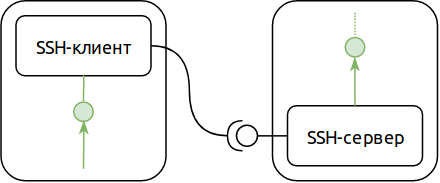
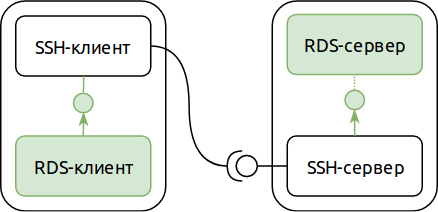
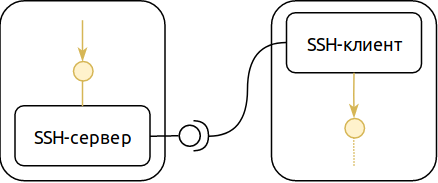
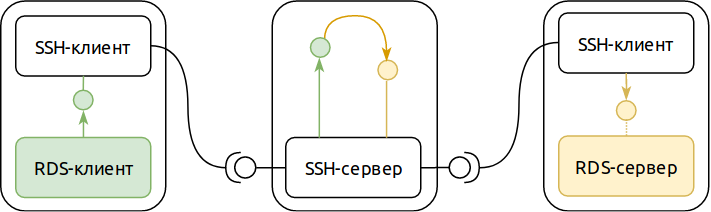
VPN без VPN или рассказ об нетрадиционном использовании SSH

* [DIY или Сделай сам](https://habr.com/ru/hub/DIY/),
* [Серверное администрирование](https://habr.com/ru/hub/s_admin/),
* [Сетевые технологии](https://habr.com/ru/hub/network_technologies/)
* [Из песочницы](https://habr.com/ru/sandbox/)

По данным ssh.com и Wikipedia, первая версия и реализация протокола SSH увидела свет в 1995 году. Задачей автора было разработать безопасную альтернативу использовавшимся тогда для удалённого администрирования rlogin, telnet и rsh. Любопытно, что появлению протокола SSH поспособствовал инцидент информационной безопасности, в результате которого злоумышленник собрал внушительную базу логинов/паролей от серверов, просто прослушивая университетскую сеть и выделяя пакеты аутентификации (пары логин/пароль в них передавались в незашифрованном виде).  
  
Протокол быстро завоевал популярность и после длительного периода доработок и улучшений был стандартизован IETF в 2006 году. С тех пор он успел стать де-факто стандартом для удалённого управления системами с текстовой консолью. Помимо собственно текстовой консоли в протоколе предусмотрена масса других полезных функций, таких как передача файлов и переадресация портов. Именно о переадресации портов (port forwarding) и её не слишком очевидном применении пойдёт речь в этой статье.  
  
В протоколе SSH предусмотрено два режима переадресации портов — прямой и обратный. Прямой режим позволяет открыть слушающий TCP-порт на стороне SSH-клиента и переадресовать все подключения к этому порту на сторону сервера.  
  
  
  
К примеру, если на нашем SSH-сервере выполняется сервер удалённого рабочего стола (Remote Desktop, RDS), мы можем использовать протокол SSH, чтобы подключиться к этому серверу, даже если прямое сетевое подключение невозможно (к примеру, заблокировано межсетевым экраном). Вот так:  
  
  
  
Второй режим переадресации портов — обратный — позволяет поменять местами роли SSH-сервера и клиента (применительно к переадресуемому порту). В обратном режиме слушающий TCP-порт открывается на стороне SSH-сервера, а приложение, обслуживающее этот порт, находится на стороне SSH-клиента. Пригождается такое редко, но, как правило, очень метко.  
  
  
  
Комбинируя эти два режима и наш пример с сервером удалённого рабочего стола можно получить вот такую конфигурацию:  
  
  
  
На первый взгляд, выглядит избыточно. Но вместе с кажущейся избыточностью мы получили два важных свойства:

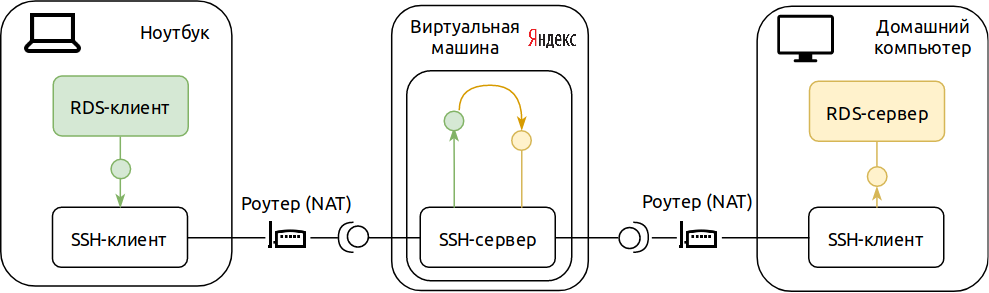
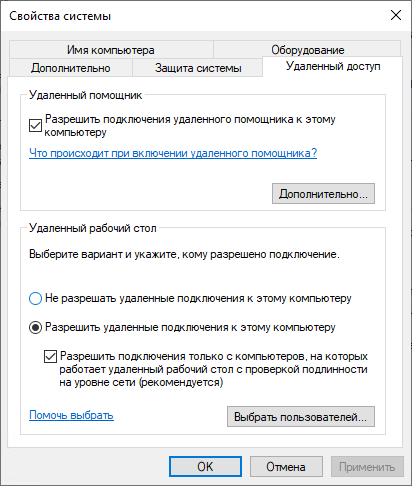
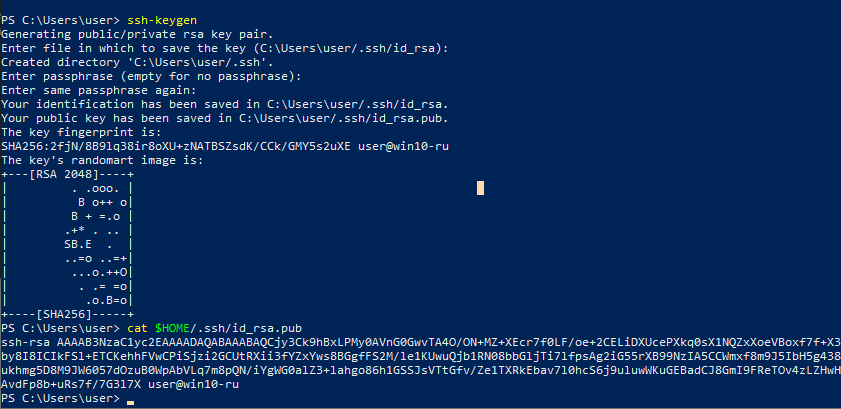
1. Единственное, что требуется от сети, чтобы такая схема работала корректно, — обеспечить постоянство сетевого адреса узла, где размещён SSH-сервер. Узлы, на которых выполняются сервер и клиент RDS, могут менять свои сетевые адреса хоть каждую минуту (либо вообще могут иметь адреса в частной сети, нам требуется лишь исходящий NAT, что часто и подразумевается под формулировкой «подключение к интернету»).
2. Несмотря на то, что для нас самих RDS-сервер доступен из любой точки интернета, другие пользователи не могут установить к нему подключение (при условии отсутствия уязвимостей в SSH-сервере). А поскольку в протоколе RDS имеется собственный контроль доступа, даже в случае успешной атаки на SSH-сервер, злоумышленнику нужно будет провести еще и атаку на RDS. Вероятность наличия одновременно уязвимости SSH-сервере и уязвимости RDS-сервера, много меньше такой вероятности для каждого компонента в отдельности.

Если приглядеться внимательнее, то на этой схеме можно разглядеть две независимые компьютерные сети. Одна — транспортная — позволяет установить SSH-подключения. Другая — внутренняя — используется для прикладных целей. Из этого наблюдения в последующих статьях мы попробуем сделать несколько интересных выводов, ну а пока вернёмся к нашим экспериментам.

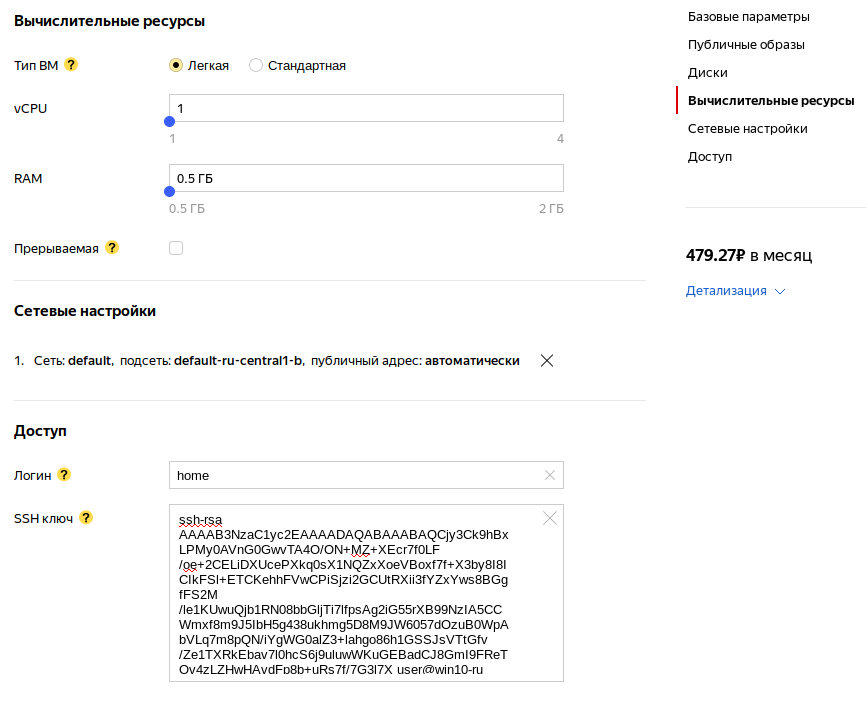
Подключись к домашнему компьютеру

На дворе XXI век, доставка сетевого пакета из Сибири в Калифорнию занимает 0.1 секунды, весь мир опутан компьютерными сетями, а TCP/IP есть даже в зубных щетках. Казалось бы, при такой-то связности всего со всем, мы должны давным давно иметь возможность управлять компьютером не только посредством физического присутствия возле устройств ввода, но и удалённо, по сети. Тем более что технологий и протоколов для этого — предостаточно. Remote Desktop от Microsoft, вариации на тему VNC в \*nix системах, решения от Citrix, тысячи их… Тем не менее мало кто из нас может похвастаться возможностью подключиться к своему домашнему компьютеру из любой точки мира при помощи какой-нибудь из этих технологий.  
  
Причин, по которым мало кто может похвастаться связью со своим собственным домашним компьютером, две, и они связаны друг с другом. Первая из них — отсутствие у типичного домашнего компьютера адреса в глобальной сети. Корни такого положения дел уходят в далёкий 1981 год, в котором был впервые описан стандарт IPv4, который и по сей день используется (само собой, с изменениями и дополнениями) для адресации большинства узлов в сети Интернет. Авторы стандарта решили, что адресного пространства мощностью в 3.7 миллиарда адресов хватит всем устройствам мира, но реальность оказалась сурова. Ожидается, что в IPv4-интернете не останется свободных адресов к сентябрю 2019 года.  
  
Кроме того, типичный пользователь интернета, который не хостит у себя веб-сайты, вполне может пользоваться всеми благами цивилизации не имея адреса в глобальной сети, вместо этого ограничиваясь лишь адресом в частной сети и провайдерским NAT'ом. Иными словами, для большинства пользователей интернета нет разницы между наличием и отсутствием глобального IP-адреса у их оборудования. В таких условиях число тех, кому провайдер выдаёт глобальный IP-адрес в глобальной сети, стремительно сокращается. В итоге типичный домашний компьютер располагается в частной сети, и глобального адреса не имеет. Даже в случае, когда провайдер всё же назначает оборудованию пользователя адрес в глобальной сети, этим оборудованием является домашний роутер, выполняющий NAT из домашней частной сети. Пользователь при этом, конечно, может «пробросить порт» на роутере, но даже в этом лучшем случае глобальный адрес роутера может меняться изо дня в день. Да, существуют провайдеры, которые за скромную плату предоставляют услугу «Статический IP», но на практике где-то к этому моменту пользователь осознаёт, что игра не стоит свеч, и если нужно что-то сделать с домашним компьютером, то можно и по телефону позвонить.  
  
Самые же настырные могут пройти этот квест до конца и столкнуться со второй причиной, по которой открывать доступ к своему компьютеру через интернет — развлечение лишь для крепких духом. Эта причина банальна — информационная безопасность. Глобальная сеть на то и глобальная, что рано или поздно в ваши ворота постучится кто-нибудь с другого конца света и с недобрыми намерениями. Насканить открытый порт, на котором отвечает сервер удалённого рабочего стола, — не слишком сложная задача, и, будьте уверены, рано или поздно к вам на супер-секретный порт 32167 прилетит TCP SYN родом из китайской деревушки.  
  
Возвращаясь назад к нашей экзотической переадресации портов при помощи SSH, можно заметить, что её особенности устраняют обе перечисленные причины.

Собери себе TeamViewer

Сразу нужно оговориться, что TeamViewer — это большой продукт с огромным количеством очень разных возможностей. Мы же в рамках этой статьи соберём себе лишь способ безопасно подключаться через сеть Интернет по протоколу Remote Desktop к компьютеру, который находится за NAT. Тем не менее, рискну предположить, что именно подключение к любому компьютеру, имеющему доступ в интернет, — главная отличительная черта TeamViewer, благодаря которой этот продукт так популярен. И именно такое подключение мы можем реализовать собственными руками, используя нашу конфигурацию SSH из первой части статьи.  
  
Итак, условия задачи: есть домашний компьютер и ноутубк, оба под управлением Windows 10. Нужно сделать так, чтобы с ноутбука иметь возможность зайти на домашний компьютер по протоколу Remote Desktop из любой точки мира. В состав нашей системы будут входить наш домашний компьютер-сервер Remote Desktop, ноутбук с клиентом Remote Desktop, и сервер SSH. Сервер SSH — единственный компонент, которому требуется глобальный IP-адрес и постоянная доступность. Самый простой вариант, удовлетворяющий этим требованиям, — разместить SSH-сервер в облаке. Яндекс.Облако отлично подходит (в первую очередь за счёт своей ценовой политики), так что будем использовать его. В итоге получается вот так:  
  
  
  
Начнём с подготовки нашего домашнего компьютера. Сперва убедимся, что удалённый доступ к нему вообще разрешён. Это можно сделать через вкладку «Удаленный доступ» в дополнительных параметрах системы:  
  
  
  
Начиная с апреля 2018 года в Windows 10 среди утилит командной строки уже присутствует ssh-клиент. Это поможет нам меньше отвлекаться на установку разного софта и сразу приступить к делу. Для начала сгенерируем себе ключ для SSH. Откроем командный интерпретатор PowerShell и выполним 'ssh-keygen'. Когда нас спросят о пароле для ключа, оставим его пустым. После генерации ключа выведем его открытую часть на консоль командой 'cat $HOME/.ssh/id\_rsa.pub'. Результат выполнения команды пригодится нам для запуска SSH-сервера в облаке. Должно получиться что-то вроде этого:  
  
  
  
Нам нужно скопировать закрытую часть ключа SSH на ноутбук. Эта часть ключа лежит в файле '$HOME/.ssh/id\_rsa' (без суффикса ".pub"), и мы можем скопировать его как обычный файл. К примеру, используя флешку (предполагаем, что она смонтирована как диск F:)

copy $HOME/.ssh/id\_rsa f:\

Теперь запустим наш SSH-сервер. Создадим виртуальную машину (ВМ) в Яндекс.Облаке. Для наших целей можно выбрать «лёгкую» ВМ с 1 vCPU и 0.5 гигабайта RAM. В разделе сетевых настроек выберем сеть по умолчанию с автоматическим IP-адресом. В раздел «доступ» в качестве логина введём «home», в поле ввода SSH-ключа скопируем то, что вывелось у нас в консоль на предыдущем шаге:  
  
  
  
Нажмём «Создать ВМ» и дождёмся завершения. После того, как создание виртуальной машины завершится, нам потребуется узнать её IP-адрес:  
  
  
  
IP-адрес нашей виртуальной машины нам понадобится, чтобы запускать SSH-клиент на домашнем компьютере и ноутбуке. Запустим его на компьютере вот таким образом (в этой и следующих командах необходимо заменить 84.201.141.36 на IP-адрес вашей ВМ):

ssh -NR 3389:localhost:3389 home@84.201.141.36

На вопрос о подключении к неизвестному серверу отвечаем «yes». Если затем мы не видим никакого текста на консоли, значит всё прошло отлично. Теперь настроим ноутбук. Скопируем закрытый ключ с нашей флешки:

mkdir -Force $HOME/.ssh

copy f:\id\_rsa $HOME/.ssh/id\_rsa

И запустим SSH-клиент:

ssh -NL 1025:localhost:3389 home@84.201.141.36

Теперь можно запускать на ноутбуке клиент удалённого доступа к рабочему столу (mstsc.exe), указав в качестве адреса подключения localhost:1025. Ура, работает!  
  
Или почти работает. Если остановить процесс SSH на домашнем компьютере, подключиться станет невозможно. Нам нужно сделать так, чтобы этот процесс автоматически запускался при старте системы, а также перезапускался при разрывах подключения. Этого можно достичь, например, создав PowerShell-скрипт и зарегистрировав его как обязательный для запуска в групповой политике при старте компьютера. Только нужно учесть, что запускаться он будет от имени системного аккаунта, а значит нужно убедиться что системному аккаунту доступен наш SSH-ключ.  
  
Сперва займёмся ключом. Запустим PowerShell от имени администратора и выполним вот такие команды:

copy $HOME/.ssh/id\_rsa "$(gwmi win32\_userprofile | where {$\_.SID -eq "S-1-5-18"} | select -ExpandProperty LocalPath)/rds\_id\_rsa"

icacls "$(gwmi win32\_userprofile | where {$\_.SID -eq "S-1-5-18"} | select -ExpandProperty LocalPath)/rds\_id\_rsa" /reset

Заодно в этом же окне PowerShell разрешим выполнение скриптов:

Set-ExecutionPolicy RemoteSigned

Теперь создадим собственно скрипт. Откроем наш любимый текстовый редактор (Notepad тоже подойдёт), и напишем вот такой скрипт (не забыв заменить IP-адрес SSH-сервера на тот, что выдан вам Яндексом):

while (1) {

& $(get-command ssh |select -expandproperty Path) `

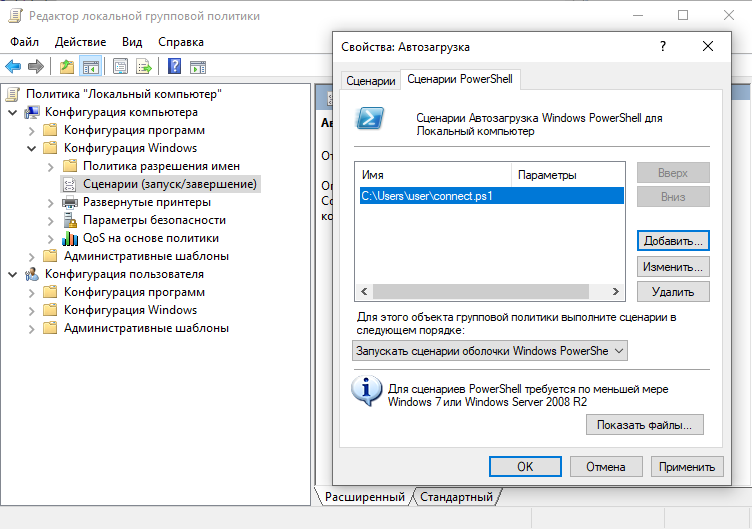
-i "$(gwmi win32\_userprofile | where {$\_.SID -eq "S-1-5-18"} | select -ExpandProperty LocalPath)/rds\_id\_rsa" `

-o StrictHostKeyChecking=accept-new -o ExitOnForwardFailure=yes `

-NR 3389:localhost:3389 home@84.201.141.36

Start-Sleep -Seconds 15

}

Сохраним скрипт в понравившуюся директорию. Наконец зарегистрируем его для автозапуска. Для этого запустим редактор групповых политик (Win+R → gpedit.msc), и откроем пункты «Конфигурация компьютера» → «Конфигурация Windows» → «Сценарии (запуск/завершение)» → «Автозагрузка». На вкладке «Сценарии PowerShell» воспользуемся кнопкой «Добавить», и укажем путь к нашему сохранённому скрипту.  
  
  
  
Аналогичные действия проделаем на ноутбуке. Сперва PowerShell от имени администратора:

copy $HOME/.ssh/id\_rsa "$(gwmi win32\_userprofile | where {$\_.SID -eq "S-1-5-18"} | select -ExpandProperty LocalPath)/rds\_id\_rsa"

icacls "$(gwmi win32\_userprofile | where {$\_.SID -eq "S-1-5-18"} | select -ExpandProperty LocalPath)/rds\_id\_rsa" /reset

Set-ExecutionPolicy RemoteSigned

Затем в текстовом редакторе подготовим скрипт (он немного отличается от предыдущего, но как и прежде, заменяем IP-адрес на тот, что был выдан вам Яндексом):

while (1) {

& $(get-command ssh |select -expandproperty Path) `

-i "$(gwmi win32\_userprofile | where {$\_.SID -eq "S-1-5-18"} | select -ExpandProperty LocalPath)/rds\_id\_rsa" `

-o StrictHostKeyChecking=accept-new -o ExitOnForwardFailure=yes `

-NL 1025:localhost:3389 home@84.201.141.36

Start-Sleep -Seconds 15

}

Неконец зарегистрируем его для запуска при старте при помощи «gpedit.msc». Перезагрузим компьютер и ноутбук (чтобы убедиться что всё корректно запускается) и вуаля! Теперь наш домашний компьютер и наш ноутбук навеки связаны друг с другом (до тех пор, пока наша виртуальная машина в Яндекс.Облаке включена и доступна).

Ну и?

Ну разве не здорово? В любом кафе, в любом аэропорту можно подключиться к себе домой и посмотреть любимые картинки с котиками. Или порадовать домашних включением 5-й симфонии Бетховена на полную громкость. Или поинтересоваться успехами своей майнинг-фермы. Или посмотреть, что творится дома при помощи веб-камеры. Да мало ли вообще применений у такой возможности «телепортироваться»? Но есть у нашего решения и недостатки.  
  
Во-первых, настройка подключения — не самый простой и приятный процесс. А в случае, если что-то пойдёт не так, отладка еще немного сложнее, чем первоначальная настройка. Само собой, эта проблема решается усидчивостью и терпением, но даже то небольшое количество труда, которое требуется затратить, может вызвать сомнения в целесообразности усилий.  
  
Во-вторых, виртуальная машина в облаке стоит денег. В случае Яндекса тот минимум, на который можно рассчитывать, это 480 рублей в месяц. Это, конечно, не запредельные деньги, но всё-таки свои, заработанные в поте лица. Стоит ли просмотр картинок с котиками этих денег — решать каждому для себя, но очень может быть, что все преимущества нашего решения будут перечёркнуты его ценой.  
  
Ценовой вопрос можно существенно сгладить, разделив расходы с друзьями и единомышленниками. Поскольку виртуальная машина используется для задач, не требующих сколь-либо заметных вычислительных мощностей, падение производительности крайне маловероятно. А экономический эффект — заметен: если арендовать виртуальную машину вдесятером, то каждому придётся заплатить лишь 48 рублей в месяц. Правда, в этом случае гармонию может нарушить вопрос доверия: любой из единомышленников имеет возможность подключиться к компьютеру собрата по SSH-серверу. В случае, когда у всех на аккаунтах установлены надёжные пароли, это не проблема. Но, скажем прямо, надёжный пароль для входа на домашний компьютер — скорее исключение, чем правило.

Продолжение следует

Итак, предположим, что мы собрали 10 единомышленников, настроили всё, как написано выше, и у всех всё работает. Наш клуб любителей картинок с котиками благополучно пользуется возможностью ходить к себе домой через интернет всего за 48 рублей в месяц без регистрации и смс, все довольны и счастливы. Вопрос в том — ограничиваются ли возможности нашей «технологии» лишь котиками и нельзя ли её использовать для чего-то более серьёзного?  
  
Само собой можно. Если заменить в наших рассуждениях «домашний компьютер» на «билд-сервер в облаке», а «ноутбук» на «рабочий компьютер в офисе», мы получаем нечто достойное звания «система доступа к инфраструктуре разработки». А если вместо билд-сервера у нас будет IP-камера, а вместо рабочего компьютера — пост охраны, получаем «систему видеонаблюдения».  
  
В обоих случаях, правда, придётся уделять больше внимания вопросам контроля доступа. В частности, при совместном использовании SSH-сервера несколькими пользователями хотелось бы изолировать этих пользователей друг от друга. А ещё при таком совместном использовании мы вынуждены назначать каждому ресурсу каждого пользователя свой отдельный TCP-порт и запоминать его номер. Адресация по номерам может вскоре стать довольно утомительной, поэтому хотелось бы иметь возможность назначать ресурсам осмысленные названия. Но о том, как улучшить ситуацию, мы поговорим уже в следующей статье.  
  
А пока спасибо за внимание и, пожалуйста, поделитесь своими мыслями в комментариях.