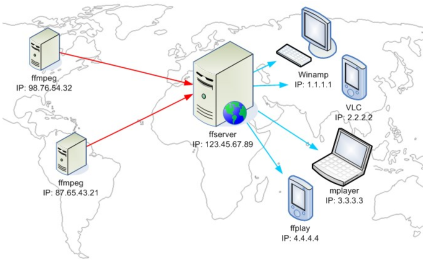
Warning: ffserver has been removed on 2018-01-06. If you still need it checkout commit 2ca65fc or use the 3.4 release branch. The original documentation has been archived and can be downloaded as HTML or PDF while the sample ffserver configuration file can be found below. We can provide no support for ffserver.

Or try an alternative such as mkvserver_mk2.

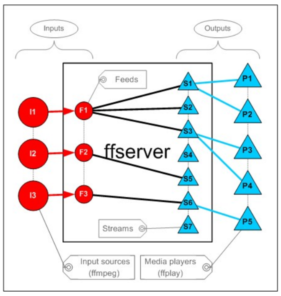
Introduction

If you need to stream your audio/video content over the internet, you'll usually need a streaming (broadcasting) server, one of which is ffserver. It is able to collect multiple input sources (usually ffmpeg applications) and transcode/remux/broadcast each of them using multiple output streams. The simple diagram is shown on the image below:



Various input sources (ffmpeg applications) can be used to "feed" the broadcasting server (ffserver) with multimedia content that will be distributed to multiple clients for viewing. The purpose of the above image is to visually show the ability to separate parts of your streaming system into pieces that can be deployed around the world, allowing you to broadcast various live events without the need to change the structure of your streaming media system.

Let's take a closer look of ffserver, to better describe its possibilities. Consider the following image:



There are several elements shown on the image. Let's name them all first:

- Input sources (I)
- Feeds (F)
- Streams (S)
- Media players (P)

Input sources

These elements are not part of internal structure of ffserver tool, but rather represent external applications (usually ffmpeg), which can send audio/video streams to ffserver that will be distributed (broadcast) to all the viewers (media players). Since ffmpeg is mostly used as an input source, we'll describe it here in this document.

Input sources will connect to ffserver and bind themselves with one or more feeds if those feeds are not bound with some other input source at that moment. Binding one input source to multiple feeds is possible and makes sense only if the input source can produce different stream input for each feed it is bound to. It's useless for the input source to provide the same stream input to several feeds, since ffserver already has got a way of associating a single feed to multiple output streams.

Feeds

Feed element is an internal part of ffserver which has a purpose to associate one input source with one or more output streams. The possibility to associate a feed with more output streams is useful when you want to stream one input source (for example, your webcam with audio) using several different output formats (for example, streaming a full HD video and a small-size preview video for mobile phones) at the same time. Shortly speaking, each feed element logically represents each of your input sources. It can be considered as an "input jack" of ffserver, to which you connect your audio/video sources.

Streams

A stream element is internal part of ffserver and represents a connection point for all your viewers who wish to get a specific stream. For example, if you want to stream one full HD video and a small-size preview video for mobile phones, you will create one feed element (to connect your input to) and associate it with two stream elements (which will define different frame size, encoding type and/or output format). Each stream element can handle multiple connecting clients, just like one web server can handle multiple web clients. It can be considered as an "output jack" of ffserver, to which your viewers (media players) can connect to view your audio/video stream. The obvious difference between a feed element and a stream element (between input/output jack) is that a single stream element can handle multiple connections with viewers, while a single feed element is always connected to only one input source.

Media players

Media player elements are not internal part of ffserver. They just represent your viewers from the "outside world" that are connecting to the various stream elements to view your multimedia content. Some of the popular media players are: ffplay, VLC, mpv, Windows Media Player, etc.

Running ffserver

To be able to successfully start ffserver, you'll need a valid configuration file first. Once you create a valid config file, you can start ffserver simply by running the following command:

```
ffserver -f /etc/ffserver.conf
```

Depending on your configuration file, your ffserver will start or not :) But more often it will not start until you debug all the issues that usually occur, including syntax errors, so you'll most probably want to run your ffserver in debug mode with "-d" option, until you sort out everything, like this:

```
ffserver -d -f /etc/ffserver.conf
```

You can always get a full list of options with:

```
ffserver --help
```

When you finally build a valid configuration file, you'll want to run your ffserver in the background (as a daemon), which can be accomplished using either a trailing ampersand character (&) in a shell command or more conveniently you can comment out "`NoDaemon`" directive inside your config file (works on Windows too).

Connecting your input sources

Once your ffserver is up and running, it's time to connect input sources to it. Without input sources, your ffserver is not going to broadcast anything to the outside world and will be pretty much useless. So, let's see how we can connect input sources to ffserver. The simplest way is to use the ffmpeg tool and the general syntax for such command is:

```
ffmpeg <inputs> <feed URL>
```

Of course, if you want to use one input source (ffmpeg) and bind it to multiple feeds (if you like to have only one application started), you might use:

```
ffmpeg <inputs> <feed URL> <more inputs> <another feed URL> <even more inputs> <yet another feed URL>
```

but, keep in mind that, if that input source crashes, all its bound feeds will become unavailable. So it's a good practice to use one input source (ffmpeg) pear each feed (1-1).

The parameter "`<feed URL>`" has got the following form:

```
http://<ffserver_ip_address_or_host_name>:<ffserver_port>/<feed_name>
```

All these things are defined in your ffserver configuration file:

- **`<ffserver_ip_address_or_host_name>`** - using the "`BindAddress`" directive
- **`<ffserver_port>`** - using the "`Port`" directive
- **`<feed_name>`** - using the "`<Feed>`" block

Let's assume that we want to stream our webcam video + audio to our friends. We will simply run an ffmpeg command line that will capture our webcam video and audio input and forward it to ffserver. The command line will look something like this:

```
ffmpeg \
 -f v4l2 -s 320x240 -r 25 -i /dev/video0 \
 -f alsa -ac 1 -i hw:0 \
 http://localhost:8090/feed1.ffm
```

This is the same thing as this:

```
ffmpeg -f v4l2 -s 320x240 -r 25 -i /dev/video0 -f alsa -ac 1 -i hw:0 http://localhost:8090/feed1.ffm
```

but it looks better and makes it more clear to understand each part of the command line.

- The first part "**`-f v4l2 -s 320x240 -r 25 -i /dev/video0`**" represents the first input for ffmpeg and captures our webcam video. For more info, you can read more about How to capture a webcam input.
- The second part "**`-f alsa -ac 1 -i hw:0`**" represents the second input for ffmpeg and captures our audio, depending on our system audio configuration. For more info, you can read more about Capturing audio with FFmpeg and ALSA.
- The last, but not the least important, part "**`http://localhost:8090/feed1.ffm`**" represents the feed URL, which tells ffmpeg to connect to ffserver and send it the audio + video streams for broadcast. In this example we used the hostname "localhost" which means that everything is running on our computer, but if you need to feed the live online ffserver, you'll need to change the "localhost" to the real host name or IP address of your ffserver computer. Also make sure that your feed name ends with "`.ffm`" and if it's not the case, then add "`-f ffm`" before your feed URL, to manually specify the output format (because ffmpeg won't be able to figure it out automatically any more), like this "**`-f ffm http://localhost:8090/blah.bleh`**".

As soon as you type the command above, you should see ffmpeg displaying some statistics about your input streams and counting output frames, which is a pretty good sign that everything works (so far).

For this example, you would need at least the following things defined in your config file (three dots "..." represent the other data that is irrelevant for this topic):

```
Port 8090
BindAddress 0.0.0.0

...

<Feed feed1.ffm>

   .  .  .

</Feed>

...
```

Viewing your streams

If you've done all the steps so far without errors, you're now ready to view your streams. The simplest way to do so is to use ffplay to connect to ffserver and view a specific stream. The general syntax for such command is:

```
ffplay <stream URL>
```

The parameter "`<stream URL>`" has got the following form:

```
http://<ffserver_ip_address_or_host_name>:<ffserver_port>/<stream_name>
```

All these things are defined in your ffserver configuration file:

- **`<ffserver_ip_address_or_host_name>`** - using the "`BindAddress`" directive
- **`<ffserver_port>`** - using the "`Port`" directive
- **`<stream_name>`** - using the "`<Stream>`" block

For example if you have appropriate stream element defined in your ffserver configuration file, you could type:

```
ffplay http://localhost:8090/test1.mpg
```

and your stream should appear (depending on the encoding used and caching enforced) relatively shortly in a matter of seconds. In this example we used the host name "localhost" which means that everything is running on our computer, but if you need to view streams from the live online ffserver, you'll need to change the "localhost" to the real host name or IP address of ffserver computer.

For this example, you would need at least the following things defined in your config file (three dots "..." represent the other data that is irrelevant for this topic):

```
Port 8090
BindAddress 0.0.0.0

...

<Stream test1.mpg>

   .  .  .

</Stream>
```

```
...
```

Creating the configuration file

It would be very wise to start off reading the sample configuration file below. It is self-documented with a lot of comments and it is a good starting point for beginners, since it contains various examples too. Also, refer to `man ffserver` and `ffserver -h` or download the archived documentation: HTML or PDF. In general, the configuration file is consisted of global directives, list of feed elements, list of stream elements and a specification of a special status stream element, which is used to provide a way for you to view the status of all your running streams.

Examples of configuration files

Sample ffserver configuration file

```
# Port on which the server is listening. You must select a different
# port from your standard HTTP web server if it is running on the same
# computer.
Port 8090

# Address on which the server is bound. Only useful if you have
# several network interfaces.
BindAddress 0.0.0.0

# Number of simultaneous HTTP connections that can be handled. It has
# to be defined *before* the MaxClients parameter, since it defines the
# MaxClients maximum limit.
MaxHTTPConnections 2000

# Number of simultaneous requests that can be handled. Since FFServer
# is very fast, it is more likely that you will want to leave this high
# and use MaxBandwidth, below.
MaxClients 1000

# This the maximum amount of kbit/sec that you are prepared to
# consume when streaming to clients.
MaxBandwidth 1000

# Access log file (uses standard Apache log file format)
# '-' is the standard output.
CustomLog -

# Suppress that if you want to launch ffserver as a daemon.
NoDaemon


##################################################################
# Definition of the live feeds. Each live feed contains one video
# and/or audio sequence coming from an ffmpeg encoder or another
# ffserver. This sequence may be encoded simultaneously with several
# codecs at several resolutions.

<Feed feed1.ffm>

# You must use 'ffmpeg' to send a live feed to ffserver. In this
# example, you can type:
#
# ffmpeg http://localhost:8090/feed1.ffm

# ffserver can also do time shifting. It means that it can stream any
# previously recorded live stream. The request should contain:
# "http://xxxx?date=[YYYY-MM-DDT][[HH:]MM:]SS[.m...]".You must specify
# a path where the feed is stored on disk. You also specify the
# maximum size of the feed, where zero means unlimited. Default:
# File=/tmp/feed_name.ffm FileMaxSize=5M
File /tmp/feed1.ffm
FileMaxSize 200K

# You could specify
# ReadOnlyFile /saved/specialvideo.ffm
# This marks the file as readonly and it will not be deleted or updated.

# Specify launch in order to start ffmpeg automatically.
# First ffmpeg must be defined with an appropriate path if needed,
# after that options can follow, but avoid adding the http:// field
#Launch ffmpeg

# Only allow connections from localhost to the feed.
ACL allow 127.0.0.1

</Feed>


##################################################################
# Now you can define each stream which will be generated from the
# original audio and video stream. Each format has a filename (here
# 'test1.mpg'). FFServer will send this stream when answering a
# request containing this filename.

<Stream test1.mpg>

# coming from live feed 'feed1'
Feed feed1.ffm

# Format of the stream : you can choose among:
# mpeg       : MPEG-1 multiplexed video and audio
# mpegvideo  : only MPEG-1 video
# mp2        : MPEG-2 audio (use AudioCodec to select layer 2 and 3 codec)
# ogg        : Ogg format (Vorbis audio codec)
# rm         : RealNetworks-compatible stream. Multiplexed audio and video.
# ra         : RealNetworks-compatible stream. Audio only.
# mpjpeg     : Multipart JPEG (works with Netscape without any plugin)
# jpeg       : Generate a single JPEG image.
# asf        : ASF compatible streaming (Windows Media Player format).
# swf        : Macromedia Flash compatible stream
# avi        : AVI format (MPEG-4 video, MPEG audio sound)
Format mpeg

# Bitrate for the audio stream. Codecs usually support only a few
# different bitrates.
AudioBitRate 32

# Number of audio channels: 1 = mono, 2 = stereo
AudioChannels 1

# Sampling frequency for audio. When using low bitrates, you should
# lower this frequency to 22050 or 11025. The supported frequencies
# depend on the selected audio codec.
AudioSampleRate 44100

# Bitrate for the video stream
VideoBitRate 64

# Ratecontrol buffer size
VideoBufferSize 40

# Number of frames per second
VideoFrameRate 3

# Size of the video frame: WxH (default: 160x128)
# The following abbreviations are defined: sqcif, qcif, cif, 4cif, qqvga,
# qvga, vga, svga, xga, uxga, qxga, sxga, qsxga, hsxga, wvga, wxga, wsxga,
# wuxga, woxga, wqsxga, wquxga, whsxga, whuxga, cga, ega, hd480, hd720,
# hd1080
VideoSize 160x128

# Transmit only intra frames (useful for low bitrates, but kills frame rate).
#VideoIntraOnly

# If non-intra only, an intra frame is transmitted every VideoGopSize
# frames. Video synchronization can only begin at an intra frame.
VideoGopSize 12

# More MPEG-4 parameters
# VideoHighQuality
# Video4MotionVector

# Choose your codecs:
#AudioCodec mp2
#VideoCodec mpeg1video

# Suppress audio
#NoAudio

# Suppress video
#NoVideo

#VideoQMin 3
#VideoQMax 31

# Set this to the number of seconds backwards in time to start. Note that
# most players will buffer 5-10 seconds of video, and also you need to allow
# for a keyframe to appear in the data stream.
#Preroll 15

# ACL:

# You can allow ranges of addresses (or single addresses)
#ACL ALLOW <first address>

# You can deny ranges of addresses (or single addresses)
#ACL DENY <first address>

# You can repeat the ACL allow/deny as often as you like. It is on a per
# stream basis. The first match defines the action. If there are no matches,
# then the default is the inverse of the last ACL statement.
#
# Thus 'ACL allow localhost' only allows access from localhost.
# 'ACL deny 1.0.0.0 1.255.255.255' would deny the whole of network 1 and
# allow everybody else.

</Stream>


##################################################################
# Example streams


# Multipart JPEG

#<Stream test.mjpg>
#Feed feed1.ffm
#Format mpjpeg
#VideoFrameRate 2
#VideoIntraOnly
#NoAudio
#Strict -1
#</Stream>


# Single JPEG

#<Stream test.jpg>
#Feed feed1.ffm
#Format jpeg
#VideoFrameRate 2
#VideoIntraOnly
##VideoSize 352x240
#NoAudio
#Strict -1
#</Stream>


# Flash

#<Stream test.swf>
#Feed feed1.ffm
#Format swf
#VideoFrameRate 2
#VideoIntraOnly
#NoAudio
#</Stream>


# ASF compatible

<Stream test.asf>
Feed feed1.ffm
```

```
Format asf
VideoFrameRate 15
VideoSize 352x240
VideoBitRate 256
VideoBufferSize 40
VideoGopSize 30
AudioBitRate 64
StartSendOnKey
</Stream>


# MP3 audio

#<Stream test.mp3>
#Feed feed1.ffm
#Format mp2
#AudioCodec mp3
#AudioBitRate 64
#AudioChannels 1
#AudioSampleRate 44100
#NoVideo
#</Stream>


# Ogg Vorbis audio

#<Stream test.ogg>
#Feed feed1.ffm
#Title "Stream title"
#AudioBitRate 64
#AudioChannels 2
#AudioSampleRate 44100
#NoVideo
#</Stream>


# Real with audio only at 32 kbits

#<Stream test.ra>
#Feed feed1.ffm
#Format rm
#AudioBitRate 32
#NoVideo
#NoAudio
#</Stream>


# Real with audio and video at 64 kbits

#<Stream test.rm>
#Feed feed1.ffm
#Format rm
#AudioBitRate 32
#VideoBitRate 128
#VideoFrameRate 25
#VideoGopSize 25
#NoAudio
#</Stream>


##################################################################
# A stream coming from a file: you only need to set the input
# filename and optionally a new format. Supported conversions:
#    AVI -> ASF

#<Stream file.rm>
#File "/usr/local/httpd/htdocs/tlive.rm"
#NoAudio
#</Stream>

#<Stream file.asf>
#File "/usr/local/httpd/htdocs/test.asf"
#NoAudio
#Author "Me"
#Copyright "Super MegaCorp"
#Title "Test stream from disk"
#Comment "Test comment"
#</Stream>


##################################################################
# RTSP examples
#
# You can access this stream with the RTSP URL:
#   rtsp://localhost:5454/test1-rtsp.mpg
#
# A non-standard RTSP redirector is also created. Its URL is:
#   http://localhost:8090/test1-rtsp.rtsp

#<Stream test1-rtsp.mpg>
#Format rtp
#File "/usr/local/httpd/htdocs/test1.mpg"
#</Stream>


# Transcode an incoming live feed to another live feed,
# using libx264 and video presets

#<Stream live.h264>
#Format rtp
#Feed feed1.ffm
#VideoCodec libx264
#VideoFrameRate 24
#VideoBitRate 100
#VideoSize 480x272
#AVPresetVideo default
#AVPresetVideo baseline
#AVOptionVideo flags +global_header
#
#AudioCodec libfaac
#AudioBitRate 32
#AudioChannels 2
#AudioSampleRate 22050
#AVOptionAudio flags +global_header
#</Stream>

##################################################################
# SDP/multicast examples
#
# If you want to send your stream in multicast, you must set the
# multicast address with MulticastAddress. The port and the TTL can
# also be set.
#
# An SDP file is automatically generated by ffserver by adding the
# 'sdp' extension to the stream name (here
# http://localhost:8090/test1-sdp.sdp). You should usually give this
# file to your player to play the stream.
#
# The 'NoLoop' option can be used to avoid looping when the stream is
# terminated.

#<Stream test1-sdp.mpg>
#Format rtp
#File "/usr/local/httpd/htdocs/test1.mpg"
#MulticastAddress 224.124.0.1
#MulticastPort 5000
#MulticastTTL 16
#NoLoop
#</Stream>


##################################################################
# Special streams

# Server status

<Stream stat.html>
Format status

# Only allow local people to get the status
ACL allow localhost
ACL allow 192.168.0.0 192.168.255.255

#FaviconURL http://pond1.gladstonefamily.net:8080/favicon.ico
</Stream>


# Redirect index.html to the appropriate site

<Redirect index.html>
URL http://www.ffmpeg.org/
</Redirect>
```

Streaming H.264 video with AAC audio in FLV format

```
Port 8090
BindAddress 0.0.0.0
MaxHTTPConnections 2000
MaxClients 1000
MaxBandwidth 1000
CustomLog -
#NoDaemon

<Feed feed1.ffm>
 File /tmp/feed1.ffm
 FileMaxSize 200K
 ACL allow 127.0.0.1
</Feed>

# if you want to use mpegts format instead of flv
# then change "live.flv" to "live.ts"
# and also change "Format flv" to "Format mpegts"
<Stream live.flv>
 Format flv
 Feed feed1.ffm

 VideoCodec libx264
 VideoFrameRate 30
 VideoBitRate 512
 VideoSize 320x240
 AVOptionVideo crf 23
 AVOptionVideo preset medium
 # for more info on crf/preset options, type: x264 --help
 AVOptionVideo flags +global_header

 AudioCodec aac
 Strict -2
 AudioBitRate 128
 AudioChannels 2
 AudioSampleRate 44100
 AVOptionAudio flags +global_header
</Stream>

##################################################################
# Special streams
##################################################################
<Stream stat.html>
 Format status
 # Only allow local people to get the status
 ACL allow localhost
 ACL allow 192.168.0.0 192.168.255.255
</Stream>

# Redirect index.html to the appropriate site
<Redirect index.html>
 URL http://www.ffmpeg.org/
</Redirect>
##################################################################
```

Streaming Theora video with Vorbis audio in Ogg format

```
Port 8090
BindAddress 0.0.0.0
MaxHTTPConnections 2000
MaxClients 1000
```

```
MaxBandwidth 1000
CustomLog -
#NoDaemon

<Feed feed1.ffm>
 File /tmp/feed1.ffm
 FileMaxSize 200K
 ACL allow 127.0.0.1
</Feed>

<Stream live.ogg>
 Format ogg
 Feed feed1.ffm

 VideoCodec libtheora
 VideoFrameRate 24
 VideoBitRate 512
 VideoSize 320x240
 VideoQMin 1
 VideoQMax 31
 VideoGopSize 12
 Preroll 0
 AVOptionVideo flags +global_header

 AudioCodec libvorbis
 AudioBitRate 64
 AudioChannels 2
 AudioSampleRate 44100
 AVOptionAudio flags +global_header
</Stream>

##################################################################
# Special streams
##################################################################
<Stream stat.html>
 Format status
 # Only allow local people to get the status
 ACL allow localhost
 ACL allow 192.168.0.0 192.168.255.255
</Stream>

# Redirect index.html to the appropriate site
<Redirect index.html>
 URL http://www.ffmpeg.org/
</Redirect>
##################################################################
```

Last modified 3年 ago

- Tags
- ffserver

附件 (4)

- ffserver_map.png (86.7 KB) - added by burek 10年 ago.
- ffserver.png (30.5 KB) - added by burek 10年 ago.
- ffserver.html (55.0 KB) - added by llogan 3年 ago. "ffserver documentation from FFmpeg 3.4.5"
- ffserver.pdf (162.7 KB) - added by llogan 3年 ago. "ffserver documentation from FFmpeg 3.4.5 (PDF)"

Download all attachments as: .zip

Note: See TracWiki for help on using the wiki.

用其他格式下载:

纯文本

Powered by **Trac 1.4.2**
By Edgewall Software .

请访问Trac开源项目
http://trac.edgewall.org/