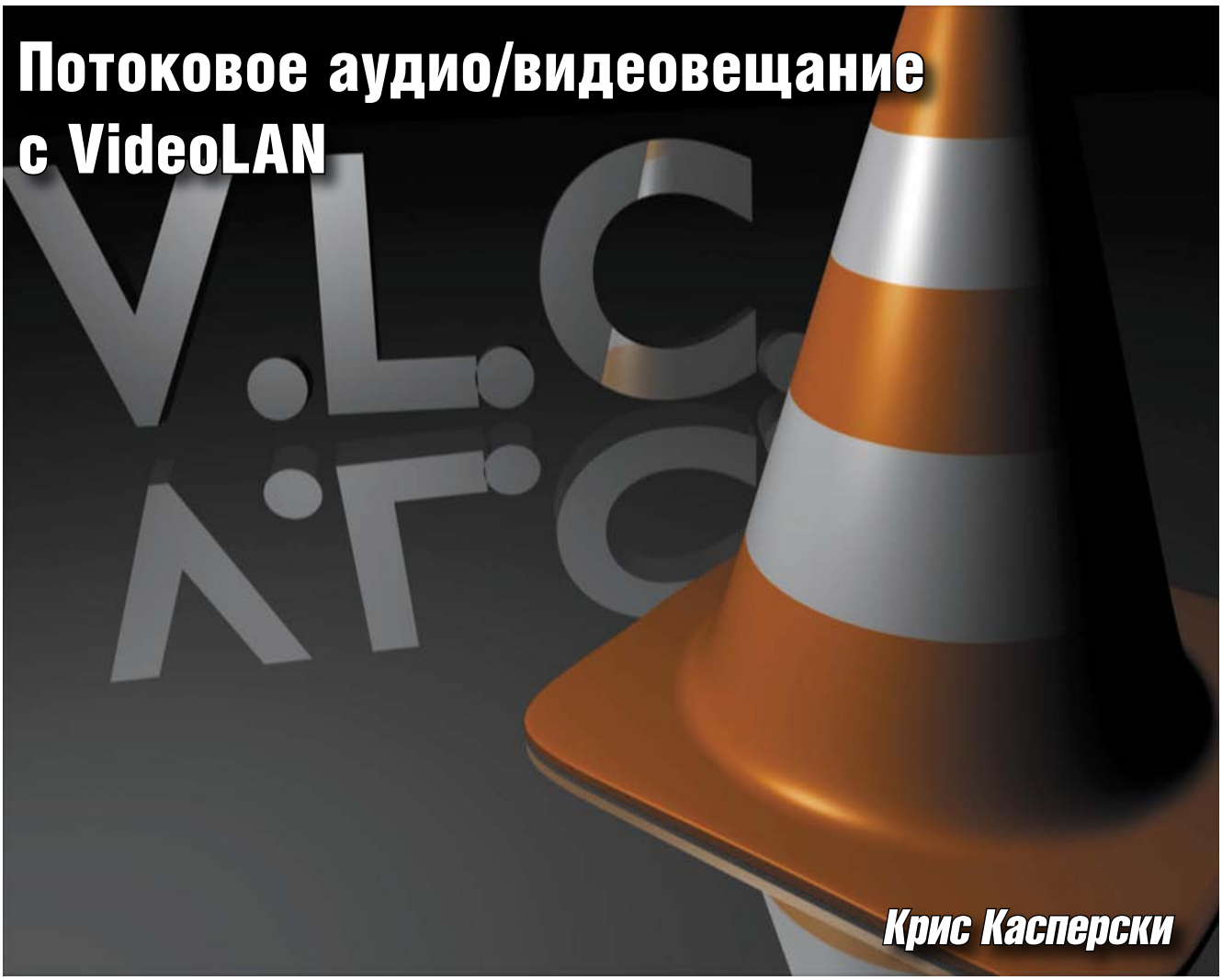
Потоковое аудио/видеовещание с VideoLAN



Крис Касперски

Прошло то время, когда для открытия собственной радио/телестанции требовалось приобретать дорогостоящее оборудование, а также собирать множество лицензий, любая из которых могла быть отозвана в самый неподходящий момент. Теперь достаточно скачать свободный программный комплекс VideoLAN и, немного повозившись с его настройками, вещать хоть на локальную сеть, хоть на весь Интернет с максимально доступным для данного канала качеством, или же использовать VideoLAN в качестве обычного медиаплейера, поддерживающего множество входных форматов с возможностью записи чужого потокового медиаконтента на жесткий диск.

Какой смысл в потоковом вещании, когда выложенный на http/ftp-сервер медиафайл может быть скачан на локальный диск пользователя и просмотрен/прослушан в любое удобное для него время бесчисленное множество раз! Потоковое вещание «привязывает» целевую аудиторию к серверу трансляции, возвращаясь к традициям десятилетней давности, когда приходилось «караулить» интересные передачи или устанавливать таймер на видеомагнитофоне для автоматической записи передачи, транслируемой в «неудобное» для зрителя время. Казалось бы, Интернет исповедует гораздо более прогрессивный подход. Или все-таки нет?!

Начнем с того, что каналы «не резиновые», их пропускная способность ограничена и все крупные серверы, раздающие медиаконтент традиционным способом, обычно очень сильно перегружены. Распределенные файло-обменные сети существенно снижают нагрузку, однако реальная скорость передачи данных у них чрезвычайно низка, да еще подолгу стоять в очередях приходится, что совсем не по-капиталистически.

Компромиссной технологией раздачи медиаконтента является онлайновое вещание по технологии Multicast, обеспечивающей одновременную доставку идентичного контента всем запросившим его пользователям, что существенно разгружает каналы передачи данных, но… ограничивает свободу пользователей в выборе контента, поскольку, если к нам подключились сто тысяч пользователей и каждый из них выбрал свой файл, то никакого выигрыша мы не получим. С другой стороны, никто нам не запрещает иметь несколько независимых Multicast-

каналов, передающих различные файлы, к которым может подключаться кто угодно. Разница между обычным скачиванием файла с сервера в том, что трансляция не позволяет слушателям/зрителям управлять потоком и они вынуждены слушать/смотреть файл с момента подключения к серверу, который к тому времени мог проиграть уже половину файла. В некоторых случаях это приемлемо, в некоторых – нет. Как показывает практика, большой аудитории пользователей совершенно не важно, что именно играет в данный момент, главное, чтобы что-то вообще играло.

К тому же в потоковое аудио/видео намного легче «врезать» рекламу или прочие вставки типа «breaking news», да и квалификация среднестатистического пользователя не позволяет сохранять потоковый контент на диске, что очень нравится держателям авторских прав и прочим медиамагнатам. Программы, сохраняющие потоковое аудио/видео либо слишком сложны в управлении (как, например, VideoLAN), либо не обеспечивают надлежащего уровня качества, в частности, большинство программ, сохраняющих видео с YouTube, страдают хронической потерей синхронизации между аудио- и видеопотоками.

Словом, существуют тысячи причин, чтобы воздвигнуть сервер потокового аудио/видео, вещающий в пределах локальной сети или даже охватывающей весь Интернет. Для потокового вещания написано огромное количество программ (и каждый день появляются все новые), но хороших из них немного, а хороших, открытых и бесплатных – еще меньше.

VideoLAN – многофункциональный комплекс, портированный практически под все операционные системы, поддерживающий множество протоколов, форматов и контейнеров, который можно использовать и как локальный аудио/видеоплейер, и как сервер трансляции (см. **рис. 1**).

Обзор основных возможностей

VideoLAN – это некоммерческий проект, который всегда можно скачать бесплатно (вместе с исходными текстами и готовыми бинарными сборками) с официального сервера http://www.videolan.org.

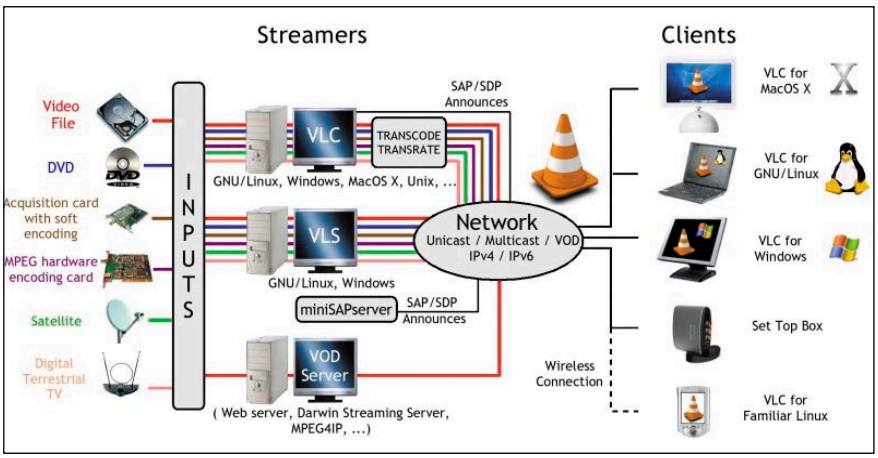


Рисунок 1. Потоковое вещание в локальных/глобальных сетях

Interfaces and control

	WxWidgets	Cocoa	WxWidgets	Native	WxWidgets	GPE
Default						
Qt 4 [11]	✓	✗	✓	-	?	✗
Skins	✓	✗	✓	✗	✓	✗
Web	✓	✓	✓	✓	✓	✓
Telnet	✓	✓	✓	✓	✓	✓
Command line	✓	✓	✓	✓	✓	✓
Infrared	✗	✗	✓	✗	✗	✗

Miscellaneous

SAP/SDP announces	✓	✓	✓	✗	✓	✗
Bonjour protocol	✗	✓	✓	?	?	?
Mozilla/Firefox plugin	✓	✓	✓	✗	✓	✗
ActiveX plugin	✓	-	-	-	-	-
SVCD Menus	◒	✗	◒	✗	◒	✗
Localization	✓	✓	✓	✓	✓	✓
CD-Text [12]	✓	✗	✓	✗	◒	-
CDDB CD info	✓	✓	✓	✗	◒	-
IGMPv3 [13]	✓	✗	✓	✗	✓	✓
IPv6 [13]	✓	✓	✓	✗	✓	✓
MLDv2 [13]	✓	✗	✓	✗	✓	✓
CPU acceleration [14]	✓	✓	✓	✓	✓ [15]	✗

Рисунок 2. Возможности программы VideoLAN на каждой из поддерживаемых ею платформ

Клиентская и серверная части исправно работают под Linux, Windows, Mac OS X, BeOS, xBSD, Solaris, Familiar Linux, Yopy/Linupy и QNX (Yopy/Liupy и Familiar Linux – это урезанные клоны Linux, предназначенные для мобильных устройств, смотреть потоковое видео на КПК, согласитесь, довольно круто), однако их функциональность различна и в зависимости от выбранной платформы варьируется в очень широких пределах (подробнее см. **рис. 2**).

Поддерживаются следующие входные форматы данных: MPEG-1, MPEG-2, MPEG-4/DivX (считываемые с локального жесткого диска или CD/DVD); «настоящие» DVD и VCD; спутниковые карты, работающие по стандарту (DVB-S); потоковое видео, «упакованное» в MPEG-1, MPEG-2 и MPEG-4 (то есть VideoLAN может работать не только как сетевой транслятор, но и как ретранслятор чужого контента, с возможностью сохранения последнего на жесткий диск).

В настоящий момент реализованы два основных протокола трансляции: Unicast («узконаправленное» вещание с доставкой контента только одному целевому узлу) и Multicast (групповая трансляция с доставкой одного и того же контента множеству узлов). Также (формально) имеется воз-

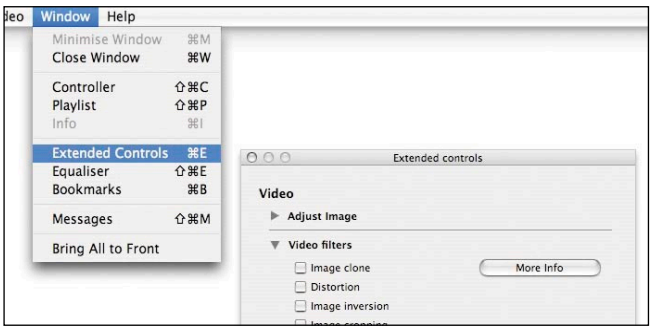


Рисунок 3. Графический интерфейс программы VideoLAN

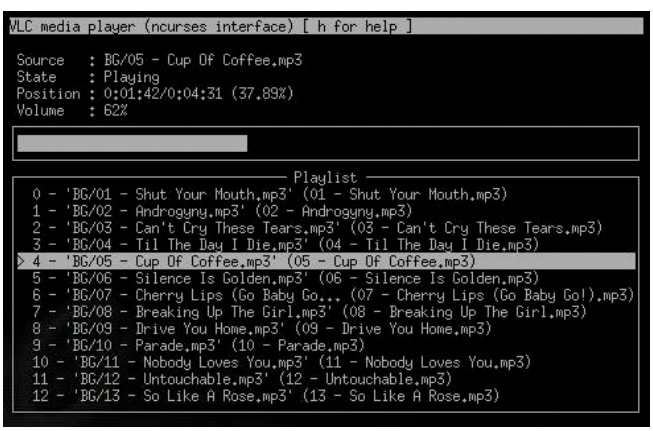


Рисунок 4. Текстовой NCURSES интерфейс

можность широковещательной рассылки контента всем узлам локальной сети (для этого достаточно указать в качестве целевого IP-адреса 255.255.255.255), но с высокой степенью вероятности она будет задавлена брандмауэрами и маршрутизаторами, так что без их радикальной перестройки сеанс вещания не состоится даже в рамках локальной сети. На некоторых узлах локальной сети сейчас стоят «агрессивные» персональные брандмауэры, отсекающие широковещательные пакеты; маршрутизаторы, объединяющие несколько сегментов локальной сети, также «не любят» широковещательные пакеты, и если автор не ошибается, то Windows XP SP2 (или Windows Server 2003 SP2) по умолчанию не выпускает широковещательные пакеты, если только их об этом не «попросить».

Еще имеется ограниченная поддержка «видео по требованию» (Video-on-Demand или, сокращенно, VoD) с возможностью выбора контента по HTTP или TELNET интерфейсам, однако эта возможность обычно используется исключительно администраторами для удаленного управления сервером трансляции.

Кстати, об интерфейсах. VideoLAN поддерживает широкий ассортимент, способный удовлетворить даже самых изысканных гурманов: GUI, NCUSERS, командная строка, HTTP/TENETLTEL и даже подключаемые модули для некоторых популярных браузеров. Контейнеры, в которые помещается транслируемый поток, зависят от типа трансляции, допустимые комбинации перечислены в **рис. 5**. Естественно, все это «хозяйство» работает как с IPv4, так и с IPv6.

Устанавливаем VideoLAN

На странице http://www.videolan.org/vlc выложены готовые бинарные сборки для следующих операционных систем:

- Windows;
- Mac OS X;
- BeOS;
- Debian GNU/Linux;
- Ubuntu Linux;
- Mandriva Linux;
- Fedora Core;
- Familiar Linux;
- SUSE Linux;
- Red Hat Linux;
- Slackware Linux;
- ALT Linux;
- YOPY/Linupy;
- Zaurus;
- Arch Linux.

Установка проходит без проблем (автор тестировал Windows, Debian, Mandriva, Fedora Core и SuSE).

Операционные системы, перечисленные ниже, формально поддерживаются, но заниматься компиляцией под них приходится самостоятельно.

- NetBSD;
- OpenBSD;
- FreeBSD;
- Solaris;
- QNX;
- Gentoo Linux;
- Crux Linux.

Поскольку VideoLAN входит в набор портов для FreeBSD (любимой BSD-системы автора!), то сборка под нее проходит без единого слова нареканий. Достаточно набрать команду:

```
# cd /usr/ports/multimedia/vlc && make install clean
```

и немного подождать. Остальные системы в этом плане несколько менее предсказуемы и способны преподнести букет неприятных сюрпризов, но всестороннее тестирование VideoLAN выходит за рамки данной статьи.

Начинаем трансляцию

Разработчики рекомендуют начинающим «телемеханикам» использовать «Мастер Трансляции» («Файл → Мастер», соответствующей горячей клавише <CTRL+W>), однако он со-

	UDP	RTP	HTTP	MMSH	File
PS	✗	✓	✓	✗	✓
TS	✓	✓	✓	✗	✓
Ogg	✗	✗	✓	✗	✓
ASF	✗	✗	✓	✓	✓
MP4	✗	✗	✗	✗	✓
QuickTime	✗	✗	✗	✗	✓
Raw	✓	✓	✓	✗	✓
MPJPEG	✗	✗	SVN only	✗	SVN only

Рисунок 5. Допустимые комбинации протоколов трансляции с контейнерами, в которых упаковывается транслируемый медиапоток

держит несколько подводных камней, кроме того, ограничивает наши возможности по изменению тонких настроек, а потому мы пойдем другим путем.

В меню «Файл» выбираем пункт «Открыть файл» (<CTRL+F>), если хотим транслировать один или несколько mp3/avi/mpeg-файлов; «Открыть каталог» (<CTRL+E>), чтобы одним махом выделить все содержащиеся в нем файлы; «Открыть диск» (<CTRL+D>) для трансляции контента непосредственно с DVD/VCD/Audio CD (правда, несмотря на все ухищрения, мне так и не удалось заставить VideoLAN транслировать музыку прямо с Audio CD без ее предварительной перекодировки); еще можно открыть URL (для ретрансляции чужого контента) или выбрать устройство типа спутниковой карты.

Начнем с простого – с обычного AVI/MPEG-файла, транслируемого по локальной сети на соседний компьютер. Жмем <CTRL+F> (см. **рис. 6**) и через «Обзор» выбираем один или несколько файлов (не обязательно одного и того же типа). Для подключения субтитров (если мы хотим их подключать) устанавливаем одноименную галочку и указываем путь к файлу с субтитрами, положение и цвет которых определяются кнопкой «Расширенные настройки». VideoLAN поддерживает множество субтитров различных типов (включая .srt и .sub), что позволяет нам, в частности, накладывать на видеопоток рекламу или различные сведения информационного характера.

Устанавливаем галочку «Вещать/Сохранить» и нажимаем «Настройки», открывающие огромный диалог (см. **рис. 7**) с кучей галочек, строк редактирования и прочих элементов управления, в которых на первых порах не так-то просто разобраться, но ведь мы же не из пугливых!

Взводим галочку «UDP» и указываем IP-адрес машины, на которую мы собираемся вещать (например, «192.168.0.6»), а также порт (по умолчанию «1234»), формат контейнера не предоставляет нам свободы выбора, т. к. UDP unicast работает только с MPEG TS (вообще-то, еще поддерживается и RAW, но в данном диалоговом окне он заблокирован, и в этом есть свой резон, поскольку RAW не лучший выбор для трансляции в реальном времени). Аудио/видео кодеки в графе «Настройки кодирования» можно выбрать на свой вкус, а можно оставить их по умолчанию (сравнение качества кодеков – тема отдельного большого разговора, традиционно сопровождаемого яростными священными войнами, которые нам ни к чему).

Время жизни пакетов (TTL) зависит от количества узлов, через которые проходит транслируемый контент, и, чтобы он не ушел чересчур далеко, это значение можно установить равным трем или даже одному. О строке «MRL выходного потока» можно не заботиться, программа сформирует ее за нас.

Остальные значения лучше пока не трогать, оставив их такими, какие они есть. Это слегка ухудшит функциональность транслятора, но на первых порах нам главное – разобраться, как его запускать!

Нажимаем <ENTER> и возвращаемся в предыдущий диалог, в котором из всех немногочисленных опций обращает на себя внимание параметр времени кэширования. В локальной сети на быстрых машинах особой разницы нет, но если возникнут проблемы и транслируемое видео

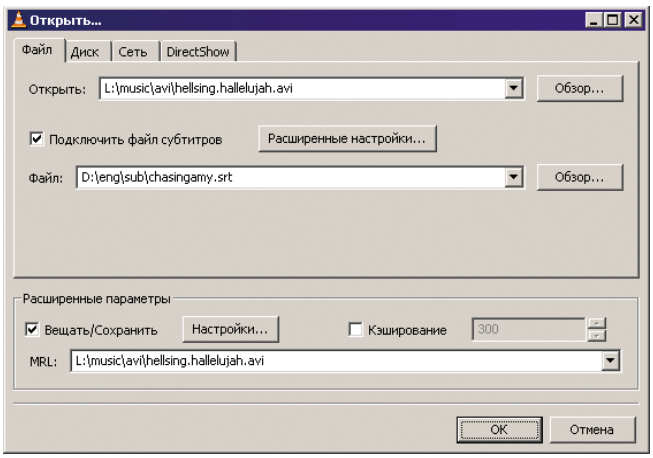


Рисунок 6. Выбор файла для трансляции в локальную/глобальную сеть

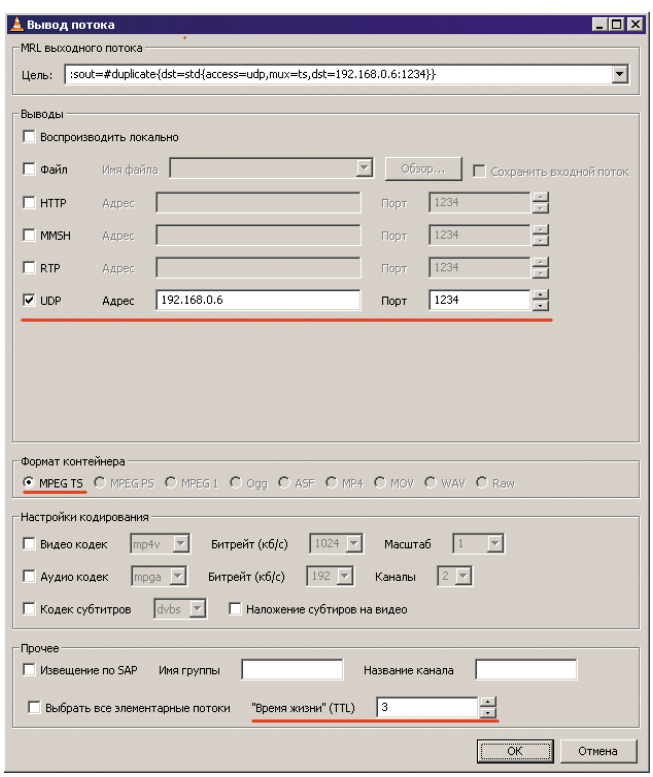


Рисунок 7. Настройки транслятора для вещания по протоколу UDP

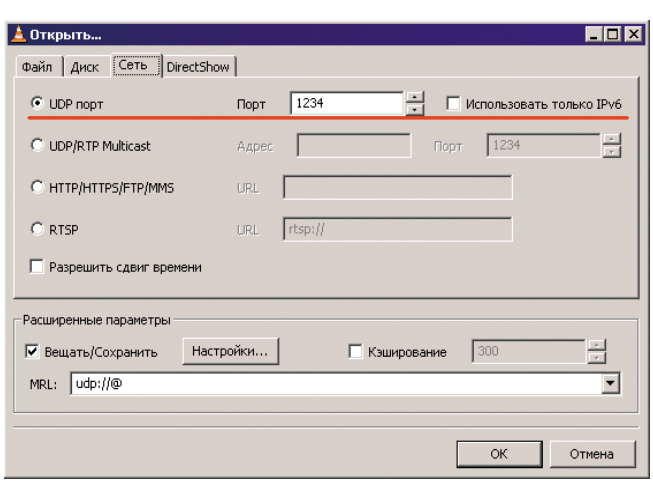


Рисунок 8. Настройки клиентского узла для приема медиапотока, транслируемого по протоколу UDP

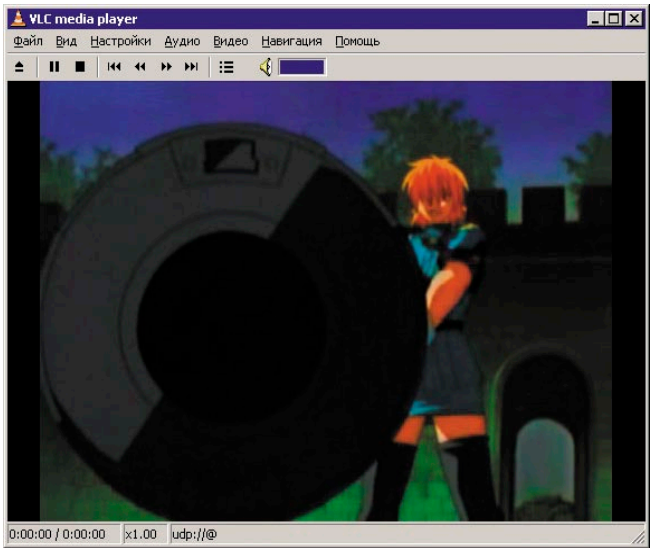


Рисунок 9. Просмотр потокового видео на клиентской стороне

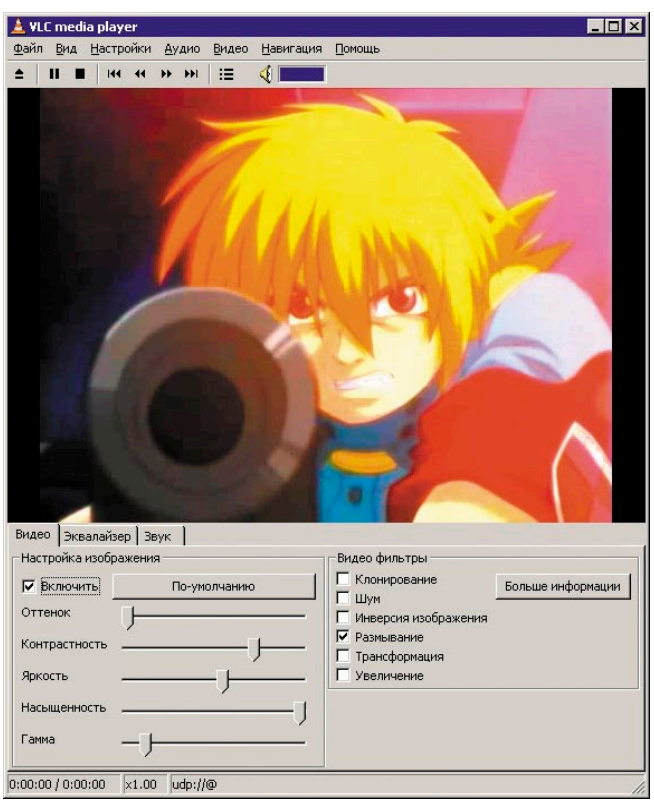


Рисунок 10. На клиентской стороне можно свободно менять основные параметры изображения (яркость, контрастность, насыщенность…), а также накладывать многочисленные фильтры

станет двигаться судорожными рывками, имеет смысл установить эту галочку.

Нажимаем «OK» еще раз, и VideoLAN начинает трансляцию на заданный узел (о чем говорит бегущая полоска, отображающая текущее положение видеофайла). Кстати говоря, UDP unicast обходит некоторые типы персональных брандмауэров типа, например, SyGate Personal Firewall/Windows Firewall, которые его не видят, а потому и не блокируют (даже если мы этого хотим).

Идем на машину с IP-адресом 192.168.0.6, запускаем на ней VideoLAN, заходим в меню «Файл», находим там пункт «Открыть URL» и в появившемся диалоговом окне (см. **рис. 8**) переводим радиокнопку в положение «UDP порт» с указанием порта-приемника, если он отличен от «1234», выбранного сервером по умолчанию. Вот и все! Нажимаем «OK», поудобнее устраиваемся в кресле и наслаждаемся транслируемым видеоконтентом (см. **рис. 9**).

При желании (на клиентской стороне) можно нажать <CTRL+G> («Настройки → Расширенный интерфейс») и проиграться яркостью, константностью, насыщенностью, эквалайзером и прочими «вкусностями» (см. **рис. 10**). На серверную сторону они не оказывают никакого влияния, и потому крутить их на узле-трансляторе никакого смысла нет.

Трансляция через WEB

Главным недостатком unicast-трансляции является невозможность вещания на произвольные узлы локальной/глобальной Сети. Сервер должен иметь список IP-узлов, для рассылки пакетов получателям же знать IP-адрес транслятора ни к чему. Им достаточно «помнить» назначенный UDP-порт, чтобы ловить трафик. Странная какая-то трансляция у нас получается… В обычной жизни все наоборот. Передатчик ничего не знает о приемнике (приемниках), а каждый из приемников в любой момент времени может настроиться на волну любого из многочисленных передатчиков и отключиться, если передача ему неинтересна.

Специально для реализации подобного способа общения VideoLAN поддерживает трансляцию через Web по TCP/IP протоколу. Возвращаясь к серверной стороне, говорим «Файл → Открыть файл → Настройки» и в уже знакомом нам диалоговом окне (см. **рис. 11**) сбрасываем (при желании) галочку «UDP» и взводим «HTTP». В поле «адрес» ничего вводить не нужно!!! Порт можно оставить в значении по умолчанию («1234») или выбрать любой другой (например, «8080», чтобы поменьше привлекать к себе внимание). Так же рекомендуется увеличить и значение TTL, особенно если мы собираемся вещать в Интернет на далекие расстояния. Строку «MRL выходного потока», как и в прошлый раз, сформирует сама программа.

Обратите внимание: сколько доступных контейнеров теперь появилось: MPEG TS/MPEG PS, MPEG 1, Ogg, ASF, MP4, MOV, WAV и RAW. Какой из них выбрать? Если все клиенты используют в качестве приемника программу VideoLAN, то особой разницы нет, и лучше оставить контейнер по умолчанию (MPEG TS), если же планируется транслировать аудио/видео-поток на компьютеры, где кроме Windows и штатного медиаплеера ничего нет, лучше выбрать ASF, однако в таком случае следует позаботиться о совместимости с кодеками, поставляемыми вместе с Windows, и в графе «видео-кодек» выбрать что-то очень хорошо известное и проверенное временем (например, DIV3, WM1, WM2), аналогичным путем поступить и со звуком, в противном случае слушателям придется рыскать в поисках нужных кодеков перед началом воспроизведения контента. Впрочем, мы не будем вдаваться в дебри совместимости различных кодеков, а поскорее нажмем на «OK».

Бегущий ползунок линейки прогресса подтверждает, что вещание началось, даже если к нам еще никто не подключен (настройки персонального брандмауэра не имеют никакого значения, поскольку VideoLAN обходит все известные автору брандмауэры).

Примечание: Windows Firewall не обращает внимания на unicast/mutlicast пакеты, а потому их не давит, в Linux/BSD с этим проблем нет, и они замечательно справляются с блокированием unicast/mulicast трафика. Касательно NAT – не все они поддерживают multicast-трансляцию и ни один – unicast (поскольку, NAT требует, чтобы узел, расположенный за ним, инициировал соединение с удаленным сервером, в противном случае NAT не будет знать, какие пакеты принадлежат этому узлу, но при unicast-трансляции по UDP все узлы-приемники ждут прихода пакетов от сервера, адреса которого они не знают, и потому инициировать соединение с ним не могут, http unicast в этом смысле намного более гибок, я спокойно использовал его через proxy-сервер, поддерживающий https (его поддерживает и VideoLAN), который с точки зрения сетевого фильтра выглядит как обычный веб-контент, который никто давить не собирается).

На клиентском узле (который может находиться где угодно) запускаем VideoLAN, говорим «Файл → Отрыть URL» и в появившемся диалоговом окне перемещаем радио-кнопку к строке HTTP/HTTPS/FTP/MMS, в графе URL указываем адрес сервера вместе с портом (например, «http://192.168.0.1:1024») и нажмем «OK».

И все это отлично работает с той лишь разницей, что теперь к серверу клиенты подключаются отовсюду, и ему совершенно не обязательно знать их IP-адреса!

Интеграция с браузерами

Скачивать/устанавливать VideoLAN или запускать штатный media-player, чтобы ввести туда адрес сервера (вместе с портом!), на это отважится далеко не каждый пользователь, особенно если media-player выдаст ошибку, потребовав установить некоторый кодек (и ведь, подлец, не скажет, какой!)

Намного удобнее внедрить видеоплейер непосредственно в HTML-страничку. И VideoLAN позволяет сделать это! Для браузеров Mozilla, FireFox и Safari выпущены специальные плагины, исходные тексты которых можно скачать непосредственно с сервера проекта VideoLAN: http://developers.videolan.org и откомпилировать их.

Каркас страницы, принимающий UDP-потоки, выглядит следующим образом (красным шрифтом выделен адрес целевого узла, на который предполагается вести трансляцию):

Листинг 1. Каркас веб-страницы для приема потокового видео по протоколу UDP

```
<html>
<head><title>Demo of VLC mozilla plugin</title></head>

<body>

<h1>Demo of VLC mozilla plugin - Example 2</h1>

<embed type="application/x-vlc-plugin"
      name="video2"
      autoplay="no" loop="no" hidden="yes"
      target="udp:@192.168.0.13 " />
<br />
  <a href="javascript:;" ↵
      onclick='document.video2.play()'>Play video2</a>
  <a href="javascript:;" ↵
      onclick='document.video2.stop()'>Stop video2</a>
  <a href="javascript:;" ↵
      onclick='document.video2.fullscreen()'>Fullscreen</a>
</body>
</html>
```

Здесь на самом деле замалчивается целый ряд вопросов. В частности, как указать адрес целевого узла.... Неужто CGI-скрипт писать, который будет каждому клиенту генерировать страничку, подставляя туда его $REMOTE_ADDR? Выходит, что так. Но как подставить адрес в поле target, пусть каждый решает сам. Пример, полностью готовый к употреблению, занимает слишком много места, чтобы быть приведенным здесь, а тема CGI не относится к данной статье.

Прием HTTP-потоков осуществляется немного иначе (полужирным шрифтом выделен адрес сервера трансляции и имя транслируемого файла):

Листинг 2. Каркас веб-страницы для приема потокового видео по протоколу HTTP

```
<html>
<head><title>Demo of VLC mozilla plugin</title></head>
```

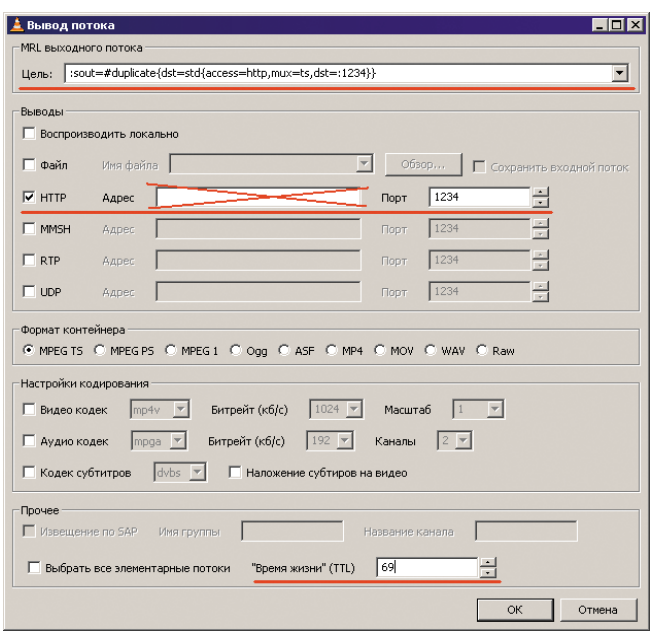


Рисунок 11. Настройки транслятора для вещания по протоколу HTTP

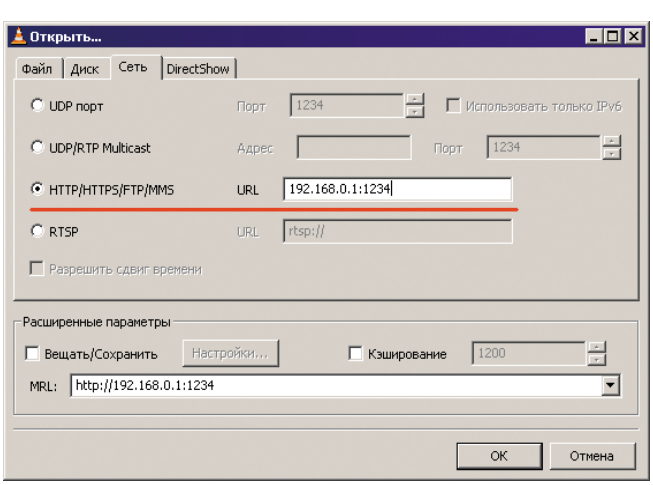


Рисунок 12. Настройки клиентского узла для приема медиапотока, транслируемого по протоколу HTTP

```
<body>

<h1>Demo of VLC mozilla plugin - Example 1</h1>

<embed type="application/x-vlc-plugin"
      name="video1"
      autoplay="no" loop="yes" width="400" height="300"
      target="http://server.example.org/video1.vob"/>
<br />
  <a href="javascript:;" ↵
      onclick='document.video1.play()'>Play video1</a>
  <a href="javascript:;" ↵
      onclick='document.video1.pause()'>Pause video1</a>
  <a href="javascript:;" ↵
      onclick='document.video1.stop()'>Stop video1</a>
  <a href="javascript:;" ↵
      onclick='document.video1.fullscreen()'>Fullscreen</a>

</body>
</html>
```

Как нетрудно видеть, в обоих случаях мы имеем дело с «видео по требованию», то есть позволяем пользователям самостоятельно выбирать, какой файл вещать (а в случае UDP-протокола – еще и куда его вещать, при работе по HTTP-протоколу указывается только имя файла, IP-адрес клиента не нужен, поскольку сам клиент должен подключиться к серверу трансляции, а для этого клиенты должны знать IP-адрес сервера).

Если подобная степень демократичности не устаивает администратора сервера, он может составить обыкновенную HTML-страничку, используя штатный ActiveX-компонент медиа-плейера, но гарантий, что ее удастся воспроизвести на произвольной машине, – никаких. Увы! Штатные Microsoft-кодеки не обеспечивает надлежащего уровня качества, а нештатные – требуют установки (с правами администратора, разумеется).

Поэтому из двух зол приходится выбирать меньшее.

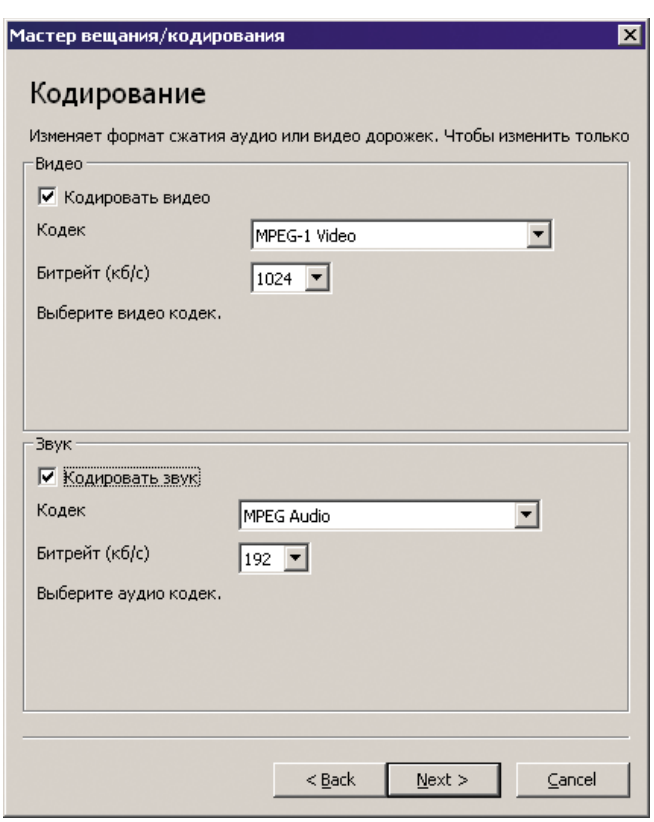


Рисунок 13. Выбор формата сохраняемого видео и аудио

Сохранение потокового контента в файл

Большинству пользователей (особенно тех, кому безлимитный Интернет всего лишь снится) не слишком нравится перспектива просмотра аудио/видео контента без возможности его сохранения на локальный диск для последующего просмотра, особенно если сервер/канал тормозит, изображение идет рывками и насладиться качеством никак не получается в силу отсутствия такового.

VideoLAN позволяет сохранять потоковый контент на диск, автоматически конвертируя его в наиболее предпочтительный формат. Удобнее всего это делать через «Мастера» («Файл → Мастер» или <CTRL+W>). В появившемся диалоговом окне переводим радиокнопку в положение «Кодировать/Сохранить в файл» и говорим «Next». Радиокнопку «Выберите поток» оставляем в значении по умолчанию и нажимаем «Выберите», где выбираем вкладку «Сеть», а в ней протокол (и порт!), на который осуществляется вещание (транслятором может выступать как сам VideoLAN, запущенный на соседней машине, так и посторонний сервер).

Следующее диалоговое окно (см. **рис. 13**) предлагает нам выбрать формат сжатия аудио и видео. MPEG1, выставленный по умолчанию, – наилучший выбор с точки зрения качества, но, если мы не хотим жертвовать размером (а у MPEG1 размер ого-го-го), лучше остановиться на MPEG4 или на других, более продвинутых форматах, не забывая, впрочем, о совместимости с популярными кодеками.

Наконец, нас спрашивают о формате контейнера. Контейнер – это то, куда ложится видеопоток, перемеженный с аудиопотоком. Наилучший выбор с точки зрения совместимости – это либо ASF, либо MPEG1. Все остальные контейнеры хоть и обладают определенными преимуществами, но требуют обязательной установки дополнительного программного обеспечения (впрочем, к VideoLAN это не относится, так как он поддерживает их всех).

Все! Последний «Next» выносит нас к строке ввода, запрашивающей имя файла, в который следует толкать весь принимаемый контент, и после нажатия на «Finish» начинается процесс грабежа (сохранение потокового контента может быть признано нелегальным, так что поосторожнее с награбленным!).

Заключение

На этом возможности программного комплекса VideoLAN не заканчиваются, а только-только начинают раскрываться. Для работы с ним необходимо не только помнить расположение пунктов меню, но и основательно разбираться в тонкостях многочисленных алгоритмов сжатия, знать, чем отличается один формат контейнера от другого, словом, быть настоящим профессионалом своего дела.

Как ни крути, а методом тыка такие вещи не осваиваются, и воздвигнутые сервера трансляции либо оказываются недостаточно производительными, либо страдают хронической несовместимостью с клиентскими машинами, однако, поскольку VideoLAN включает в себя как серверную, так и клиентскую части, да и к тому же портирован под все платформы, наилучшим решением будет использование этой программы как на стороне транслятора, так и на стороне приемника.