STT-TTS and Internet

Multimedia Networking - AI

P. Bakowski

bako@ieee.org

AI - Multimedia networking

prompt:

Write GStreamer pipeline to send **uncompressed** audio over RTP and UDP protocols.

```
gst-launch-1.0 -v autoaudiosrc ! audioconvert !
audio/x-raw,format=S8,channels=1,rate=16000 !
rtpL8pay ! udpsink host=<receiver_ip> port=<port>
```

compression ratio = 1

prompt:

Write GStreamer pipeline to send **compressed MP3** audio over RTP and UDP protocols.

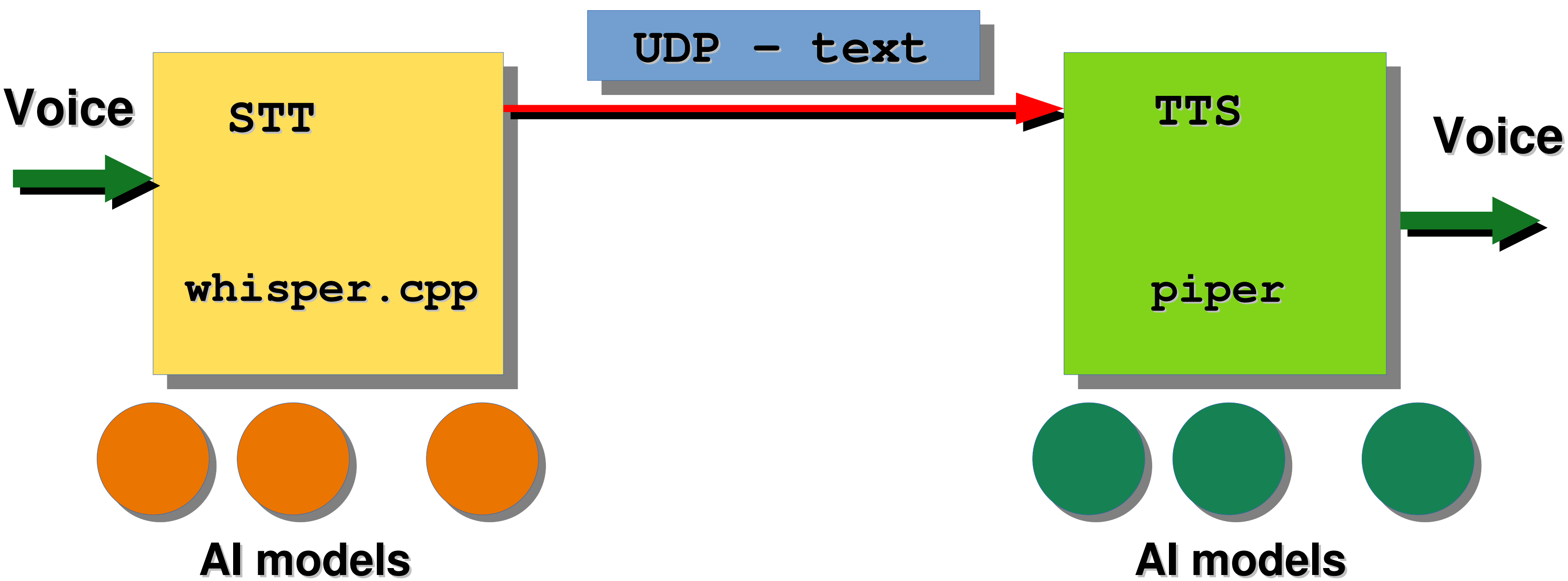
```
gst-launch-1.0 -v autoaudiosrc ! audioconvert !
lamemp3enc ! rtpmpapay ! udpsink host=<receiver_ip>
port=<port>
```

compression ratio ~= 5

AI - Multimedia networking

ASP – Automatic Speech Recognition – **Speech To Text**

Text To Speech – Text to Voice



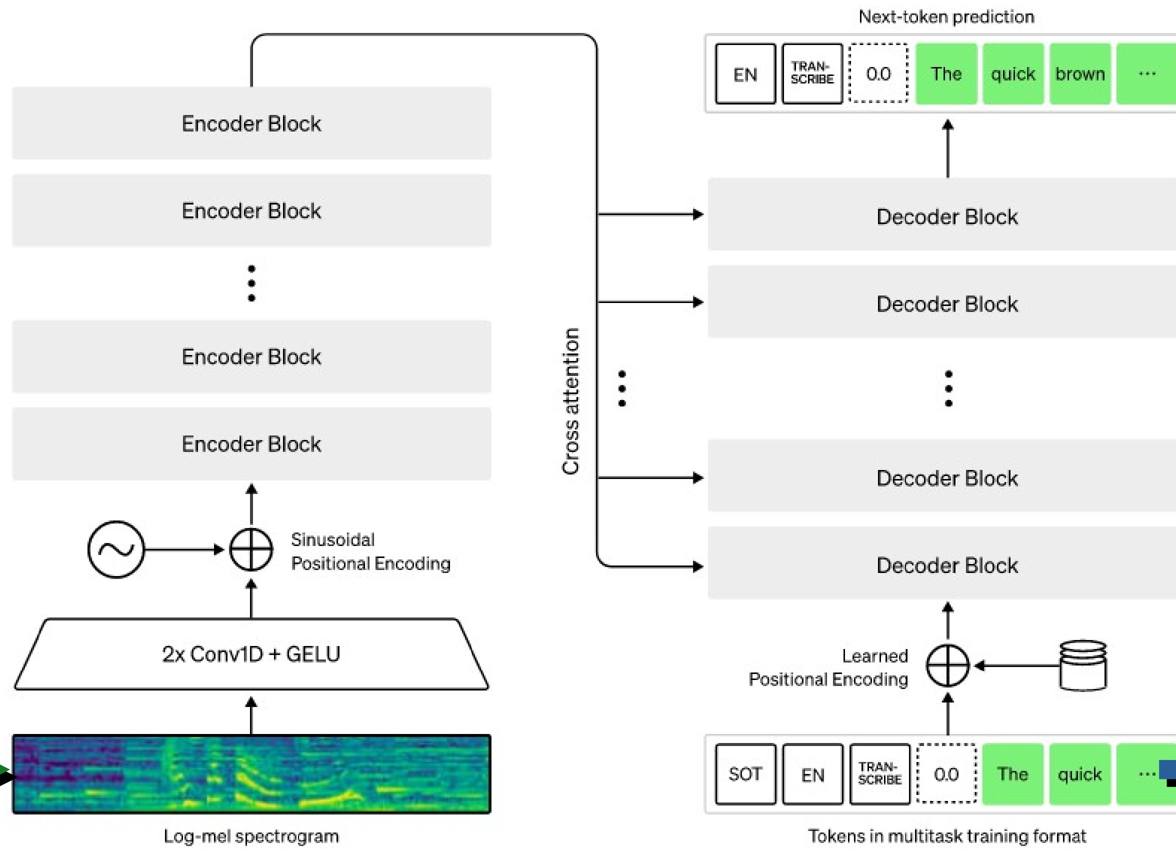
"compression" ratio ~= 50

AI – Speech To Text : whisper.cpp

The **whisper architecture** is a simple end-to-end approach, implemented as an **encoder-decoder**

whisper.cpp

model



voice

text

AI – Speech To Text : `whisper.cpp`

`whisper.cpp` is the framework – `./stream` is executable

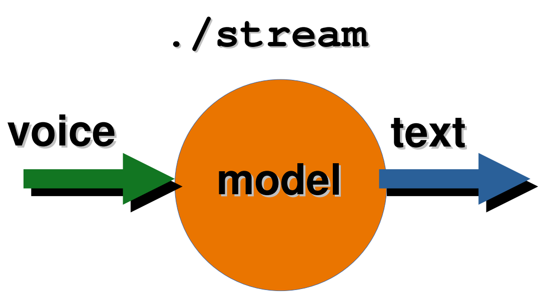
```
rock@rock-5a:~/whisper.cpp$ ./stream -h
usage: ./stream [options]
options:
  -h,        --help          [default] show this help message and exit
  -t N,      --threads N     [4       ] number of threads to use during computation
             --step N        [3000    ] audio step size in milliseconds
             --length N      [10000   ] audio length in milliseconds
             --keep N        [200     ] audio to keep from previous step in ms
  -c ID,     --capture ID    [-1      ] capture device ID
  -mt N,     --max-tokens N  [32      ] maximum number of tokens per audio chunk
  -ac N,     --audio-ctx N   [0       ] audio context size (0 - all)
  -vth N,    --vad-thold N   [0.60    ] voice activity detection threshold
  -fth N,    --freq-thold N  [100.00  ] high-pass frequency cutoff
  -su,       --speed-up      [false   ] speed up audio by x2 (reduced accuracy)
  -tr,       --translate     [false   ] translate from source language to english
  -nf,       --no-fallback   [false   ] do not use temperature fallback while decoding
  -ps,       --print-special [false   ] print special tokens
  -kc,       --keep-context  [false   ] keep context between audio chunks
  -l LANG,   --language LANG [en      ] spoken language
  -m FNAME,  --model FNAME   [models/ggml-base.en.bin] model path
  -f FNAME,  --file FNAME    [        ] text output file name
  -tdrz,     --tinydiarize   [false   ] enable tinydiarize (requires a tdrz model)
  -sa,       --save-audio    [false   ] save the recorded audio to a file
  -ng,       --no-gpu        [false   ] disable GPU inference
```

AI – Speech To Text : whisper.cpp

```
./stream -m models/ggml-base.en.bin -ac 1024
```

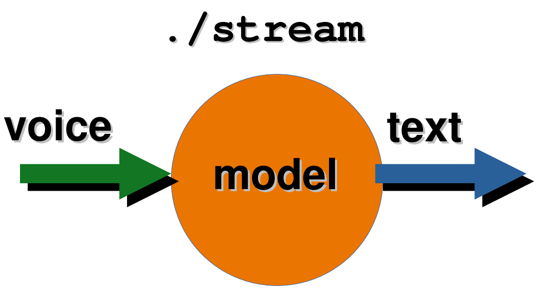
```
./stream -m models/ggml-base.en.bin --step 2000 --length 8000 \
-c 0 -t 6 -ac 512 -vth 0.6
```

```
rock@rock-5b:~/rockAI/whisper.cpp$ ./stream -m models/ggml-base.en.bin -ac
1024
init: found 3 capture devices:
init:    - Capture device #0: 'Built-in Audio Stereo'
init:    - Capture device #1: 'HD Pro Webcam C920 Analog Stereo'
init:    - Capture device #2: 'Built-in Audio Stereo (2)'
init: attempt to open default capture device ...
init: obtained spec for input device (SDL Id = 2):
init:     - sample rate:       16000
init:     - format:            33056 (required: 33056)
init:     - channels:          1 (required: 1)
init:     - samples per frame: 1024
..
```



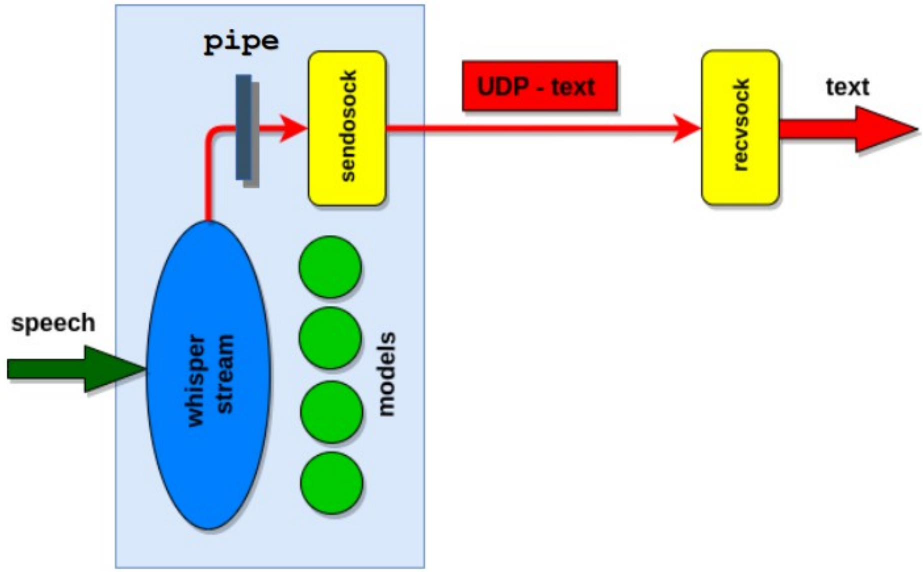
AI – Speech To Text : whisper.cpp

```
rock@rock-5b:~/rockAI/whisper.cpp$ ./stream -m models/ggml-base.en.bin -ac 
1024
init: found 3 capture devices:
init:    - Capture device #0: 'Built-in Audio Stereo'
init:    - Capture device #1: 'HD Pro Webcam C920 Analog Stereo'
init:    - Capture device #2: 'Built-in Audio Stereo (2)'
init: attempt to open default capture device ...
init: obtained spec for input device (SDL Id = 2):
init:     - sample rate:       16000
init:     - format:            33056 (required: 33056)
init:     - channels:          1 (required: 1)
init:     - samples per frame: 1024
..

[Start speaking]
 [ Silence ]
 (crickets chirping)
 Hello, how are you going?
 [ Silence ]
 Where is the next bus station?
```



./stream – filtering pure speech

```
#include <stdio.h>
#include <string.h>

int main() {
    int c;
    int txt=0,nl=0,sp=0,sb=0,nb=0,count=0;
    char buff[512];int i=0;
    while ((c = getchar()) != EOF) {
        putchar(c);   // to be commented
        count++;
        if(c==0x0D) { count=0; }
        if(count==2)
        {
        switch(c)
            {
            case 0x20: txt=0;sp=1; break;
            case 0x5B: txt=0;sb=1; break;
            case 0x28: txt=0;nb=1; break;
            default: txt=1;sp=0;sb=0;nb=0;i=0;memset(buff,0x00,512);break;
            }
        }
        if(txt) buff[i++]=c;
        if(c=='.' || c=='?' || c=='!') { txt=0; printf("\n%s\n",buff);}
    }
    return 0;
}
```



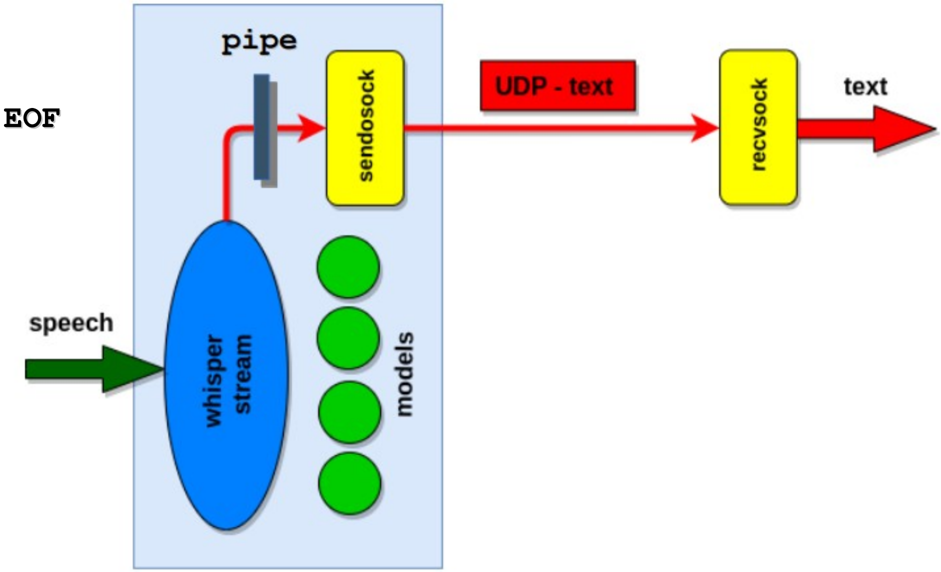
filtered “pure speech” text

./stream – filtering pure speech

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define BUFLEN 512      // Max length of buffer
#define PORT 8888       // The port on which to send or listen for data
#define REMOTE_ADDR  "192.168.1.31"

int main() {
  int c;
  int txt=0,nl=0,sp=0,sb=0,nb=0,count=0;
  char buff[512];int i=0;

  struct sockaddr_in si_me, si_other;
  int s, slen = sizeof(si_other) , recv_len;
  if ((s=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP))==-1){exit(1);}
  memset((char *) &si_other, 0, sizeof(si_other));
  si_other.sin_family = AF_INET;
  si_other.sin_port = htons(PORT);    // to work remotely
  si_other.sin_addr.s_addr = inet_addr(REMOTE_ADDR);   // to work remotely
```

preparing UDP socket

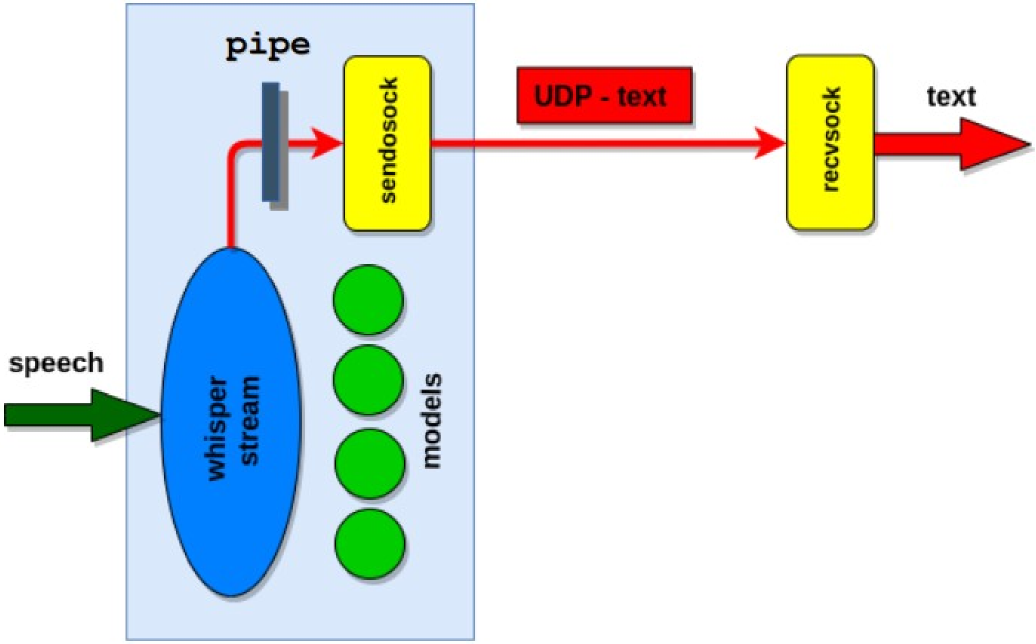
./stream – filtering pure speech




```
// Read characters from standard input until EOF
memset(buff,0x00,512);
while ((c = getchar()) != EOF)
  {
      putchar(c);
      count++;
      if(c==0x0D) { count=0; }
      if(count==2)
      {
      switch(c)
          {
          case 0x20: txt=0;sp=1; break;
          case 0x5B: txt=0;sb=1; break;
          case 0x28: txt=0;nb=1; break;
          default: txt=1;sp=0;sb=0;nb=0;i=0;memset(buff,0x00,512);break;
          }
      }
      if(txt) buff[i++]=c;
      if(c=='.' || c=='?' || c=='!')
        { txt=0; printf("\n%s\n",buff);
          sendto(s,buff,strlen(buff)+1,0,(struct sockaddr *)&si_other,slen);
        }
    }
  return 0;
}
```

sending UDP packet

./stream .. | sendsock



On the receiving side :

```
  ..
while(1)
    {

    if((rlen=recvfrom(s,buf,BUFLEN,0,(struct sockaddr *)&si_other,&slen))==-1)
      {printf("recvfrom error\n"); }
    printf("%s\n", buf);
    }
..
```

```
./stream -m models/ggml-base.en.bin -ac 1024 | ./sendsock
```

echo 'text' | ./piper (TTS)

`piper` is optimized to perform well on ARM (NEON); it takes 1 second to generate 1.6 seconds of speech audio at a medium quality level.

`piper` can stream raw audio to `stdout` as its produced.

`piper` needs pre-trained models (`.onnx`) of the speaker

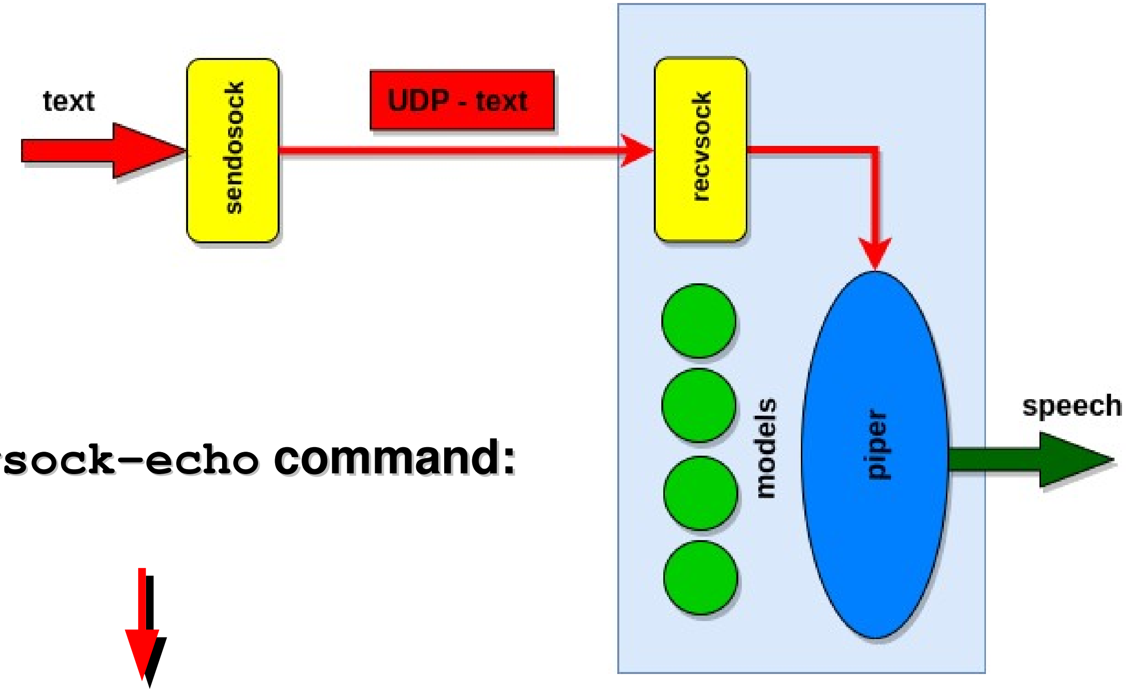
Streaming audio with `echo` command:

```
echo 'This sentence is spoken first. This is Smart Computer Lab
multimedia,internet and AI module.' | \
./piper --model GB_female_south/en_GB-southern_english_female-low.onnx --
output-raw | \
aplay -r 15000 -f S16_LE -t raw -
```

`aplay` to reproduce the samples of the voice on audio output

./recvsock | ./piper

`./recvsock` **is a program that receives UDP datagrams with text and redirects them to the** `./piper` **via** `echo` **command.**

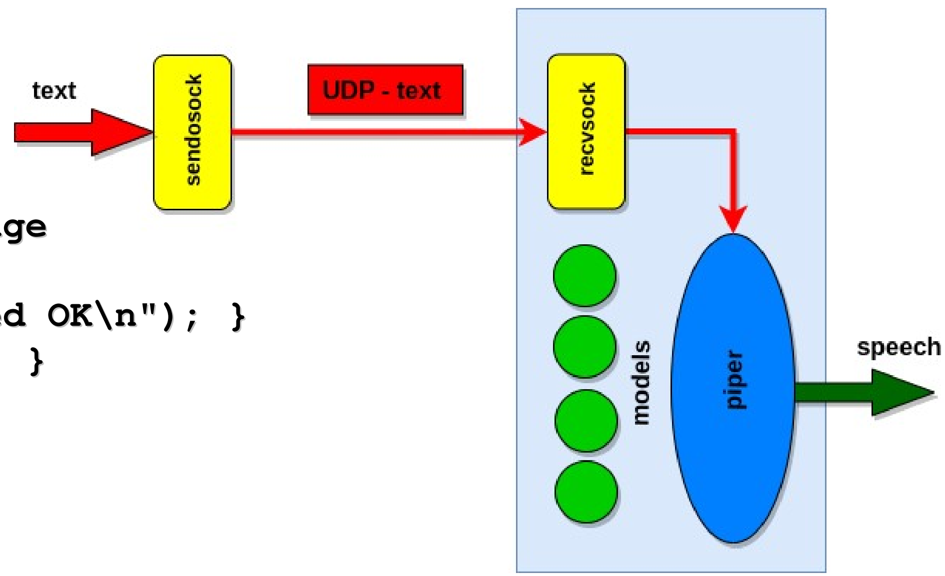


Streaming audio with `./recvsock-echo` **command:**

```
./recvsock | \
./piper --model GB_female_south/en_GB-southern_english_female-low.onnx \
--output-raw  |  aplay -r 15000 -f S16_LE -t raw -
```

./recvsock | ./piper

```
..
while(1)
    {
    memset(message,0x00,512);
    if((rlen=recvfrom(s,message,512,0,(struct sockaddr *)&si_other,&slen))==-1)
      {printf("recvfrom() error\n"); }
    memset(buffer,0x00,1000);
    strcpy(buffer,"echo ");
    strcat(buffer,message);
    sleep(1);
    // generate echo command with message
    retcode=system(buffer);
    if (retcode == 0) { printf("executed OK\n"); }
    else { printf("execution error\n"); }
    }
```

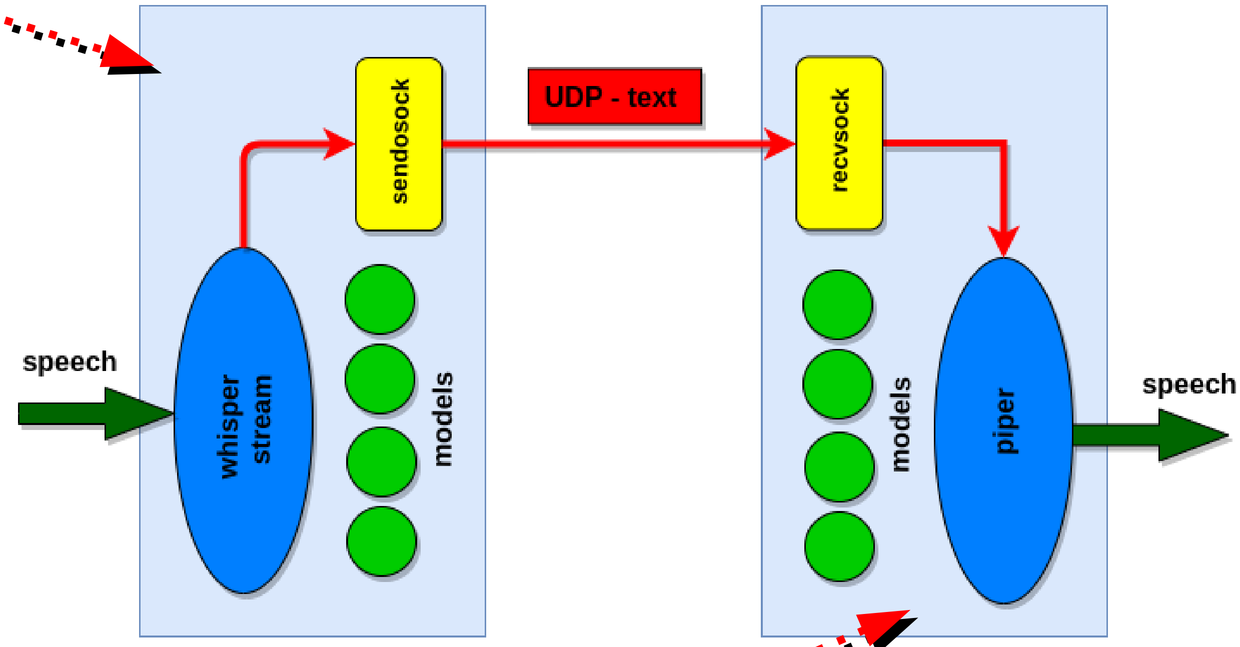


echo

```
./recvsock | \
./piper --model GB_female_south/en_GB-southern_english_female-low.onnx \
--output-raw  |  aplay -r 15000 -f S16_LE -t raw -
```

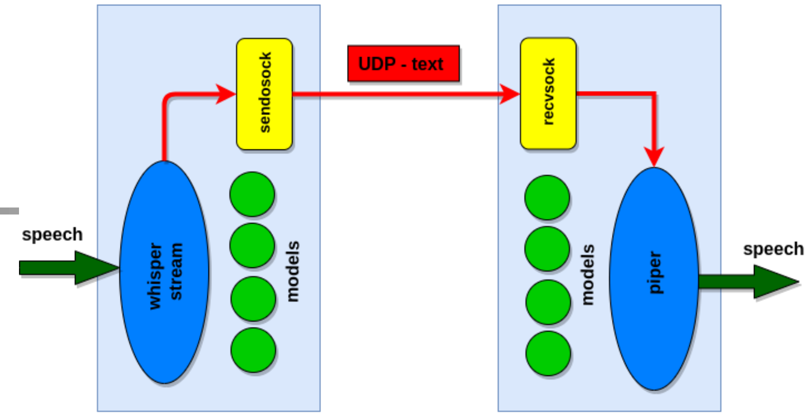
whisper.cpp - UDP - piper

```
./stream -m models/ggml-base.en.bin  -ac 1024  | ./sendsock
```




```
./recvsock | \
./piper --model GB_female_south/en_GB-southern_english_female-low.onnx \
--output-raw  |  aplay -r 15000 -f S16_LE -t raw -
```

Summary



ASP – Automatic Speech Recognition – Speech To Text with `whisper.cpp` (`./stream`)

Filtering speech from generated text (sounds, noise, silence, and speech)

Sending and receiving text speech over UDP

Text To Speech with `./piper`

receiving text speech over UDP and playout