Table of Contents

Desktop Experience

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.

PulseAudio modules

Just making this short section to put a reference to the documentation for all modules. It lists all parameters for the different modules and might be of use.

https://www.freedesktop.org/wiki/Software/PulseAudio/Documentation/User/Modules/

Split output on sound card (PulseAudio)

I have a surround sound card in my system, currently an old Soundblaster Live connected over PCI. It has 3 line outputs, and 1 input.

By default, if set to Stereo Duplex in PulseAudio, it only uses the main audio output, the green front headphone/line output.

I wanted to be able to plug in both my headphones and speakers in separately, without having to use a splitter that adds extra noise.

Solution 1: Just set the configuration in PulseAudio to 4.0 Surround + Stereo Input. This will enable 4 channel sound, so it will use the green front output for the front 2 channels, and the black rear speaker out for the rear 2 channels. PulseAudio will detect the incoming stream as Stereo, and upmix it to all 4 channels. Then plugging in your headphones to the front output (still on the back of your computer on the sound card, it's just labelled the front output, it's the green one), and the speakers into the rear output (again, in the sound card, it's the black one usually) you can get sound from both of them.

Now, while this works, all programs get upmixed to both outputs. You do get control over volume by setting the output channels separately in either PulseAudio, or preferrably in alsamixer.

We can do better. An option is to make 2 dummy outputs, say called “Headphones” and “Speakers” that programs can output to in pulse, and then map those 2 virtual sinks into the front and rear channels of the 4.0 output.

First, get the alsqa name for your sound card. It's the one labelled alsa_output.pci-0000 so on in my example below. You can get it by just looking at the PulseAudio sink list.

```
pactl list sinks
```

Right under the sink number, it will be the one labelled “Name:”. Copy this value, to add to PulseAudio configuration.

```
sudo vim /etc/pulse/default.pa
```

And add the 2 following lines in the bottom of the file, based on your configuration. You can also do this with pactl load-module module-remap-sink to test it. Adding it do default.pa will run it each time PulseAudio starts.

```
### Split sound card
load-module module-remap-sink sink_name=speakers
sink_properties="device.description='Speakers'" remix=no
master=alsa_output.pci-0000_09_03.0.analog-surround-40 channels=2
master_channel_map=front-left,front-right channel_map=front-left,front-right
load-module module-remap-sink sink_name=headphones
sink_properties="device.description='Headphones'" remix=no
master=alsa_output.pci-0000_09_03.0.analog-surround-40 channels=2
master_channel_map=rear-left,rear-right channel_map=front-left,front-right
```

This will create 2 sinks called speakers and headphones (the name is internal, it doesn't matter, the description is what you see). We then disable remixing, and give it the sound card as the output. We want each output to send 2 channels to the sound card, in this case, we want to map the front-left and front-right from our speakers virtual sink to the front-left and front-right on the sound card.

Similarly, we want to map the front-right and front-left of our headphones virtual sink (the virtual sinks are only 2 channels as mentioned) to the rear-left and rear-right on the sound card, which corresponds to the 2nd output on the card as decribed earlier. master_channel_map is the sound card output, channel_map is the virtual sink output.

Make sure you change the sound card name for your card!

You can do the same think with a 3rd virtual sink to set the card in 5.1 mode and use the 3rd output if you need to as well.

Now, when you open pavucontrol, just set programs to use the “Headphones” or “Speakers” sink to correspond to that output. This basically makes your sound card behave as 2 outputs. If you want a program to play on both speakers and headphones, send it to the sound card sink instead of the virtual sinks. By using 2 new sinks, we also get independant volume control over both (though both are capped by the volume on the main sound card sink).

By the way, this same thing can work backwards. You can create a 7.1 surround sound virtual sink (8 channels), and map the individual channel to MULTIPLE sound cards. In this case, your master will be the sound card you want to send those 2 channels, then set the master map to front-left and front-right of the card, and channel map to the channel from the virtual sink to map, eg side-left and side-right.

This really makes me miss how simple it was to reroute audio in JACK... I guess this will have to do with PulseAudio for now.

Microphone Loopback

If you're recording something and want to be able to hear something, then you can load the pulseaudio loopback module.

To load the module and start loopback:

```
pactl load-module module-loopback latency_msec=1
```

To unload and stop playback:

```
pactl unload-module module-loopback
```

If it's looping the wrong device, you can change it using the 'pavucontrol' program.

Microphone noise reduction

You can use the echo-cancel module to some success. RMNoise from XiphMont is better

```
pactl load-module module-echo-cancel
```

Pipewire

Pipewire is great. Instead of all the above, you can use either any PulseAudio or Jack method for both splitting outputs and microphone loopback.

To split the output, just use Pipewire's “Pro Audio” mode. This will let you send a different stream to all the outputs individually.

Instead of using Pavucontrol to map and manage the outputs, you can use a Jack controller, which can more easily route any output to any input.

An example of this is to use QJackControl with Pipewire Jack instead of Jack2

```
pw-jack qjackctl
```

From there, you can map any program to output or input. It's so much more flexible then Pulseaudio if you want to run filters and whatnot, or you have a DAW like Ardour that you want to map multiple inputs and outputs to. For example, if you have an audio interface with 4 inputs, you can map it to record 4 tracks simultaneously in Ardour (or more or less, depending on what you want to accomplish).

Microphone loopback is also a lot easier, just connect the input device directly to the output device in Jack, and you'll get monitoring with less latency then Pulseaudio.

Now, you do pay with extra CPU cycles and memory usage, and I still find the audio starts glitching sporadically if I run both pulse and jack applications at the same time with different sampling rates, but it's neat.

Oh, also Bluetooth is buggier - as in, sometimes it just doesn't send the audio even though it's selected as an output when first turning on, and you need to mess with settings.

Seriously though, give it a try!