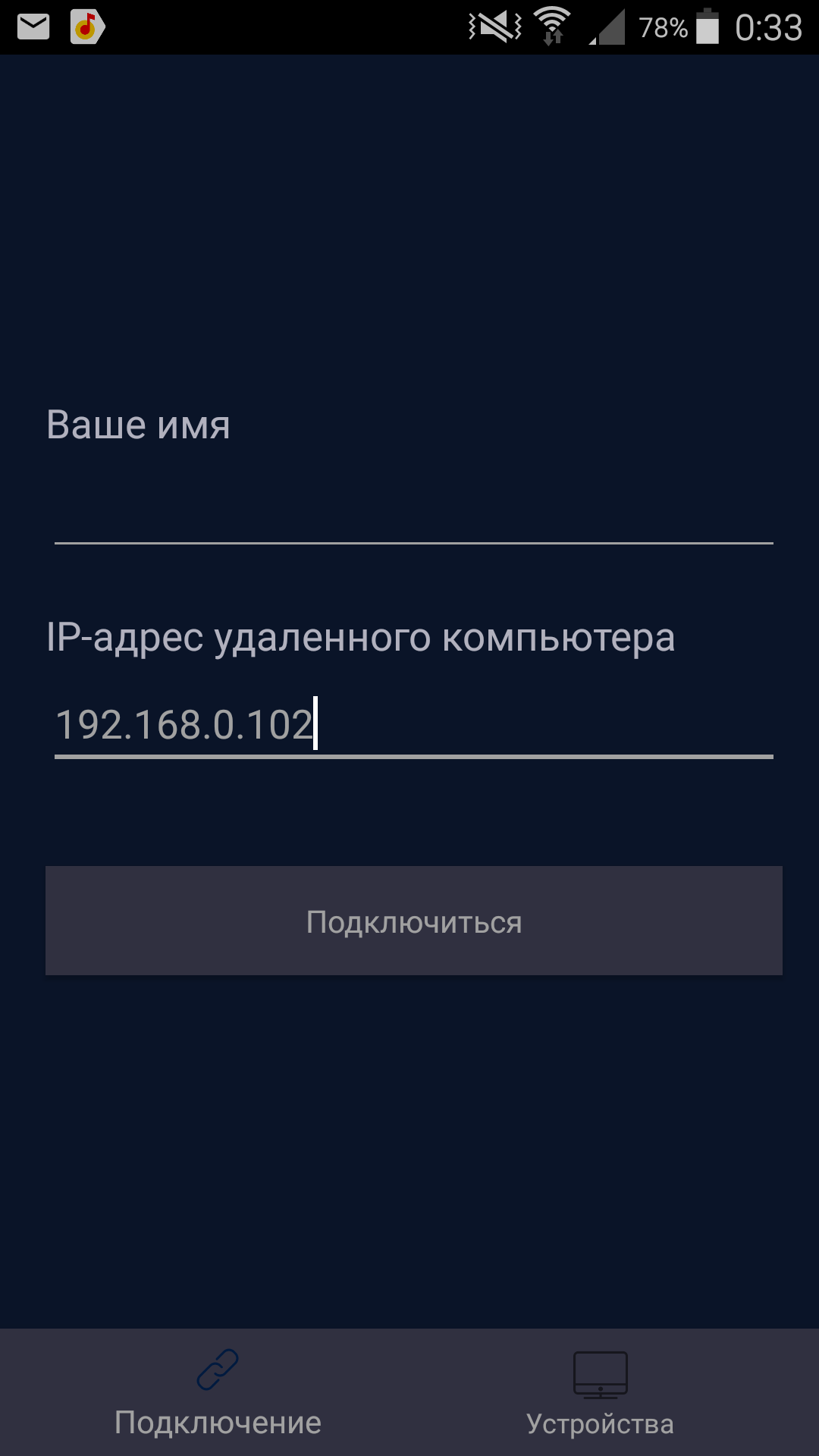


Рисунок 2.3 – Окна приложения-клиента

**2.4. Проектирование механизма передачи данных**

При реализации любого клиент-серверного приложения, прежде всего, необходимо продумать способ взаимодействия двух устройств, связываемых программным продуктом. На этапе обзора сетевых протоколов для передачи данных был выбран датаграммный протокол UDP, который предоставляет возможность быстрого обмена данными без установки соединения.

Данные между клиентом и сервером передаются в виде пакетов, представленных массивом байтов. Структура такого пакета раскрыта в таблице 2.1.

Таблица 2.1 – Общая структура пакета данных

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Команда | \ |  | , | … | , |  |
| 0 – 4 | 5 | 6 – |  |  | – |

**Описание структуры команд**

Разрабатываемое клиент-серверное приложение должно иметь четкую структуру команд для взаимодействия двух управляемых устройств. Каждая команда подает устройству, которому она предназначена, сигнал на выполнение определенного набора действий и взаимодействия с данными, которые приходят в пакете вместе с этой командой.

Пакет считается пустым, если он состоит только из кода команды.

*Команда NONE*. Данная команда, в большинстве случаев, не используется. Она может быть отправлена только в случае непредвиденной ошибки для того, чтобы дать устройству-адресату понять, что на данный момент никаких действий предпринимать не нужно.

Команда имеет код 0x00. Пакет при отправке этой команды состоит из пяти байтов – кода команды.

Команда одинаково интерпретируется и клиентом, и сервером.

*Команда HELLO*. Команда HELLO используется в случае подключения через роутер для инициализации портов для передачи данных. Она предназначена для избавления от аварийных ситуаций, возникающих из-за того, что WIFI-роутер не пропускает исходящие сообщения через порт, с которого устройству не приходило ни одного входящего сообщения.

Команда имеет код 0x01, а пакет при ее отправке также состоит из 5 байтов.

Команда HELLO одинаково интерпретируется клиентом и сервером.

*Команда INIT*. Эта команда используется для инициализации подключения. Ее может отправить только приложение-клиент.

Клиент отправляет команду INIT после нажатия пользователем кнопки «Подключиться». В пакет заносятся имя пользователя и наименование устройства, с которого будет производиться подключение.

Команда имеет код 0x02. Длина пакета суммируется из следующих значений:

– пяти байтов – длины кода команды в байтовом представлении;

– длины имени пользователя в байтовом представлении плюс один байт – разделитель;

– длины наименования устройства в байтовом представлении.

Сервер, приняв команду INIT, считывает значения имени пользователя и наименования удаленного устройства, и начинает процесс авторизации подключения.

*Команда PASSWORD*. Команда PASSWORD используется и клиентом, и сервером для проведения авторизации подключения удаленного устройства.

Сервер отправляет клиенту пустой пакет с запросом пароля.

Пакет PASSWORD, отправляемый клиентом, должен хранить в себе восьмизначный пароль, значение которого сервер сравнит со своим, сгенерированным при запуске программы, значением.

Команда имеет код 0x03. Длина пакета, отправляемого клиентом – 13 байтов. Она складывается из следующих значений:

– пять байтов – длина кода команды в байтовом представлении;

– восемь байтов – длина пароля в байтовом представлении.

*Команда CONNECT* предназначена для успешного завершения процесса авторизации подключения.

Сервер отправляет клиенту в CONNECT-пакете имя пользователя серверного приложения и имя компьютера, к которому ведется подключение. Клиент, в свою очередь, получив пакет от сервера, отправляет ему пустой пакет с командой CONNECT для завершения авторизации подключения и начала инициализации экрана.

Команда имеет код 0x04. Длина пакета, отправляемого сервером суммируется из следующих значений:

– пяти байтов – длины кода команды в байтовом представлении;

– длины имени пользователя в байтовом представлении плюс один байт – разделитель;

– длины имени компьютера в байтовом представлении.

*Команда EXIT* отправляется при необходимости остановки подключения. Пакет с данной командой может быть пустым или содержать одно значение – строку, в которой описана причина прекращения соединения.

Команда имеет код 0x06, а длина пакета с этой командой равна пяти байтам, либо суммируется из длины байтового представления кода команды и байтового представления строки, передаваемой как значение.

Команда EXIT одинаково интерпретируется и клиентом, и сервером.

*Команда SCREENINFO* оперирует при инициализации данных об экране управляемого компьютера. После установки соединения клиент отправляет серверу пустой пакет с командой SCREENINFO, а сервер должен ответить ему пакетом со всеми значениями параметров экрана. К этим параметрам относятся: количество частей, на которые будет делиться снимок экрана, количество строк и столбцов в таблице частей снимка экрана, разрешение экрана, размеры одной части снимка экрана.

Код команды – 0x09. Длина пакета, передаваемого клиенту, составляет 33 байта и суммируется из следующих значений:

– пяти байтов – длины кода команды в байтовом представлении;

– трех байтов – количества частей экрана плюс разделитель;

– шести байтов – количества строк и столбцов в таблице частей экрана, плюс двух разделителей;

– десяти байтов – разрешения экрана плюс двух разделителей;

– девяти байтов – размеров отдельной части плюс одного разделителя.

*Команда SCREEN* используется для передачи кода изображения определенной части экрана. Клиент отправляет серверу команду-запрос, состоящую из семи байтов – кода команды и номера части экрана. Сервер должен среагировать на такой запрос и отправить ответ с номером части экрана, ее координатами и байтовым представлением изображения, вырезанного из снимка экрана.

Код команды – 0x08. Пакет, отправляемый клиенту, имеет длину, суммирующуюся из следующих значений:

– пяти байтов – длины кода команды в байтовом представлении;

– трех байтов – номера части экрана плюс разделитель;

– десяти байтов – координат части экрана плюс два разделителя;

– длины байтового представления изображения, вырезанного из снимка экрана по указанным координатам с указанными размерами.