Fancy Math for a Stupid Chat

By Milkey Mouse

I just feel like somebody's watching me

- School tracks all Wi-Fi, Bluetooth, and radio communications from any school computer
- DyKnow makes the situation even worse if you're not in Safe Mode
- If you're on your own device and connected to LWSD-GUEST they also track you
- They scramble signals from cell towers so nobody uses their data plan
- Their Microsoft Exchange server is set to keep all emails and conversations for 100 days after deletion, even if perma-deleted by the user
- But the one thing they DON'T track…

AUDIO!

Audio as a Transport

- It’s just too weird for them to think of
- Why would DyKnow need audio mirroring?
- Worst case scenario some kid is listening to Justin Bieber, right?
- WRONG!
- By using sound you can transfer quite a bit of data (I’ll get back to this in a second)

But wait, there's more

- The Mosquito Tone
  - Invented by a shopkeeper and an age-discriminatory sonic weapon
  - It would make an interesting argumentative essay
  - Very high-pitched
  - **Can only be heard by people under 25** (so no teacher snooping!)
    - The cutoff age can change by changing the frequency minute amounts
- No known side effects

Transmission Formats (or encodings, as they're known in the 1337 h4x0r biz)

- Using different frequencies (and rounding to the nearest bin in case of error) for multiple characters
- Morse code (…---…)
  - Pretty much the original, used as early as 1844
  - Very good margins for error with only 2 possibilities
  - But slooow
    - If each dot took 0.2 seconds and each dash took 0.5, this would take 1.8 seconds to transmit the word "sos"
- ASCII
  - 128 characters of stuff
  - What early computers/computer enthusiasts used
  - Only Latin characters, numbers, and some punctuation
  - Needless to say, that's not what they use today
  - Margin for error would be too small with 128 discrete frequencies/"bins"
- Base64
  - Uses only 64 characters to represent any other character, it just takes more of them
  - Doubles the size of each bin for the frequencies
- Base32
  - Like Base64, but uses 32 characters instead of 64 and the resulting messages are even longer
  - Doubles the margins again from Base64

So I chose Base32, because after testing even Base64 didn't have small enough bins. On the bright side, the bins of each frequency are so large that each character can be transmitted in under 0.1 seconds.

Now how to generate the sound?

"Hello World"

- If using Base32/Base64, encode the message now

  "JJBSWY3DPEBLW64TM MQ= = = = = ="

- Get a numerical index for each character

  [9, 9, 1, 18, 22, 24, 27, 3, 15, 4, 1, 11, 22, 30, 28, 19,
  12, 33, 12, 16, 32, 33, 32, 33, 32, 33, 32, 33, 32, 33, 32]

- For each resulting number, multiply by 100 and add 17000 Hz (just a little bit below the Mosquito Tone). Save this to an audio file in memory, so it never gets stored on the hard drive and becomes trackable.

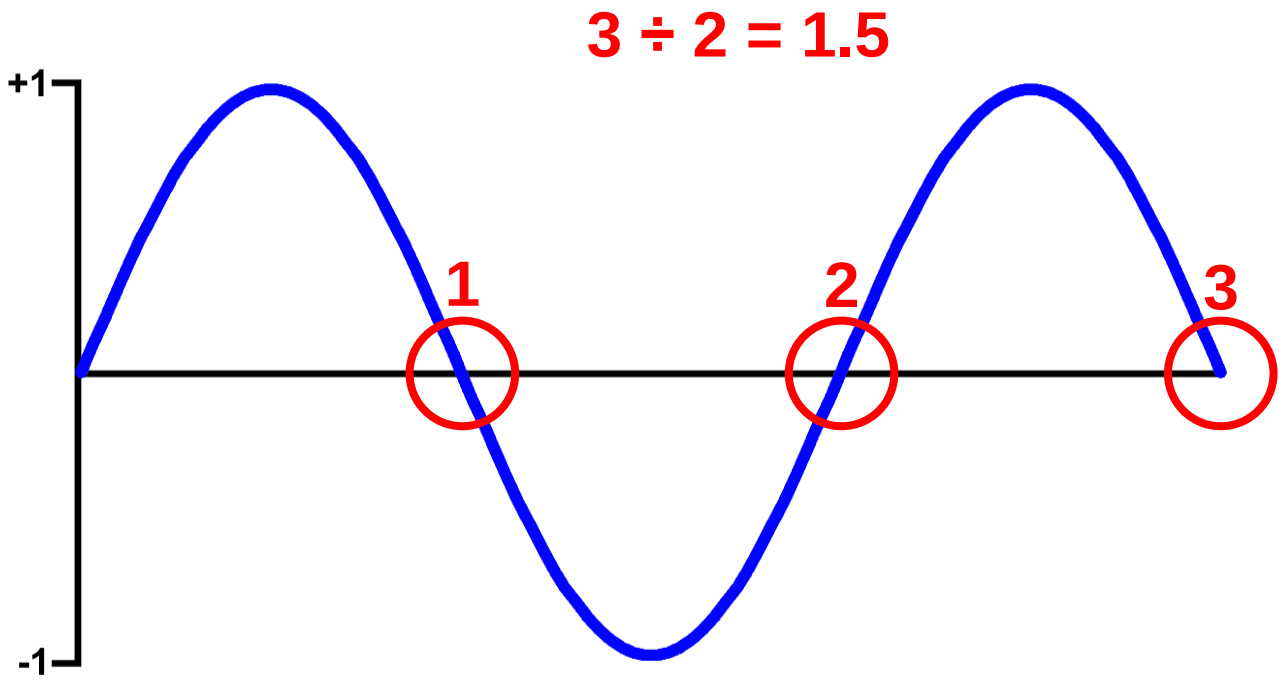
- Play the audio file at full volume. (Since the sounds are so high-pitched, it doesn't matter how loud it is.

But there’s another problem…

WARNING:
MATH INCOMING

The Zero-Crossing Algorithm

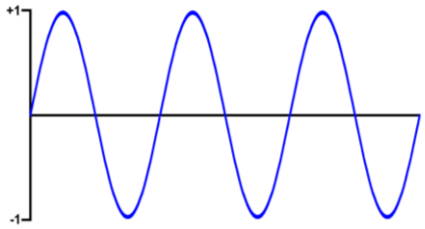
Just count the times the wave crosses zero!



I saw a sine
And it was made of points and lines

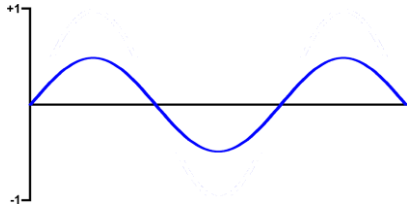
```
signal = np.fromstring(signal, 'Int16');
crossing = [math.copysign(1.0, s) for s in signal]
index = find(np.diff(crossing));
f0=round(len(index) *RATE /(2*np.prod(len(signal))))
return f0;
```

Problems with the Zero-Crossing Algorithm



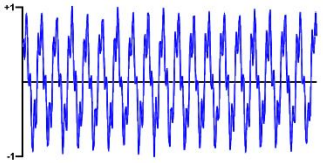
Ultrasonic frequency

+



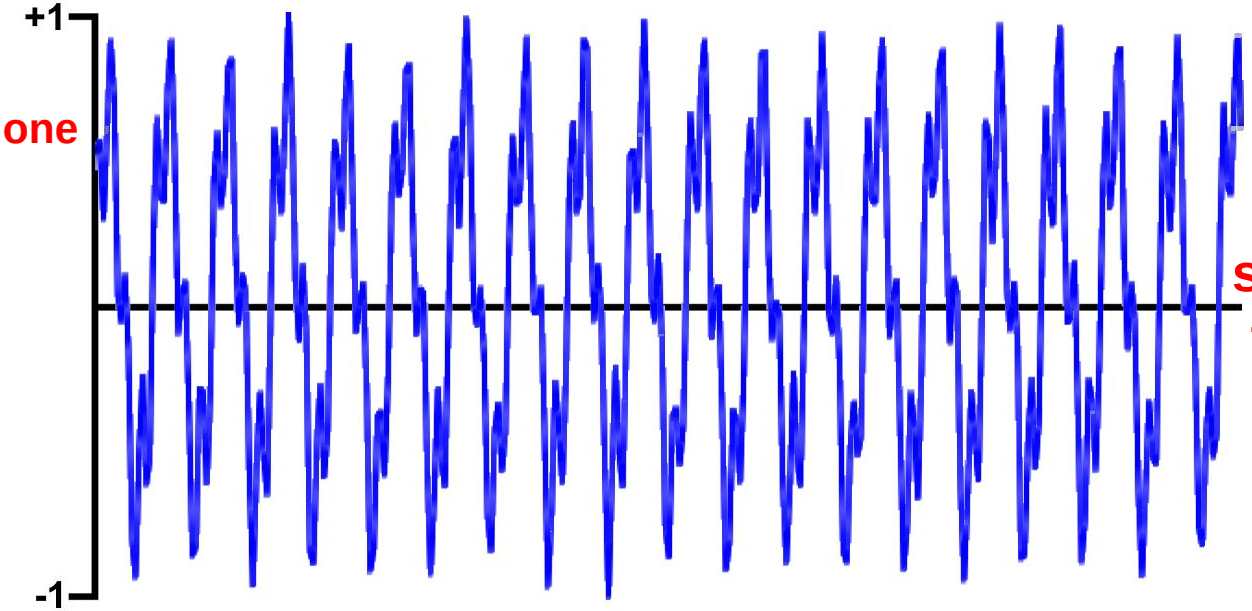
Audible frequency (e.g. talking)

=



A weird thing

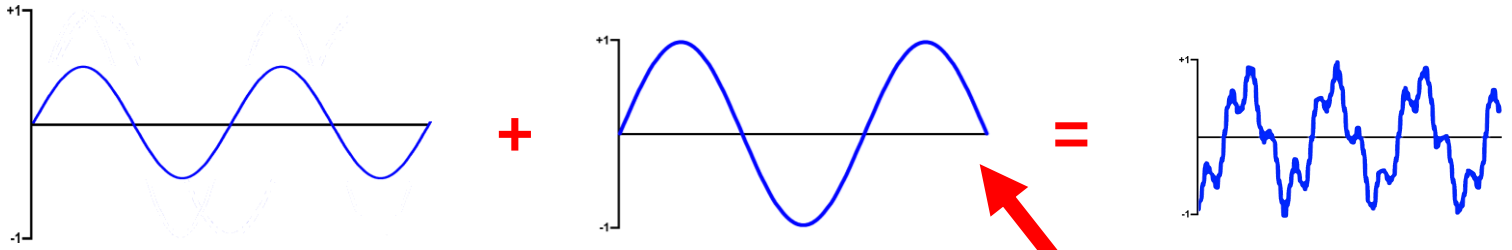
closer to this one



Result is the same as that of the ultrasonic transmission

All fine and dandy when the ultrasound is louder than the speech…

Problems with the Zero-Crossing Algorithm

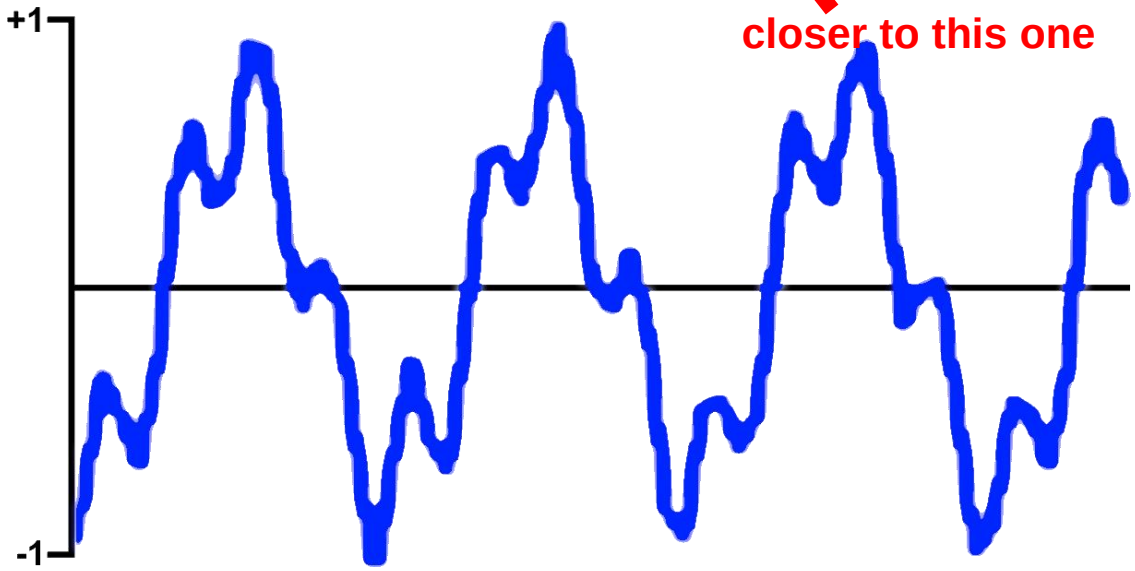


Ultrasonic frequency

Audible frequency (e.g. talking)

An even weirder thing

closer to this one



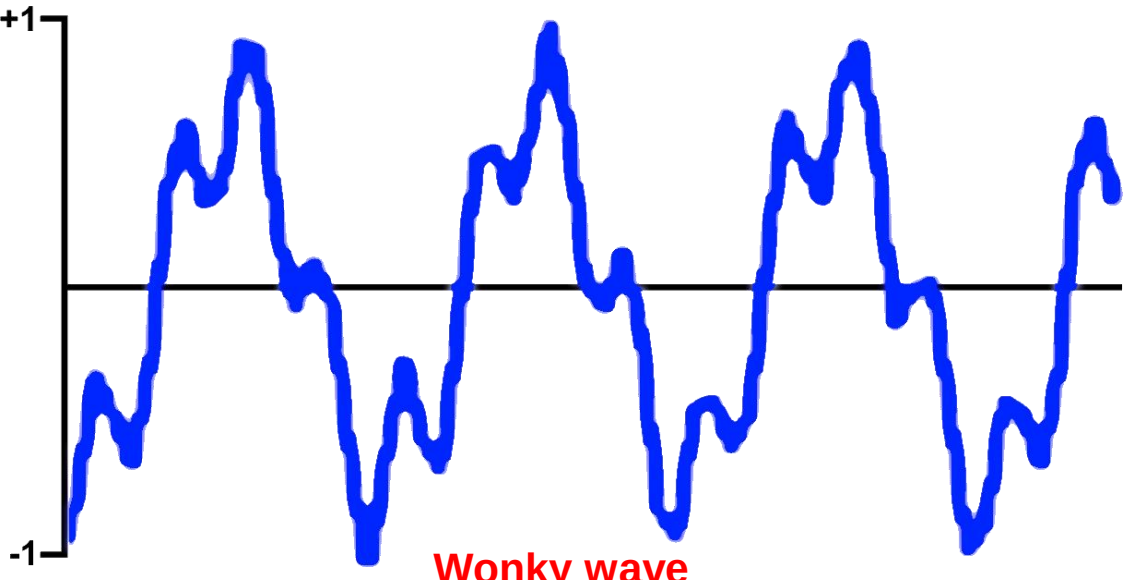
Result is the same as the talking!
☹

…but when talking is louder than the transmission it wonks out (the technical term)

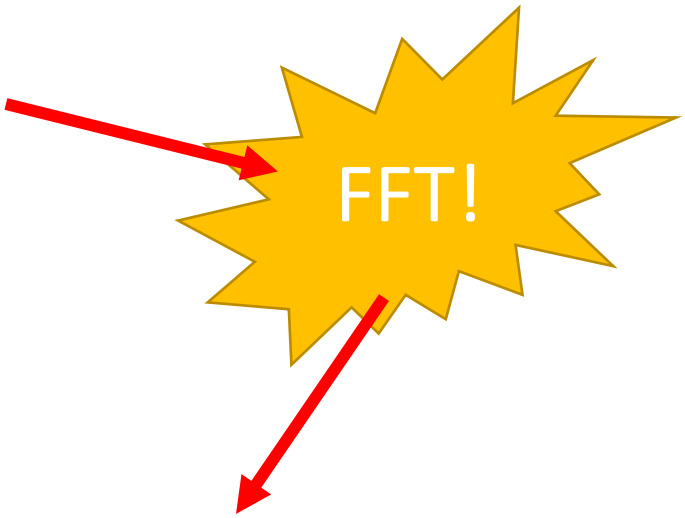
So with any noise in the background the Zero Crossing Algorithm won't work.

Enter Fast Fourier Transform (FFT)

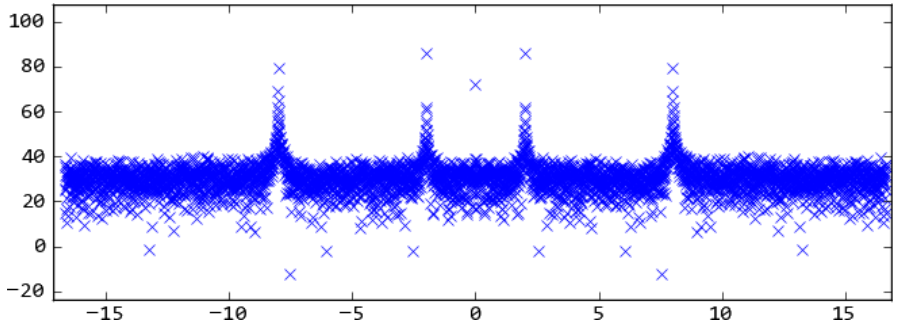
(pronounced four-ee-yay)



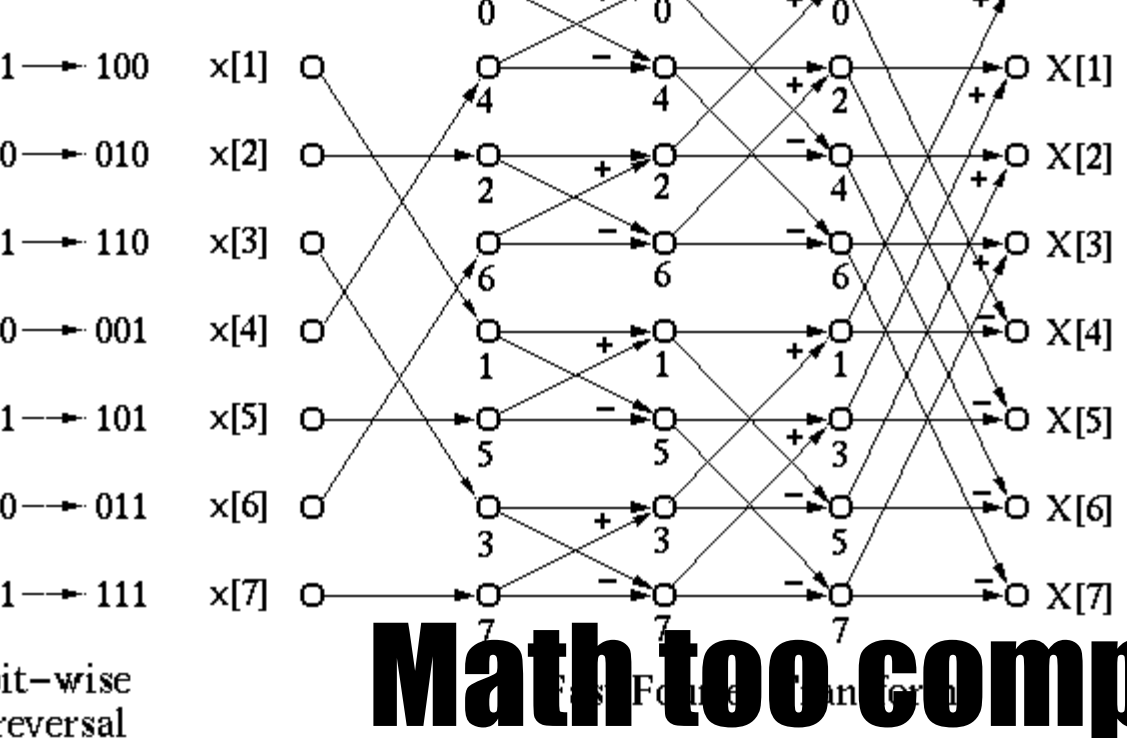
Wonky wave

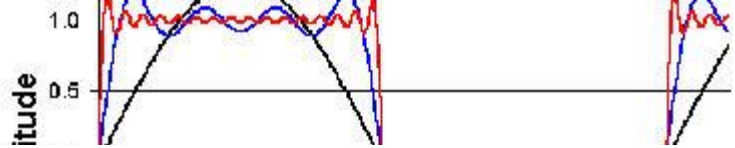


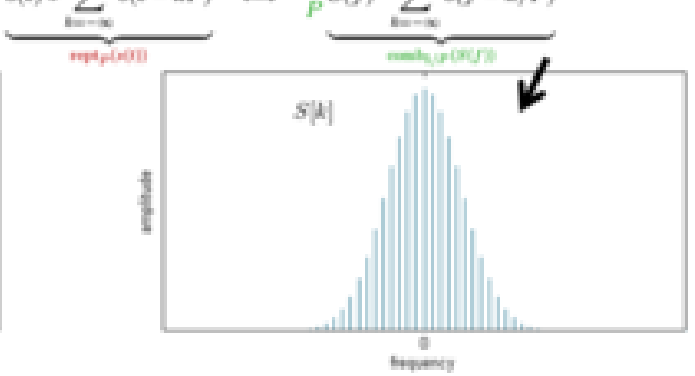
An FFT outputs a nice pretty histogram of possible frequencies.



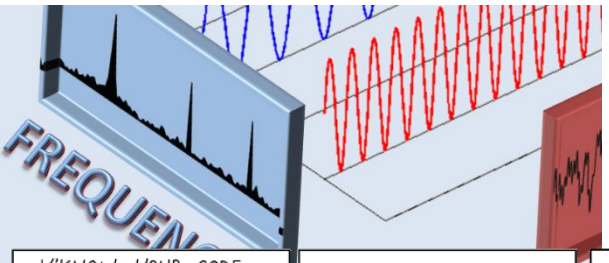
How does it do this?

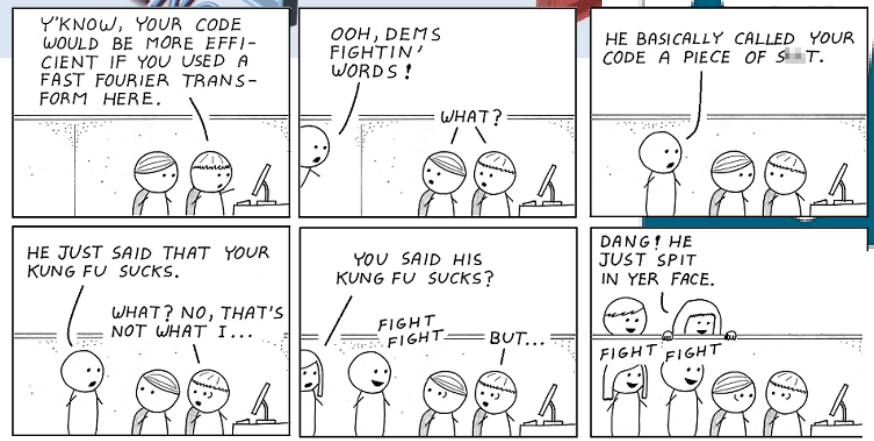


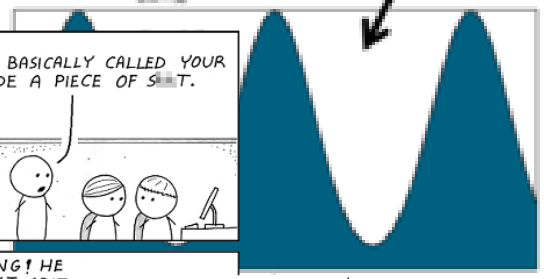


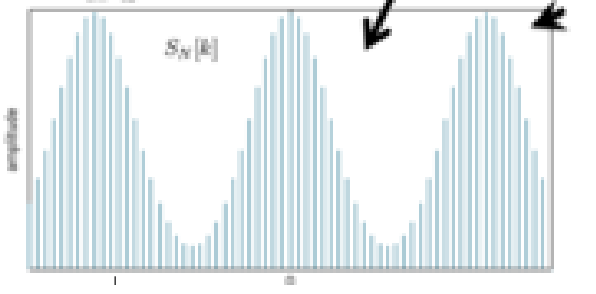


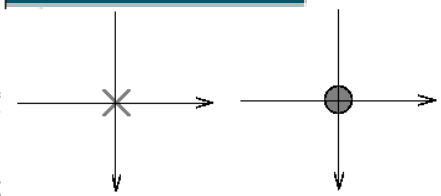
Math too complicated to go into here.

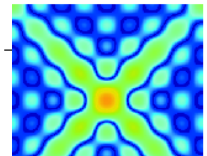


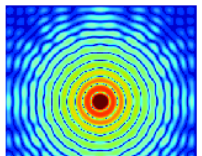


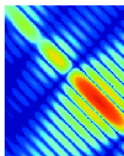








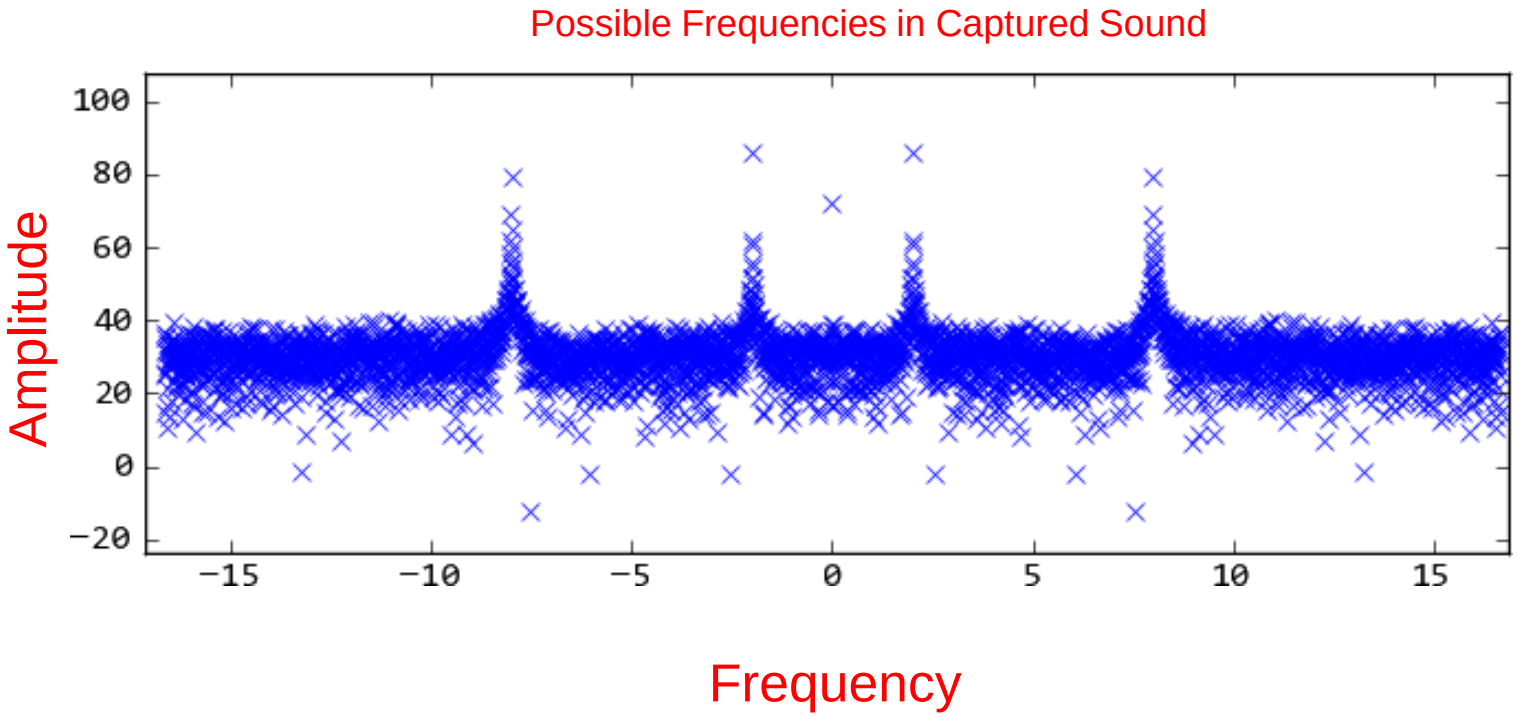




Enter Fast Fourier Transform (FFT)

(pronounced four-ee-yay)

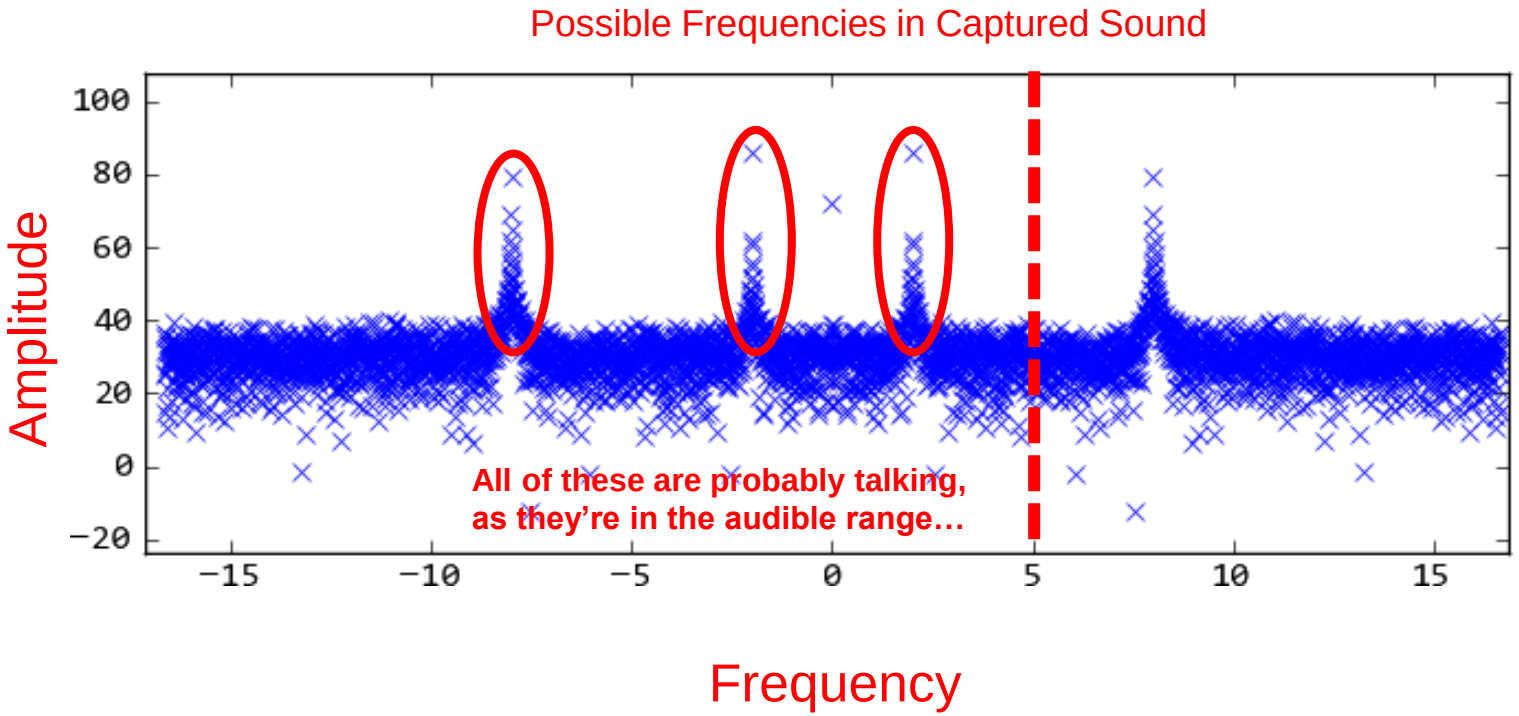
Now let's take a look at that histogram, shall we?



Enter Fast Fourier Transform (FFT)

(pronounced four-ee-yay)

First order of business: cut off the audible range

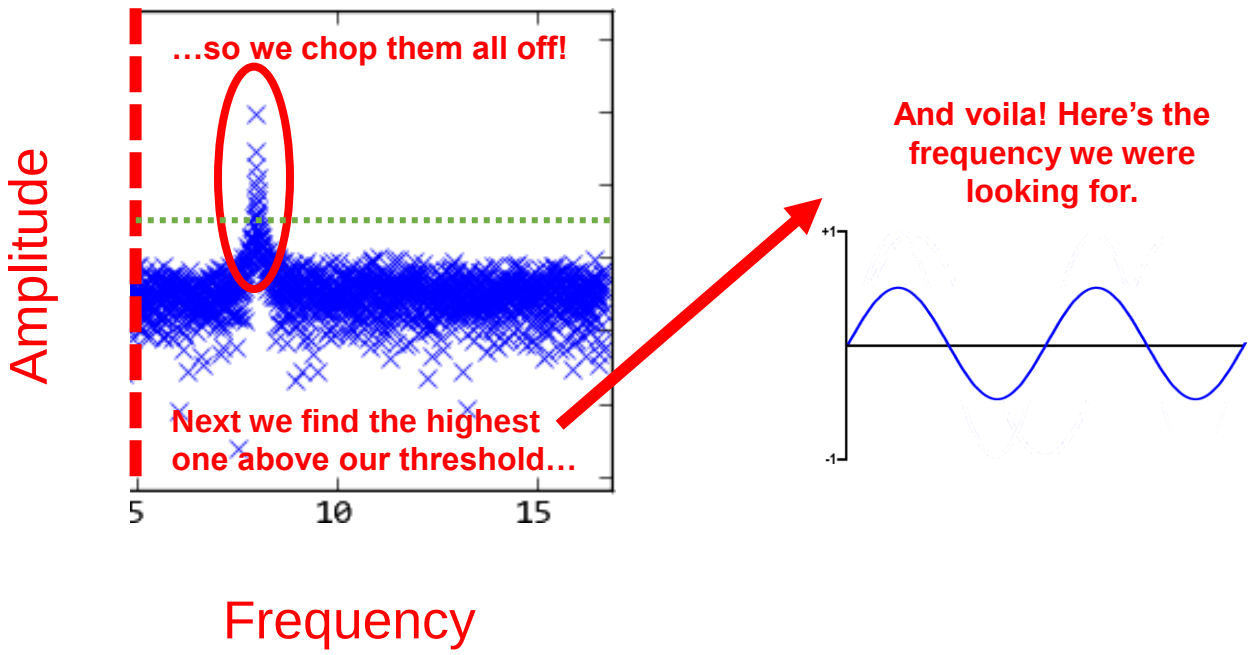


Enter Fast Fourier Transform (FFT)

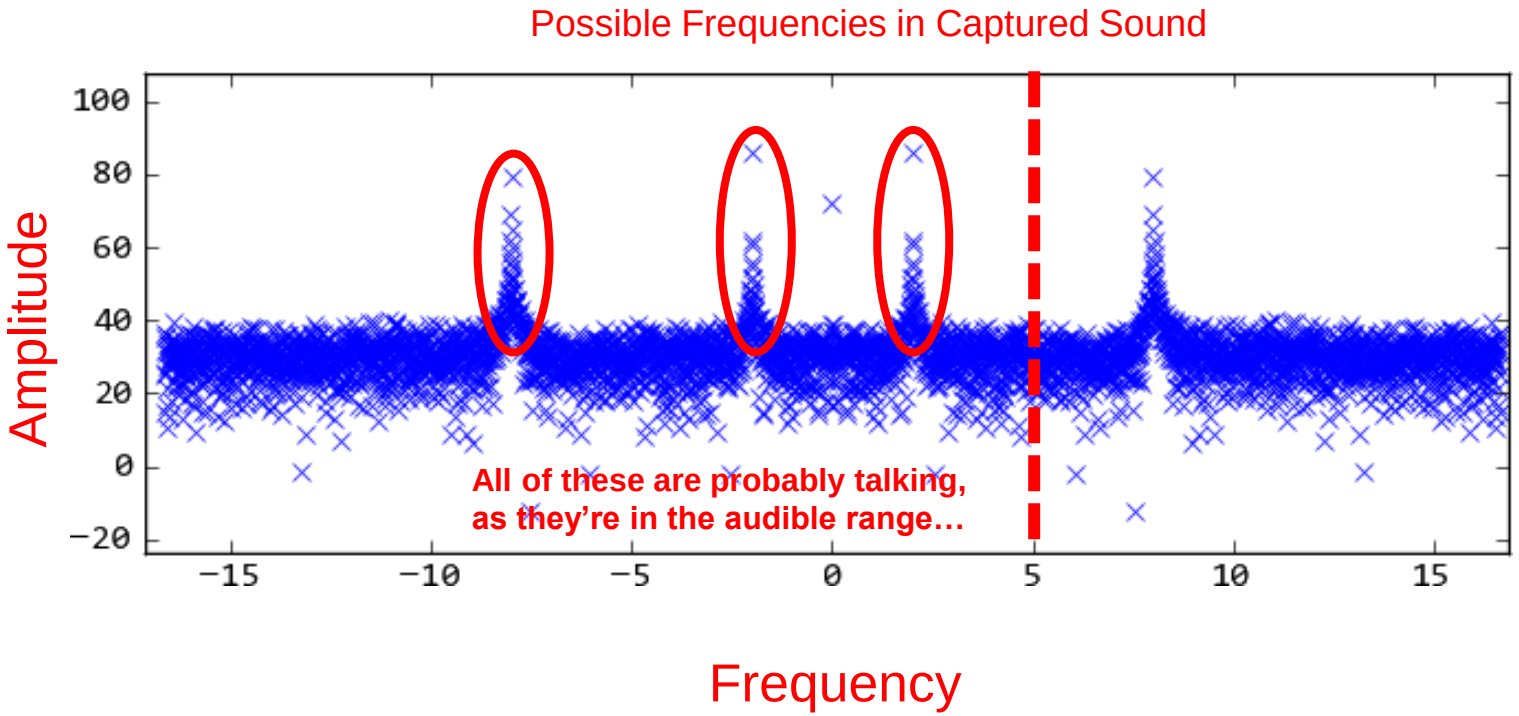
(pronounced four-ee-yay)

Then we get the frequency over the threshold

Possible Frequencies in Captured Sound



As evidenced by the heights of the original audible peaks, the FFT algorithm filters noise very well and can be used to pick up transmissions even from across the room!



Conversion back to text is pretty much the same but backwards

- Stream audio from the mic and run FFTs until you get outlier frequencies (the FFT conveniently has bin sizes of 100 by default)

```
[9, 9, 1, 18, 22, 24, 27, 3, 15, 4, 1, 11, 22, 30, 28, 19,
12, 33, 12, 16, 32, 33, 32, 33, 32, 33, 32, 33, 32, 33, 32]
```

- With the same conversion table as last time, turn the numbers back into letters and process the data.

```
“JBSWY3DPEBLW64TMMQ======”
```

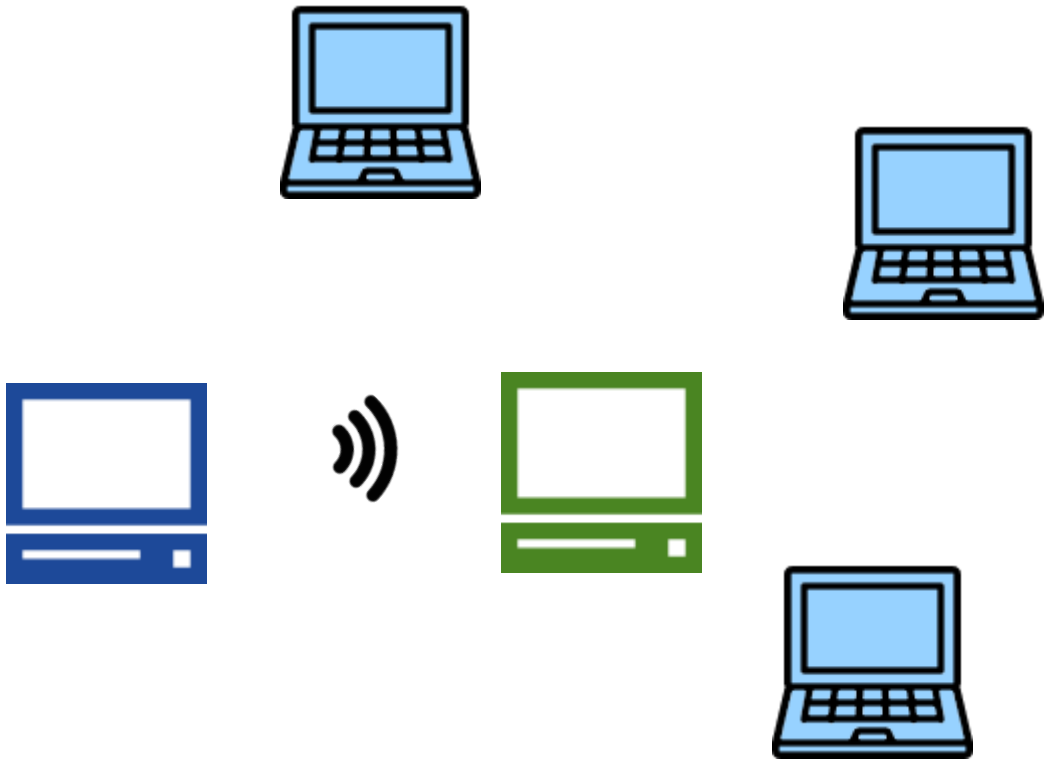
- Use Base64/Base32 if needed to decode the message into a string again.

```
“Hello World”
```

But what if you're *really* far away?

Take clues from torrents and make it P2P

Take clues from torrents and make it P2P



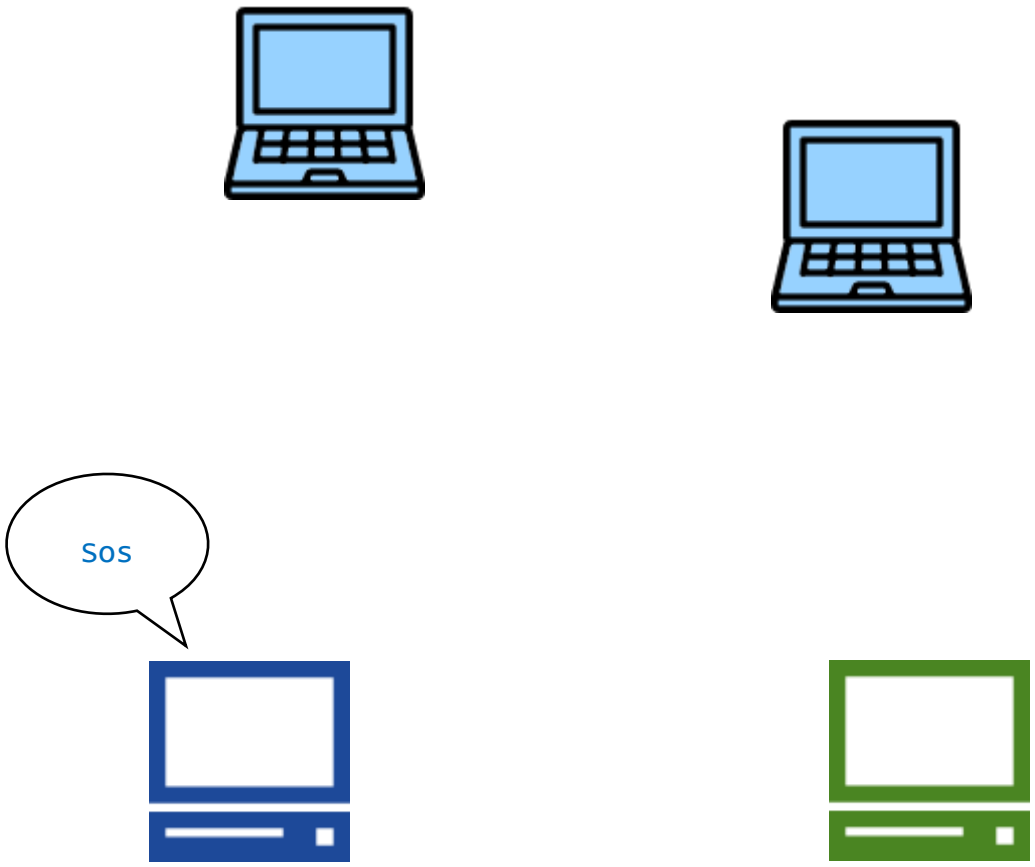
Take clues from torrents and make it P2P

Take clues from torrents and make it P2P

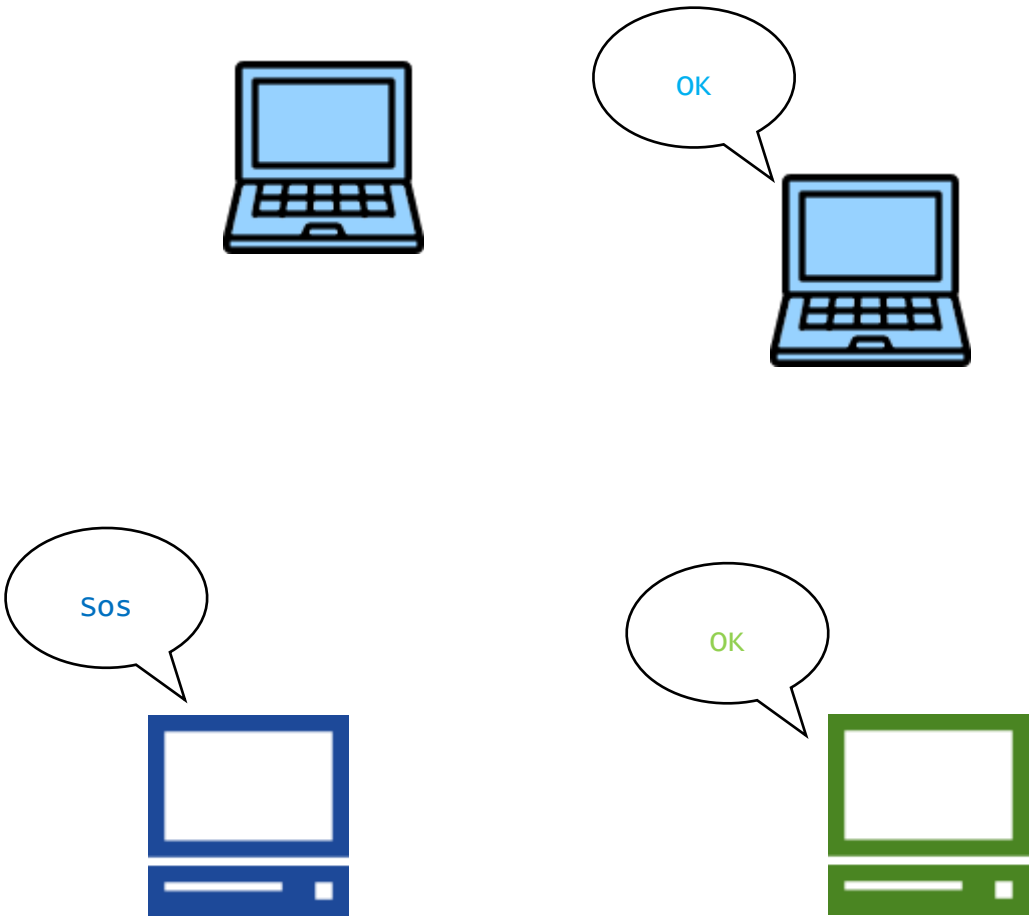
Take clues from torrents and make it P2P

And have error correction

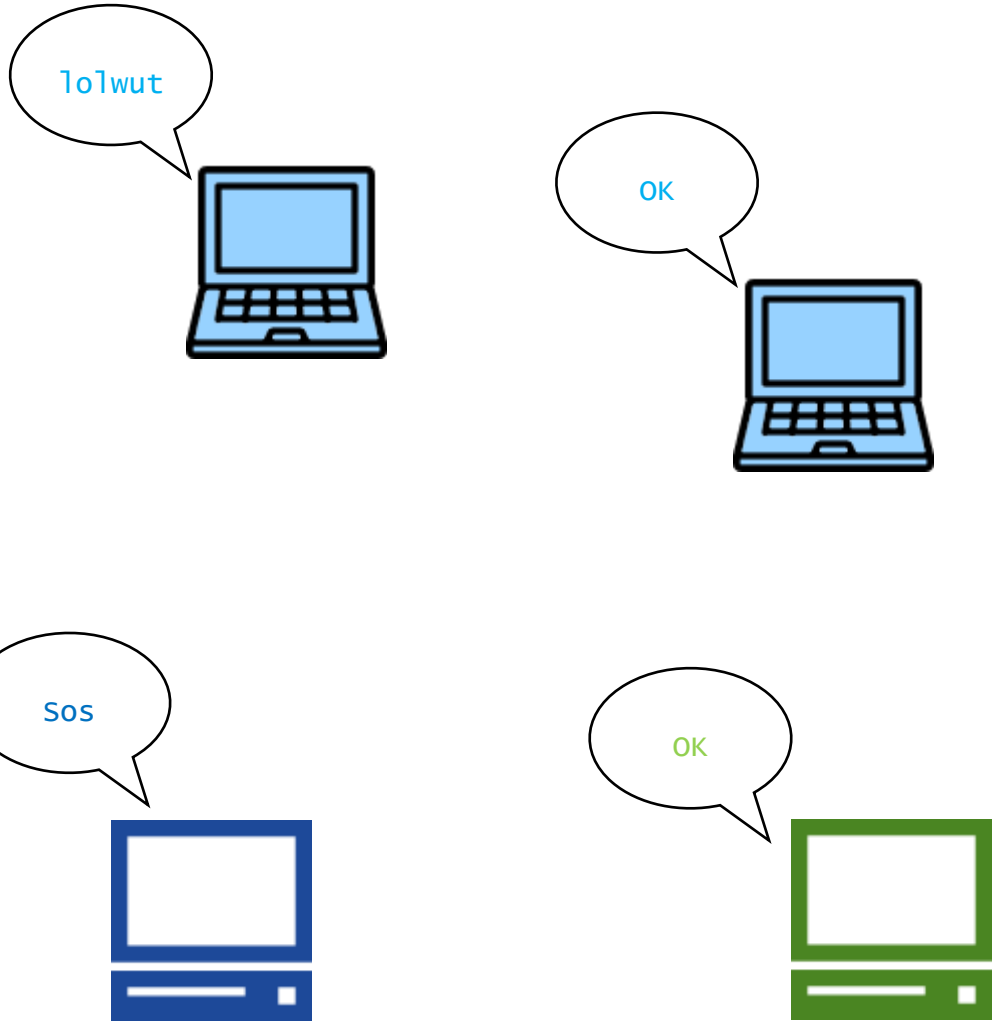
And have error correction



And have error correction

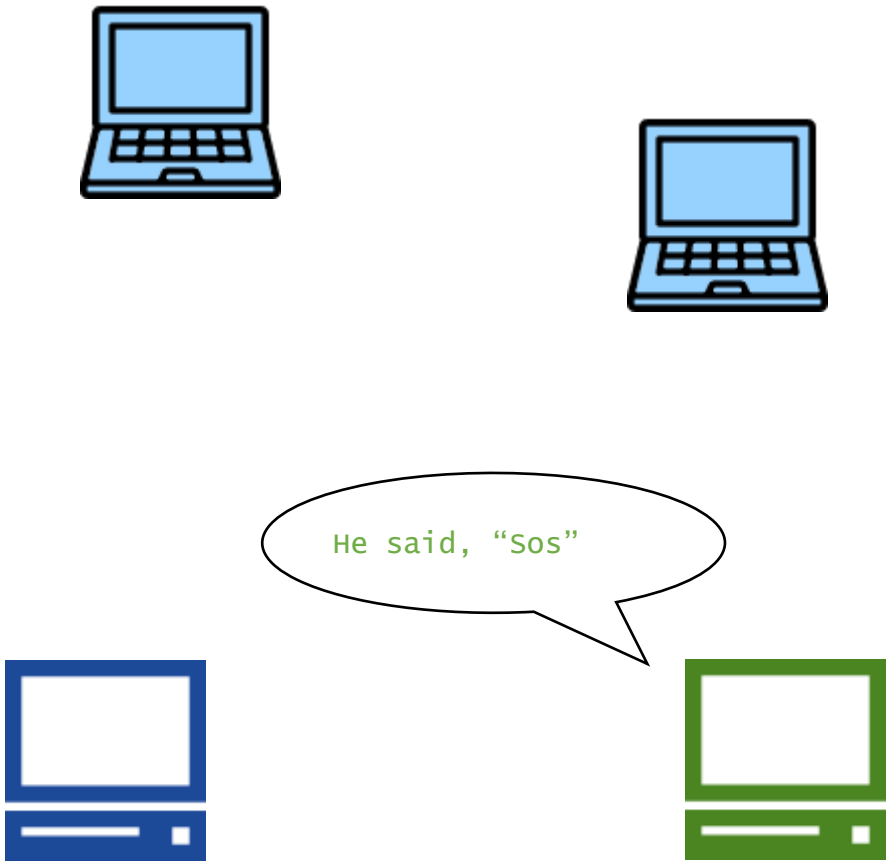


And have error correction

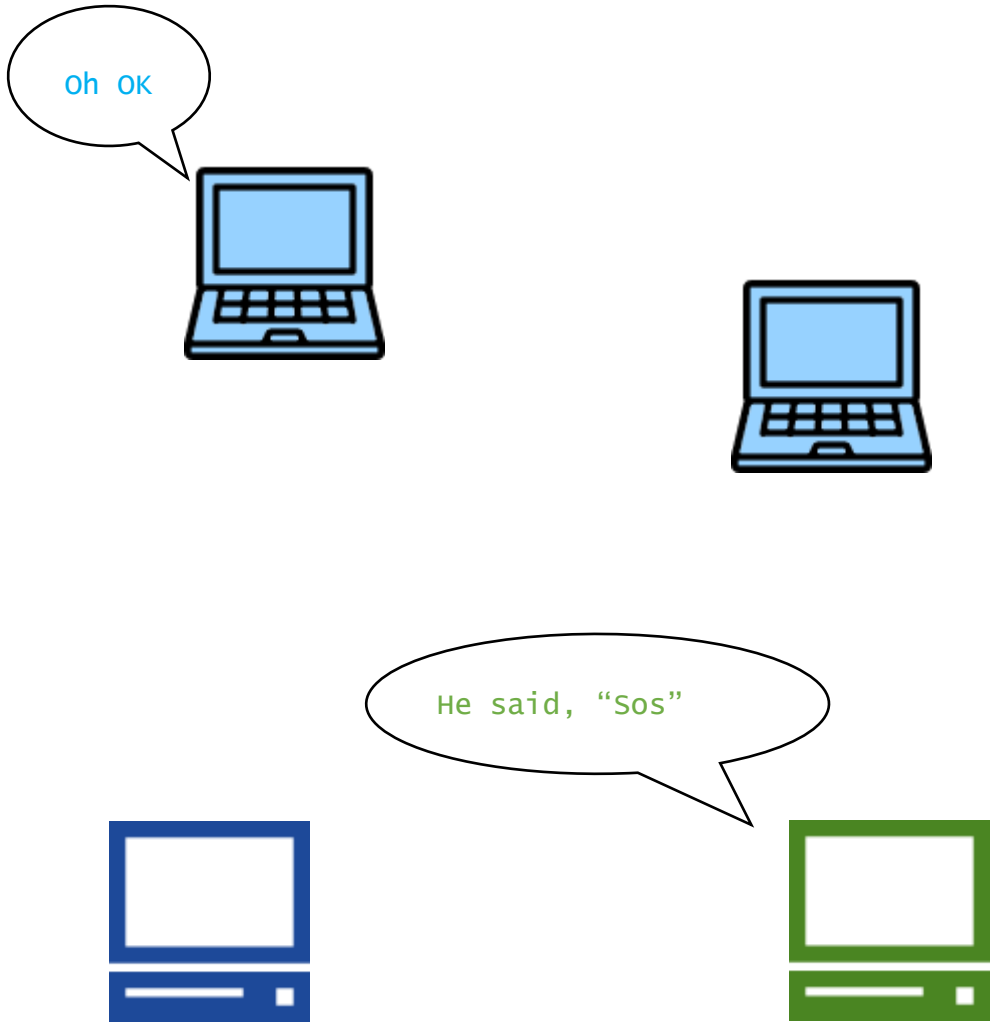


And have error correction

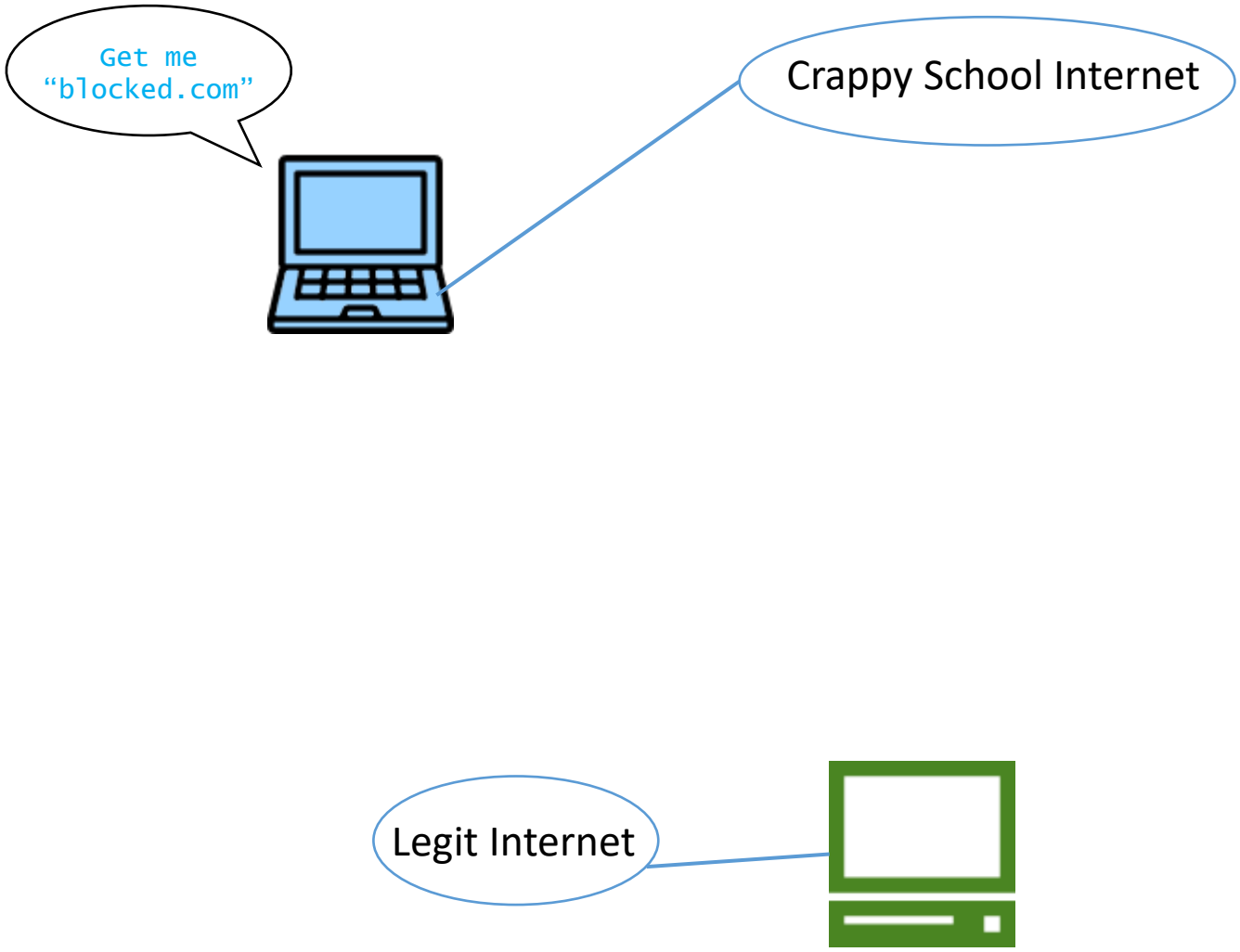
And have error correction



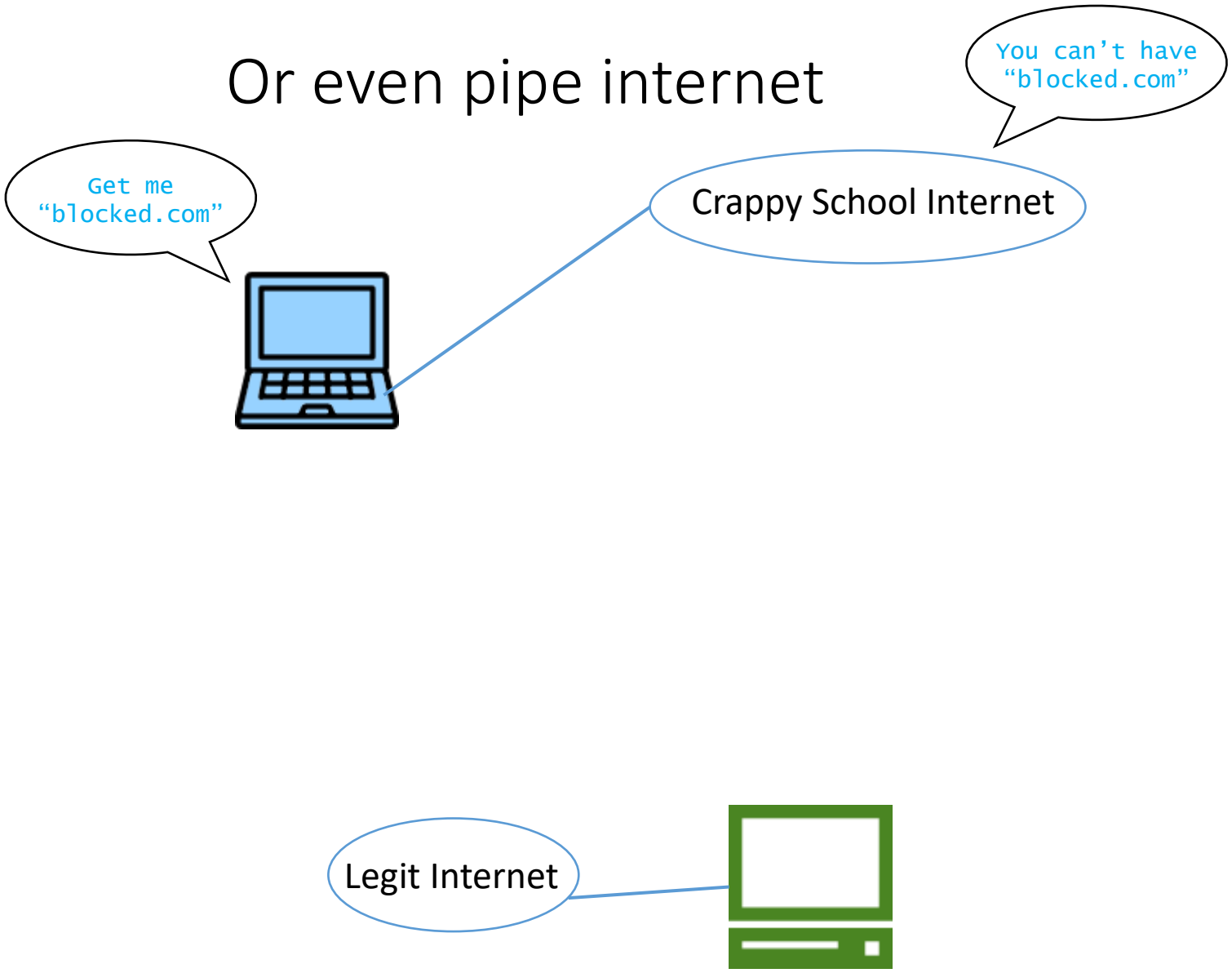
And have error correction



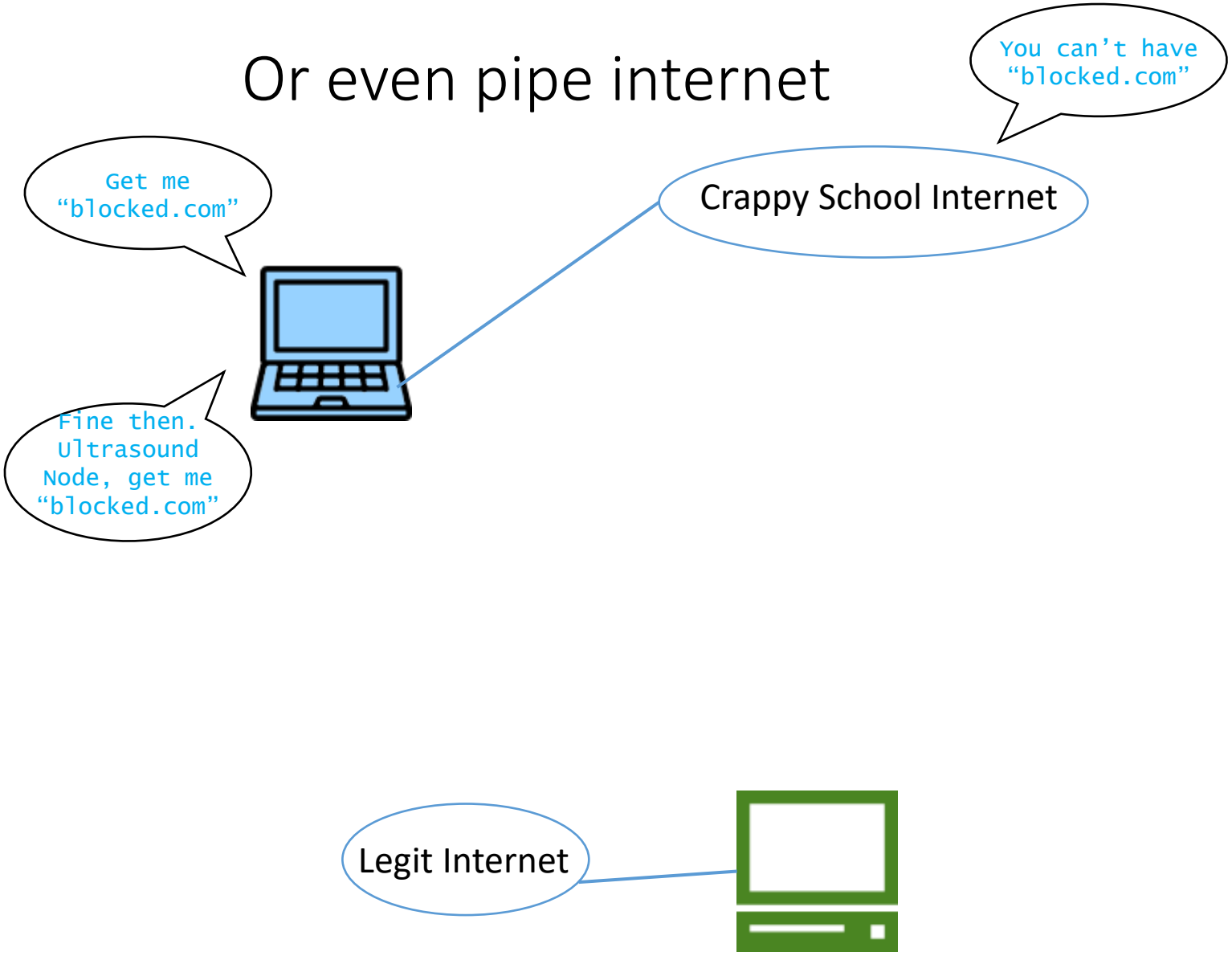
Or even pipe internet



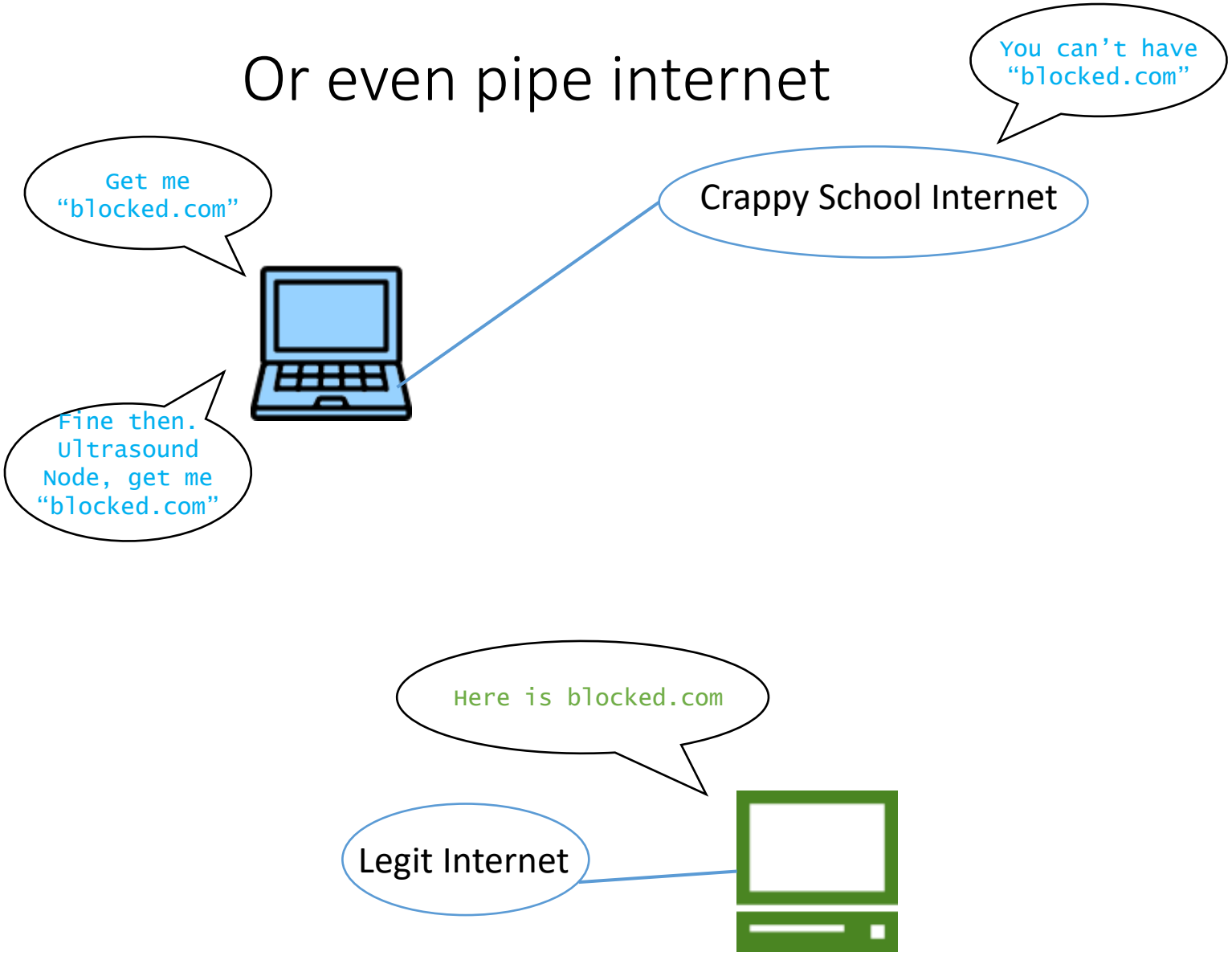
Or even pipe internet



Or even pipe internet



Or even pipe internet



Messages are untraceable (except by literally following the sound, which is hard) and above the hearing ranges of most people above 25

And who knows, maybe we won’t get caught this time.

(LOL JK)

Time for a demo!