Автор: Симановский Марк Анатольевич

10 класс, Лицей № 1580 при МГТУ имени Н.Э.Баумана

Научный руководитель: Савельева Дарья Александровна

Заместитель директора по организации и оценке качества образования

Аннотация

В отличии от существующих схем управления Normal Shell и Reverse Shell, моё решение не нуждается в статическом IP-адресе, динамическом DNS, открытии экзотических портов и настройке NAT роутера или провайдера на самих персональных компьютерах, что ранее усложняло их использование, а также оно является кроссплатформенным и не потребляет много ресурсов, в том числе и интернет-трафика.

Кроссплатформенное удаленное управление персональным компьютером без привязки к IP-адресу.

RemothConsole.

Содержание

[Введение 2](#_Toc503799474)

[Актуальность 2](#_Toc503799475)

[Цель и задачи работы 2](#_Toc503799476)

[Метод 2](#_Toc503799477)

[Практическая значимость 2](#_Toc503799478)

[Теоретическая часть 3](#_Toc503799479)

[Компьютерные сети 3](#_Toc503799480)

[Сетевая архитектура «Клиент-сервер» 3](#_Toc503799481)

[Протокол передачи данных 4](#_Toc503799482)

[Программный интерфейс «Сокет». Сокеты Беркли 4](#_Toc503799483)

[TCP и UDP 5](#_Toc503799484)

[Модели удаленного администрирования 5](#_Toc503799485)

[Normal Shell 5](#_Toc503799486)

[Reverse Shell 5](#_Toc503799487)

[Основная часть 6](#_Toc503799488)

[Анализ существующих решений 6](#_Toc503799489)

[Моделирование минимально-жизнеспособного продукта 7](#_Toc503799490)

[Преимущества и недостатки модели 8](#_Toc503799491)

[Создание рабочего прототипа 9](#_Toc503799492)

[Выбор языков программирования 9](#_Toc503799493)

[Разработка серверной части 9](#_Toc503799494)

[Разработка клиентской части 10](#_Toc503799495)

[Результат 10](#_Toc503799496)

[Перспективы 11](#_Toc503799497)

[Заключение 11](#_Toc503799498)

[Список литературы 11](#_Toc503799499)

[Приложение 12](#_Toc503799500)

[Приложение 1. Иллюстрации к теоретической части 12](#_Toc503799501)

[Приложение 2. Сравнение программ удаленного администрирования 13](#_Toc503799502)

[Приложение 3. Модель удаленного управления «Middle Shell» 15](#_Toc503799503)

[Приложение 4. Листинг регистрации нового клиента-клиента 15](#_Toc503799504)

[Приложение 5. Отрисовка таблицы пользователей 16](#_Toc503799505)

[Приложение 6. Результат 16](#_Toc503799506)

# Введение

Программы удаленного администрирования компьютером позволяют дистанционно управлять другим компьютером внутри сети в реальном времени.

## Актуальность

Удаленное управление невероятно удобно, если вдруг понадобится доступ к компьютеру на расстоянии, например, чтобы скачать какой-нибудь файл с его жесткого диска или выполнить на нем какую-либо другую операцию вроде копирования и удаления файлов, запуска приложений, ведь все действия выполняются почти с любого наиближайшего ПК и нет необходимости в непосредственной близости к основному компьютеру. Средствам удаленного управления можно найти множество применений, в том числе для бытового использования и в сфере ныне популярного IoT[[1]](#footnote-1).

## Цель и задачи работы

Целью своей работы я поставил разработать кроссплатформенное средство удаленного администрирования персональным компьютером, удовлетворяющее потребностям пользователей. компьютером без перечисленных ранее недостатков коммерческих решений: отсутствие привязки к статическому IP-адресу, низкое потребление системных ресурсов, в том числе и Интернет-трафика, максимальный контроль пользователем передаваемых по сети пакетов

## Метод

1. Теоретический уровень. Оценить существующие решения.
2. Эмпирический уровень. Опросить потенциальных пользователей о недостатках текущих решений и предложениях к будущему решению.
3. Экспериментально-теоретический уровень. Моделирование удаленного администрирования на основе собранной информации.
4. Практический уровень. Создание рабочего прототипа решения
5. Экспериментально-эмпирический уровень. Пробное тестирование, опрос о недостатках и предложениях к прототипу.
6. Практический уровень. Исправление ошибок в прототипе.

## Практическая значимость

Практическая значимость моей работы заключается в том, что ее продукт уже может быть использован для удаленного администрирования персональных компьютеров с различными операционными системами.

# Теоретическая часть

## Компьютерные сети

Компьютерная (Вычислительная) сеть – система, обеспечивающая обмен данных между вычислительными устройствами.

Первая компьютерная сеть была применена на территории СССР в 1956 году в комплексе противоракетной обороны «Система А» [[2]](#footnote-2)для объединения компьютеров серии «Диана» в ИТМиВТ[[3]](#footnote-3).

В 1969 году Агенство Минобороны США создает свою сеть ARPANet[[4]](#footnote-4), изначально тоже из двух компьютеров, один из них находился в Калифорнийском Университете, второй в Стэндфордском, в 600 км от него. Тест заключался в том, что на первом оператор напечатает слово «login», а на втором оно отобразится. Но эксперимент не удался, сначала отобразились только две первые буквы (Hauben 1997). Через час ошибку устранили и слово пришло в целостности. Так зародился Интернет. В 1982 году Arpanet перейдет на новый протокол передачи данных – TCP/IP – основной протокол передачи в современной сети Интернет.

Сети принято классифицировать по следующим признакам:

1. Территориальность сети
   1. PAN – Personal Area Network – персональная сеть.
   2. LAN – Local Area Network – локальная сеть
   3. WAN – Wide Area Network – глобальная сеть
2. Архитектура
   1. Клиент-сервер
   2. Децентрализованная

## Сетевая архитектура «Клиент-сервер»

Клиент-сервер[[5]](#footnote-5) — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Клиентом и сервером могут быть ПО на вычислительных машинах, общающиеся внутри компьютерной сети с помощью сетевых протоколов. Программа-сервер работает продолжительное время и ждет поступления запросов от программы клиента, которая работает лишь во время своего запроса. После получения запроса программой-сервером, он обрабатывается и выполняется, чаще всего результатом выполнения служит ответ с данными, которые программа-клиент получает обратно. Так как сервер и клиент в общем случае лишь программы, то на одной машине может быть запущен и сервер, и клиент, таким образом для конечного клиента машина, являющаяся сервером может быть клиентом для другого сервера (так, например, устроены анонимайзеры). Это дает возможность построения многоуровневых сетей.

## Протокол передачи данных

Сетевой протокол - это набор правил, управляющих тем, как передаются данные между программами и, в данном контексте, между клиентом и сервером баз данных. Эти правила задают, помимо прочего, какой формат данных используется при их пересылке по сети.

Роль сетевого протокола заключается в его возможности обеспечивать возможность взаимодействий клиента с сервером даже в тех случаях, когда клиент и сервер баз данных находятся на разных компьютерах с разными архитектурами и операционными системами.

Существует много различных протоколов для различных целей, например, для пересылки Web-страниц используют HTTP[[6]](#footnote-6), файлов – FTP[[7]](#footnote-7), почты – SMTP[[8]](#footnote-8), протокол DTN[[9]](#footnote-9) используют в космосe. Раньше для удаленного управления вычислительной техникой использовали Telnet, но в нем не было предусмотрено проверки подлинности данных, шифрования, теперь появились и другие протоколы удаленного администрирования, например RDP, VNC, RFB.

## Программный интерфейс «Сокет». Сокеты Беркли

Сокет — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения.

API сокетов Беркли сформировал фактически стандарт абстракции для сетевых сокетов. В его интерфейсе было 6 методов:

1. Socket() – создает конечную точку соединения
2. Gethostbyaddr() – возвращает указатель на объект, описывающий сетевой узел по его адресу
3. Connect() – устанавливает соединение с сервером
4. Bind() – связывает сокет с конкретным адресом
5. Listen() – подготавливает сокет к «слушанию» входящих соединений
6. Accept() – принимает запрос на соединение

## TCP и UDP

TCP — ориентированный на соединение протокол, что означает необходимость «рукопожатия» для установки соединения между двумя хостами. Как только соединение установлено, пользователи могут отправлять данные в обоих направлениях.

UDP — более простой, основанный на сообщениях протокол без установления соединения. Протоколы такого типа не устанавливают выделенного соединения между двумя хостами. Связь достигается путём передачи информации в одном направлении от источника к получателю без проверки готовности или состояния получателя.

## Модели удаленного администрирования

Назовем компьютер, на котором мы хотим выполнять команды A, а компьютер, с которого хотим осуществлять удаленное управление – B.

### Normal Shell

В модели Normal Shell[[10]](#footnote-10) на компьютере A запущена программа-сервер, на компьютере B – программа-клиент. Для выполнения удаленной команды компьютер B посылает компьютеру A, как серверу, пакет данных с командой внутри, в ответе сервера получаем результат выполнения команды.

### Reverse Shell

В модели управления Reverse Shell на компьютере А запущена программа-клиент, на B – программа-сервер. В специальном файле на B сохраняют команду, c какой-то частотой компьютер A обращается к серверу с параметром-вопросом, есть ли для него новая команда, если она есть – то она сохраняется на компьютере A и выполняется, ответ отправляется в параметре запроса к серверу на компьютере B. Такая модель превосходит Normal Shell в простоте настройки, так как сервер в этой модели только один, а компьютеров А может быть целое множество, но если вдруг изменятся данные подключения к серверу – придется перенастраивать сразу всех клиентов.

# Основная часть

## Анализ существующих решений

Рассмотрим существующие способы удаленного администрирования ПК и их недостатки (Altbai 2018):

1. Microsoft RDP
   * Хоть и клиентская версия существует почти под все виды ОС, серверная (та, которой управляют) только для Windows
   * Проприетарная
   * Требование купить лицензию в особых случаях применения
   * Бессильна против NAT и динамических IP-адресов
2. Radmin
   * Требует покупки лицензии по истечении 30 дней использования
   * Проприетарная
   * Только для Windows в графическом режиме
   * Бессильна против NAT
3. TeamViewer
   * Требует покупки лицензии в коммерческих целях
   * После 5-15 минутного сеанса закрывает соединение с подозрениями на коммерческое применение
   * Пропиетарная, нет возможности контролировать посылаемые пакеты на удаленный сервер, используемый в целях обхода NAT
   * Большая нагрузка на процессор, сеть
4. RealVNC
   * Бессильна против NAT и динамических IP-адресов
   * Бесплатная версия не шифрует данные
   * Версии серверов только для настольных ОС, исключая OS X
5. Ammyy Admin
   * Ограничение по времени в бесплатной версии в 15 часов на месяц
   * В бесплатной версии есть реклама
   * Система премиальной роутификации: чем дороже аккаунт, тем меньше аккаунтов на один сервер
   * Пропиетарная, нет возможности контролировать посылаемые пакеты на удаленный сервер, используемый в целях обхода NAT
6. UltraVNC, TurboVNC
   * Бессильна против NAT и динамических IP-адресов
   * Версии только для настольных ОС, исключая FreeBSD, но включая Java
7. Telnet
   * Бессильна против NAT и динамических IP-адресов
   * Отсутствует шифрование
   * Нет графической версии
8. SSH
   * Бессильна против NAT и динамических IP-адресов
   * Нет графической версии

На основе этих данных можно заключить, что основные проблемы систем удаленного администрирования: отсутствие кроссплатформенности, неумение обходить NAT, закрытость исходного кода, а, следовательно, невозможность контроля действий программы. Дополнительные: отсутствие шифрования и высокое потребление ресурсов.

## Моделирование минимально-жизнеспособного продукта

Попробуем решить основные проблемы. Проблема закрытости исходного кода в новом решении решается выбором соответствующей лицензии (Симановский 2017), кроссплатформенность решим с помощью выбора языка, транслируемого в байт-код и исполняемого в виртуальной машине, например Python, Java или C#.

Попытаемся обойти NAT. Заметим, что во всех тех случаях, когда решения, не умеющие обходить NAT, доступ к сети Интернет на этих устройствах есть. При подключении к веб-серверу нет необходимости у клиента иметь уникальный IP-адрес, таковым должен обладать только сам веб-сервер. Известные модели Normal Shell и Reverse Shell не подходят, так как требуют статических данных хотя бы одного из членов схемы управления (сервера). Так как в общем случае таковых нет, то и компьютер, которым управляют, и компьютер, с которого управляют первым, новой модели оба являются клиентами. Но так как клиент не может общаться с клиентом, то нужен промежуточный сервер. Вынесем на промежуточный сервер обязанности хранения команд и их ответов, которые будут поступать от клиентов разных видов, а также выделим два вида клиентов: клиент-клиент (компьютер А из Модели удаленного администрирования) и клиент-мастер (компьютер B из той же главы). В итоге имеем следующую схему: <Приложение 3. Модель удаленного управления «Middle Shell»>. Работает она следующим образом: клиент-мастер посылает серверу пакет данных с командой, идентификатором клиента-клиента, информацией о том, что он «мастер» и аутентификационные данные; сервер принимает данные, в случае корректных аутентификационных данных – сохраняет их у себя с пометкой, какому клиенту-клиенту они предназначаются. Клиент-клиент с определенной частотой отправляет серверу пакет данных с информацией о том, кто он (уникальный идентификатор) и вопросом, нет ли для него новой команды. Сервер принимает его запрос и, если есть для клиента-клиента с соответствующим идентификатором команда – отдает ее, иначе – специальное зарезервированное слово “\_\_NOCMD\_\_” и очищает поле с переданной командой. Клиент-клиент, получив ответ, несовпадающий с указанным зарезервированным словом, выполняет текст ответа в консоли (cmd.exe для windows, sh для GNU\Linux и unix), потом отправляет серверу пакет данных с результатом команды, считанным с буферов STDOUT и STDERR. Сервер принимает и сохраняет его в соответствующее поле. Клиент-мастер через некоторое время после отправки команды пошлет запрос серверу с вопросом, нет ли ответа выполнения команды от клиента-клиента, если таковой имеется – клиент-мастер получает его в ответе, а сервер очищает поле с отправленной командой.

## Преимущества и недостатки модели

Преимущества:

* Иерархия сети не имеет значения
* Открытием портов занимается ОС
* Для использования необходимо знать лишь DNS или публичный IP адрес промежуточного сервера
* Пакеты размером от 40 Байт до 4 Килобайт отправляются в установленный пакет времени

Недостатки:

* Нужен сервер с выделенным DNS, поддержкой CGI-BIN и СУБД (НО: Серверное ПО можно установить на серверный компьютер в организации, удаленное управление будет доступно через локальную сеть)
* Нет графического представления (НО: графическое представление as is[[11]](#footnote-11) урезает кроссплатформенность. В будущем планируется построить рабочий стол на псевдографике, работающий по примеру X Windows Server)

## Создание рабочего прототипа

### Выбор языков программирования

Разработка всего прототипа делится на два этапа: разработка серверной части для промежуточного сервера и клиентской части для клиента-мастера и клиента-клиента. Так как большинство хостинг-провайдеров предлагают свои сервера с языком PHP, являющимся лидером для создания динамических веб-страниц (Tiobe 2018), то языком бэк-энда он и будет. В том же рейтинге среди кроссплатформенных языков, транслируемых в байт-код для виртуальных машин, на первом месте стоит Java еще с 2001 года (Tiobe 2018), воспользуемся ей для создания клиентской части программного комплекса. Текста команд удобнее и безопаснее хранить в базе данных на сервере, потому воспользуемся любой доступной и популярной РСУБД[[12]](#footnote-12) на основе MySQL, например MariaDB.

### Разработка серверной части

Пусть клиент-клиент всегда обращается к одному скрипту, а клиент-мастер к другому. Тогда в скрипте клиент-клиента нам необходимо два метода: регистрация нового компьютера в системе управления и работа с данными о командах и результатах их взаимодействия. Уникальным идентификатором пользователя будет последовательность из 16 случайных заглавных латинских символов. Хоть и вероятность получения двух одинаковых последовательность равна , что невероятно мало, на всякий случай будем генерировать последовательность, пока сгенерированная ранее существует в базе. В итоге получаются такие функции: <Приложение 4. Листинг регистрации нового клиента-клиента>. В главном методе просто выводим команду, проверяем, пришла ли какая-нибудь информация через POST-метод, если да – сохраняем ее в БД.

У клиента-мастер есть два режима: 1-ый – выбор компьютера для управления, второй – непосредственно сама консоль управления выбранным ранее. Для удобного отображения клиентов-клиентов выведем их в таблице. Зададим массив длин отступов в соответствии с каждым из столбцов таблицы: число и время формата “M-d-y H:i:s” занимает 27 символов, IP-адрес состоит из 4 блоков длиной в 3 цифры с разделителем-точкой, в сумме 15 символов. Будем дополнять строки до нужной длины с помощью функции str\_pad с системой заполнения STR\_PAD\_BOTH. Отрисовав шапку таблицы можем вывести ее содержимое, благодаря использованию ассоциативного массива длин отступов код вывода получается небольшим и красивым: <Приложение 5. Отрисовка таблицы пользователей>.

Обработка данных такая же простая, как и у клиента-клиента. В поле to методом POST приходит строка “GET” или “SET”, в зависимости от которой мы или сохраняем в таблицу команду, или выгружаем ее.

### Разработка клиентской части

Разобьем код на три пакета[[13]](#footnote-13): пакет выполнения команды, пакет работы с сетью и пакет работы с потоком.

1. Пакет выполнения команды «Execution»
   1. Класс Execution – выполнение строки как команды, возвращение результата как строки
      1. Проверяем операционную систему с помощью параметра System.getProperty("os.name"). Если перед нами Windows – строка для запуска консоли будет “cmd.exe”, в ином случае – “sh”
      2. Запускаем процесс консоли и подключаем к нему потоки ввода, вывода и ошибки
      3. Через поток ввода передаем команду
      4. Считываем в StringBuilder текст с потоков вывода и ошибки и возвращаем строку
2. Пакет работы с сетью
   1. Класс HTTPPost - обертка для удобной отправки пакетов по протоколу HTTP, структурный паттерн «Декоратор»
   2. Класс Service – абстрактный родительский класс клиенту-клиенту и клиенту-серверу с заранее определенными общими методами, поведенческий паттерн «Стратегия»
   3. Класс Master – реализация родительского класса для клиента-мастера
   4. Класс Client – реализация родительского класса для клиента-клиента
3. Пакет потоков – потоки вынесены в отдельный класс отдельного пакета – поведенческий паттерн «Шаблонный метод»

## Результат

В итоге получился исполняемый файл .jar программы удаленного администрирования персонального компьютера без привязки к статическому IP-адресу и умением обходить NAT, скомпилированный под настольные версии операционных систем с Java VM, со сдвоенным функционалом: его можно запустить как в роли клиента-клиента, так и клиента-мастера, достаточно лишь передать соответствующую информацию в аргументах командной строки при вызове или написать вручную желаемый режим в поток стандартного ввода и два файла скриптового языка PHP, размещаемые на сервере в любом доступном из вне каталоге.

Исходный код доступен для просмотра в моем профиле github, пример работы программы на YouTube, ссылки в приложении.

### Технические требования

|  |  |
| --- | --- |
| Фронт | Бэк |
| * Linux, OS X, Windows | * Apache/Nginx |
| * Java JRE 7+ | * PHP 4.3+ |
| * 20KB RAM для программы, 512MB+ ram для ОС и JRE | * MySQL или другая реляционная БД |
| * 15 KB+ rom | * Доступ к сети Интернет, постоянный ip-адрес или dns-имя |
| * Доступ к сети Интернет |  |

## Перспективы

* Добавить поддержку символов Unicode
* Добавить шифрование RSA
* Добавить систему макрокоманд
* Добавить веб-интерфейс управления
* Сверстать GUI и скомпилировать под android
* Добавить псевдографический сервер рабочего стола

# Заключение

В итоге мной разработана модель передачи данных между двумя компьютерами в различных локальных сетях без привязки к статическому IP-адресу, на основе которой построено удаленное управление ПК.

# Список литературы

Altbai. 2018. «Comparison of remote desktop software.» *Wikipedia - The free encyclopedia.* 12 Январь. Дата обращения: 13 Январь 2018 г. https://en.wikipedia.org/wiki/Comparison\_of\_remote\_desktop\_software.

Hauben, Michael. 1997. *History of the ARPANET.*

IBM. 2011. «Сетевой протокол.» *IBM Informix.*

John Nelson, Ivan Griffin. 1998. «Linux Network Programming.» *Linux Journal.*

Tiobe. 2018. «The Java Programming Language.» *Tiobe.* Январь. Дата обращения: 13 Январь 2018 г. https://www.tiobe.com/tiobe-index/java/.

—. 2018. «TIOBE Index for January 2018.» *TIOBE <the software quality company>.* Январь. Дата обращения: 13 Январь 2018 г. https://www.tiobe.com/tiobe-index//.

В., Коржов. 1997. *Многоуровневые системы клиент-сервер.* Открытые системы.

Симановский. 2017. «Лицензия.» Январь. Дата обращения: 13 Январь 2018 г. https://raw.githubusercontent.com/sinapsel/remothconsole/master/LICENSE.

# Приложения

## Приложение 1. Иллюстрации к теоретической части

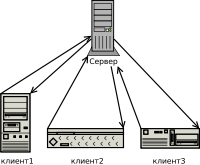


Рисунок 1. Пример клиент-серверной архитектуры

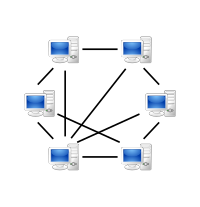


Рисунок 2. Пример децентрализованной сети

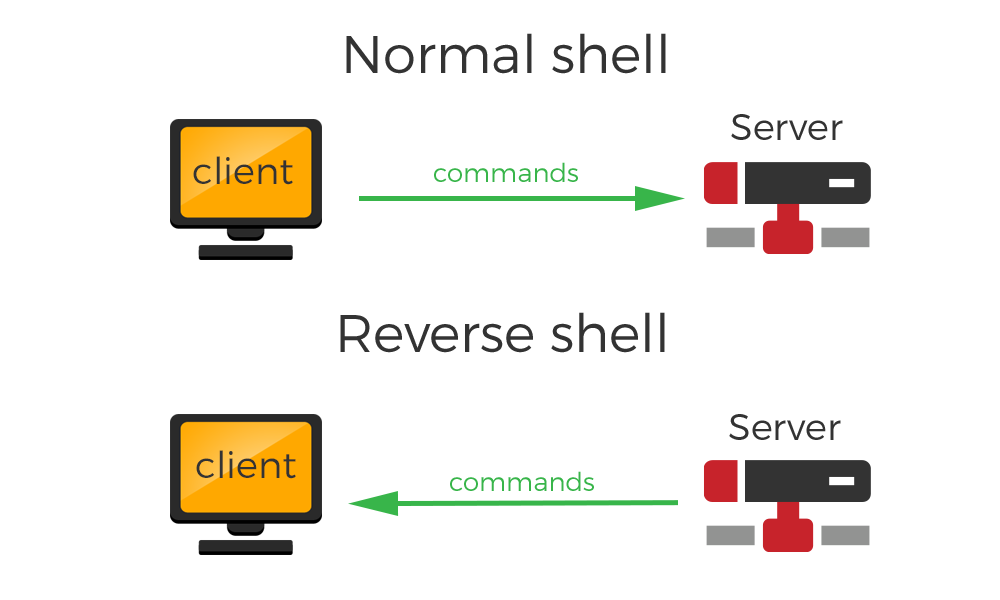
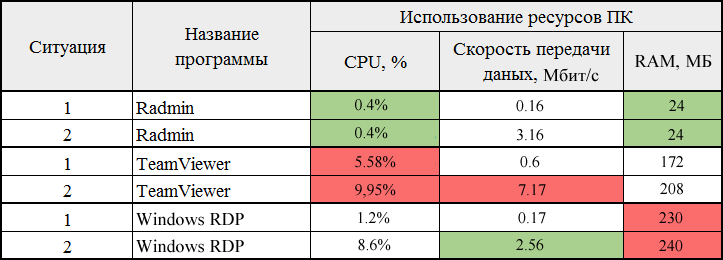


Рисунок 3. Схемы управления Normal Shell и Reverse Shell

## Приложение 2. Сравнение программ удаленного администрирования

Таблица 1. Сравнение программ удаленного администрирования. Ситуация 1 - "режим ожидание", 2 - активная работа



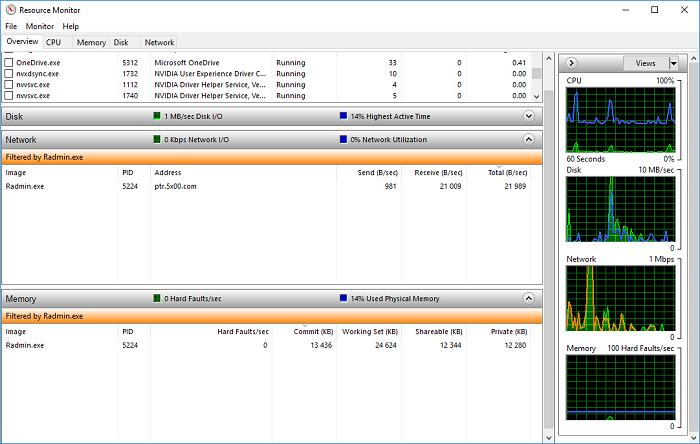


Рисунок 4. Отчет Монитора Ресурсов при использовании Radmin

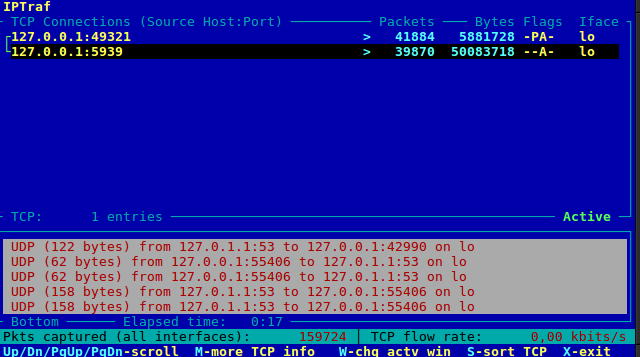


Рисунок 5. Отчет IPTraf при использовании TeamViewer на Linux

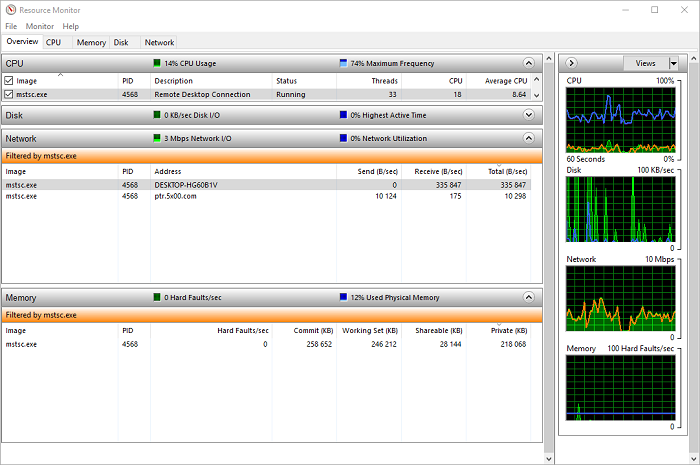


Рисунок 6. Отчет Монитора Ресурсов при использовании Microsoft Remote Desktop

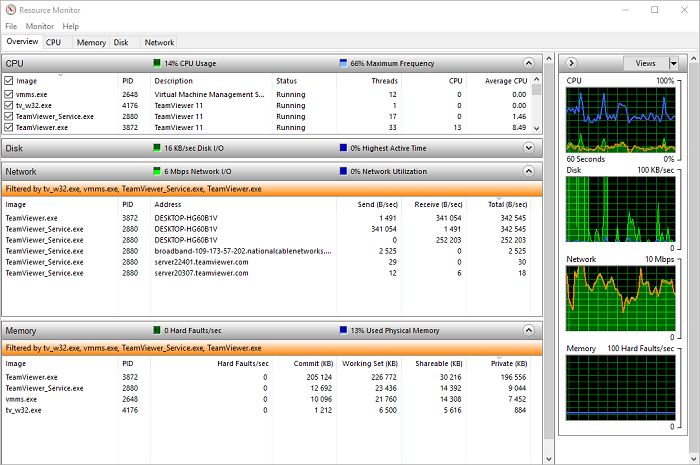


Рисунок 7. Отчет Монитора Ресурсов при использовании TeamViewer

## Приложение 3. Модель удаленного управления «Middle Shell»

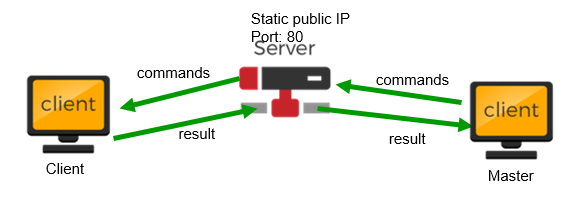


Рисунок 8. Модель удаленного администрирования "Middle Shell"

## Приложение 4. Листинг регистрации нового клиента-клиента

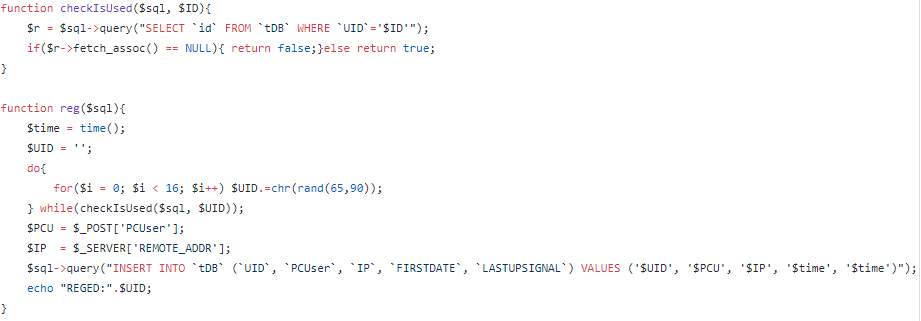


Рисунок 9. Листинг регистрации нового клиента-клиента

## Приложение 5. Отрисовка таблицы пользователей

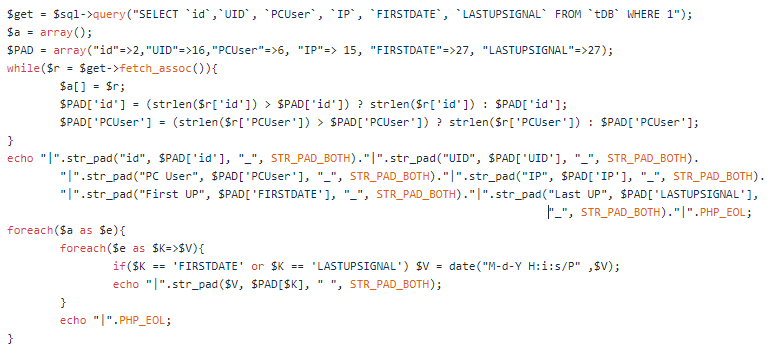


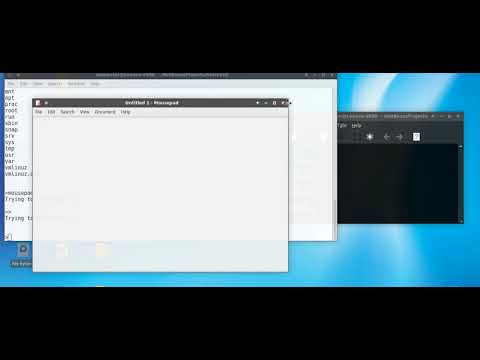
Рисунок 10. Листинг отрисовки таблицы пользователей



Рисунок 11. Пример отрисовки таблицы клиентов-клиентов

## Приложение 6. Результат

Репозиторий проекта на GitHub: <https://github.com/sinapsel/remothconsole>

Демонстрация работы на YouTube:[](https://www.youtube.com/watch?v=ve0RHmxJZ9E) <https://youtu.be/ve0RHmxJZ9E>

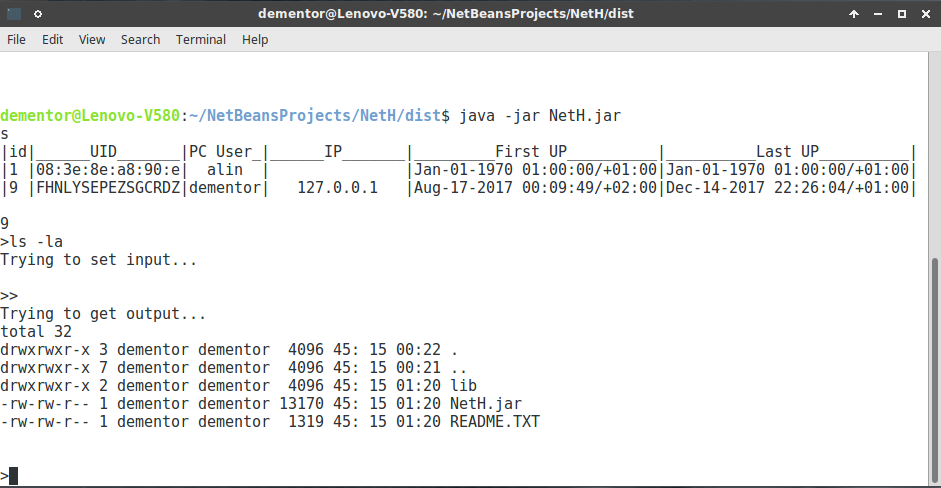


Рисунок 12. Снимок экрана при работае программы в роли клиент-мастера

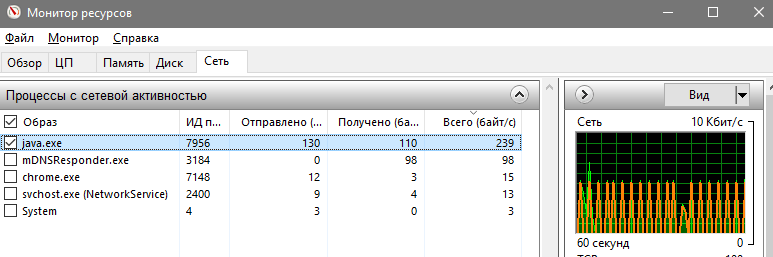


Рисунок 13. Отчет монитора ресурсов при использовании RemothConsole

1. IoT – англ. Internet of Things – Интернет Вещей - концепция вычислительной сети физических предметов («вещей»), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой [↑](#footnote-ref-1)
2. Система «А» — первый в СССР экспериментальный комплекс стратегической противоракетной обороны, 4 марта 1961 года впервые в мире осуществлён перехват боеголовки баллистической ракеты средней дальности [↑](#footnote-ref-2)
3. Институт точной механики и вычислительной техники имени С. А. Лебедева Российской академии наук [↑](#footnote-ref-3)
4. **A**dvanced **R**esearch **P**rojects **A**gency **Net**work [↑](#footnote-ref-4)
5. Англ. Client-server [↑](#footnote-ref-5)
6. Hyper Text Transfer Protocol [↑](#footnote-ref-6)
7. File Transfer Protocol [↑](#footnote-ref-7)
8. Simple Mail Transfer Protocol [↑](#footnote-ref-8)
9. Delay-tolerant networking – сеть, устойчивая к разрывам [↑](#footnote-ref-9)
10. Иногда в литературе встречается Bind Shell [↑](#footnote-ref-10)
11. As is – как оно есть [↑](#footnote-ref-11)
12. Реляционная Система Управления Базами Данных [↑](#footnote-ref-12)
13. Пакеты в java – аналоги пространств имен в C++\C# [↑](#footnote-ref-13)