Table of Contents

Stream a video or screen to another computer using netcat

There are a couple ways to use netcat with this. For example: server just sends, client receives when it wants to - very much like udp, or, server is on standby until receiver starts the connection.

By server, I'm referring to the computer that will do the transmitting, and the client is the computer that will receive the stream.

Server

First, the server. The ffmpeg command depends on what you want to do. Again, there are 3 parts to the ffmpeg command

1. The input - ie: do you want to tx a video, do a screen capture with x11grab, so on
2. Conversion into a stream
3. Output and piping to the other system

Example

For example, to send a screen capture with x11grab:

```
ffmpeg -video_size 1280x720 -framerate 30 -f x11grab -i :0.0+0,0 -c:v mjpeg -r 30 -q:v 8 -f matroska - | nc -l -p 9000
```

Input

Screen capture with x11grab

For this command, the input is a 720p capture from 0,0 on the screen to 1280×720. You can change the +0,0 to change the top left corner, and -video_size is the resolution to be captures.

If you want to capture a 1080p screen, you can make it 1920×1080, and add a -s 1280×720 in the command for example to downsample it if you need to.

Note on x11grab and other methods

X11 capture is kinda terrible to be honest, anything above 1080p starts dropping frames (unrelated to the streaming stuff - x11 capture on OBS is equally terrible).

And I don't mean dropping frames in the video, x11 capture at 4k lags the system mouse and any application…

If you have a GPU, you can use -f kmsgrab to use the GPU to do a capture, and encode with vaapi. This is fine for recording and can even do 4k60, but sucks if you want to stream it to another computer. (you need to do sudo setcap cap_sys_admin+ep /usr/bin/ffmpeg first to use kmsgrab if you want to play with it - see FFMPEG Tricks)

With video instead of screen cap

Finally, you can also just transmit a video with something like ffmpeg -re -i your-video.webm as your input. Note that you might need the -re to parse the input in real time or else it'll go at like 100x speed. Also, you can add -an to remove the audio track.

Transcoding

with mjpeg

Next, in this case, I'm sending video only and just converting to mjpeg. This has the advantage that mjpeg uses intra-frame compression, so it has very low latency.

Also, it can be played on a potato like the pentium laptop I was transmitting to... Obviously, mjpeg is severely lacking in the efficiency department though.

If you try to get high quality with something like -q:v 3, anything above 720p will use >100 Mbit/s, which my laptop with the 100 Mbit NIC can't keep up with. Anything below -q:v 10 will start looking compressed. I guess start with -q:v 8.

Other codecs

I also tried this with mpeg2video (used in DVDs and digital TV), and it also needs ~ -q:v 8 to start looking usable, but since it has inter-frame compression, it starts to add latency. It can still be kept short enough.

OGG Theora, VP8/VP9 are all out of the question - too much latency for this use. Libx264 can keep up with -preset ultrafast, but you'll need like -g 1 to force it to intra frame.

Transmitting

Ffmpeg output to matroska

Finally, the output. I've tested -f matroska and ~~-f mpegts~~ (update: matroska only. mpegts seems broken) to work), as those seem to support streams in packets. I think mpegts was a bit faster, but matroska is more flexible, esp with variable frame rate and multiple tracks.

Netcat

I pipe this to netcat which is set to listen on port 9000. Note that I have the server listen, so ffmpeg won't actually start transcoding until the client connects. -p is the port, I'm just using 9000 for testing.

Note, you'll need to allow outgoing connections from that port on your firewall. Eg: ufw allow 9000 if you use ufw.

Another example

Here is a similar example sending a video with mpeg2 and no audio at 60fps.

```
ffmpeg -re -i video.mkv -c:v mpeg2video -s 1920x1080 -r 60 -an -f mpegts - |
nc -l -p 9000
```

Client

Next, the client. This is much simpler. I used to use mplayer with -benchmark, but mpv seems the better choice these days.

Example

The base command I have noted down is

```
nc 127.0.0.1 9000 | mpv --no-cache --untimed --no-demuxer-thread  -
```

Netcat

Basically, netcat will connect to 127.0.0.1 (replace this with your server IP) on port 9000 and just pipe out what it receives.

If it fails to connect, try just doing echo “hello” | nc … on the server and see if the nc on the client can print that for debugging.

As mentioned, you'll need to allow outgoing from that port on the server., and remember to replace 127.0.0.1 with the server LAN IP. (eg 10.11.12.1 or 192.168.0.100, etc)

MPV

Next, the output of this is piped to MPV. –no-cache –untimed and –no-demuxer-thread reduced the

latency a great deal for me (but might be more unreliable).

Especially since client buffering is usually ~2s, which ruins any optimization done on the server such as special codecs... I'm sure there's a VLC equivalent, I haven't discovered it yet.

Conclutsion

I thinks that's about it for the base server/client setup. I'm sure there's a more optimal and better way.

Note on Netcat

BTW, that netcat trick can be used to pipe any command should it be needed - I've done it with tar at least...

Note that there are 2 versions of netcat, GNU netcat and OpenBSD. I'm pretty sure I used GNU netcat since it's the default on Linux, but might be worth checking.