FREQUENCY HOPPING SPREAD SPECTRUM

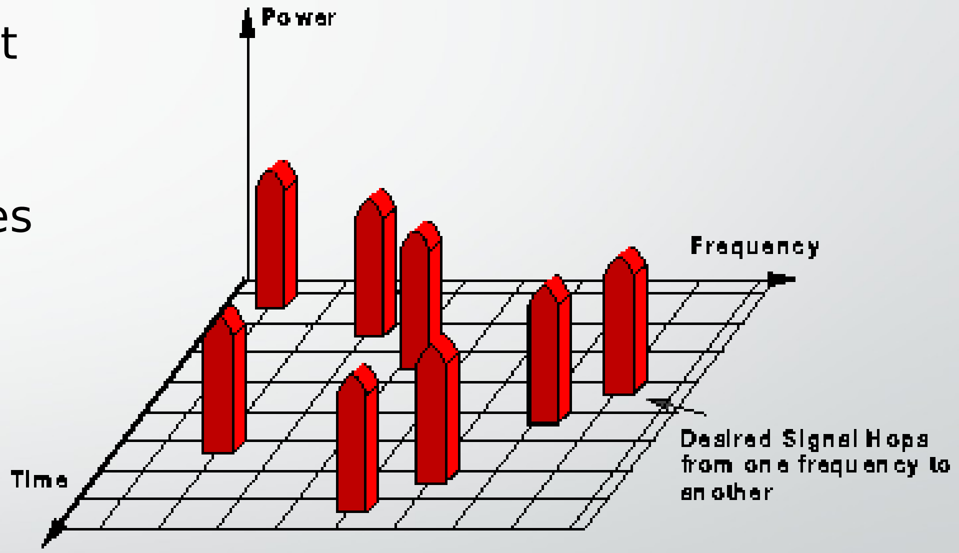
HARSHVARDHAN TIBREWAL   AGRIM GUPTA

INSPIRATION

- Previously used by us while implementing XBee network for a personal project. The Xbee module hopped over various frequencies in the start to select the frequency with the lowest SNR.
- Has wide usage in Bluetooth and Wi-Fi technologies, like in wireless mouse. Bluetooth utilizes frequency-hopping spread spectrum technology to avoid interference problems. The signal switches carrier channels rapidly, at a rate of 1600 hops per second, over a determined pattern of channels

INTRODUCTION

- Different parts of the message are sent out at different carrier frequencies
- A good algorithm to generate carrier frequencies in a random manner is by using Pseudo Random Sequences
- PN sequences have a property that autocorrelation is very low unless they overlap. (We will exploit this property in acquisition)
- Retrieval will be in two stages:
- Acquisition: Our block will identify which frequency is being transmitted using the autocorrelation property
- Tracking: Once we know the frequency, using our PN sequence we can track next frequencies and retrieve message signal without data loss

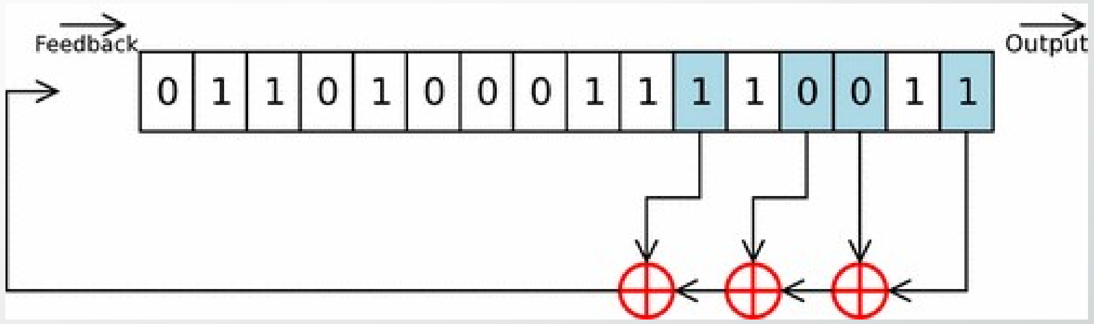


Modulation

- We are using Phase modulation
- The message is spread in time and parts are placed in the carrier, its carrier frequency hopping periodically
- Phase modulation has more resistance for additive noise than DSB-SC

Pseudo Random Sequences

- We are implementing Random sequences using Linear Feedback Shift registers
- The bit sequence in the system at any state decides which carrier frequency is going to be used for transmission
- These sequences repeat at regular intervals hence periodically hopping in frequencies
- Although the PN sequence is completely deterministic given the initial seed, it shows many properties of random sequences, thus it’s name
- PN sequences are unique to the receiver and transmitter and hence cannot be tracked easily



Transmitter

- At the Transmitter end, we have the PM modulation block, but with a twist. The carrier frequency is being hopped continuously, through the hopping block.
- We have made a complete block “Mod_64” for the same (PM modulation and Hopping)

At the receiver’s end

- Our block has:
  - The acquisition and tracking system used for detection of carrier of frequencies
  - The demodulation scheme for phase modulation
- Input to the Block is the transmitted signal and output is the demodulated message signal

Serial Acquisition Technique

- We evaluate autocorrelation of the incoming samples with each frequency and take the highest one (above a threshold), which gives us the “Locked On Frequency”
- Procedure: <multiply conjugate, square, integrate >
- Timing issues: how many samples does each frequency require to give a successful autocorrelation?
- Accuracy and redundancy of the autocorrelation integral saves the day
- The Tracking stage has a periodic down counter of 10 cycles of hop (i.e. after 10 full cycles of hopping, the code goes to acquisition part again, for periodic updates)

Limitations to Serial Acquisition

- Since we need at a minimum of 30 samples to detect a frequency, number of frequencies we can check before the signal is over is…
- X=n*S/f; where n is number of 2pi cycles, S –Sampling rate, f- max frequency, X is # of samples (X<8192, GNURadio Constraint)
- This gives us at best 64 hops, since for 128 hops number of 2pi cycles reduce to 16, which hinders the autocorrelation (eg Bluetooth)
- Hence an attacker is limited to checking only a certain number of frequencies in a limited period of time (order of 200msec)
- Without the knowledge of PN sequence it is almost impossible to tract FHSS

Tracking

- Once frequency acquired, we use that frequency as input to the PN sequence and start generating next frequencies
- Our acquisition runs periodically (self.counter) during this stage because if there are changes (due to noise or wrong transmissions) it will **adapt automatically**

Demodulation

- We demodulate the PM wave in house and output the message
- Procedure:
  - Convert float to complex of the transmitted signal - in the polar form $Ae^{j(2*pi*fc*t + Kp*m(t))}$
  - Now remove the carrier frequency (by multiplication with conjugate)
  - What’s remaining - our Message!

PROBLEMS FACED IN GNURADIO CODING

- BUFFER:
  - Gnuradio randomly changes buffer transmission rate , and as our signal is crucial in variations of frequencies and time we cannot acquire them easily
- DELAY IN BLOCKS:
  - Blocks in series create a significant delay, which hinders synchronization within the flowgraph

The Buffer Problem

- One very big challenge faced by us is the buffer rate of the block giving the transmitted frequency to the receiver
- Since buffer rate keeps on changing **randomly (varying from any number between 0 and 8192)**, we cannot be sure that what we have received in the current set of samples contains just 1 frequency or 2 frequencies (ex : half set of f1 and second half set of of f2)
- If we directly demodulate there is a high chance that we lose data if we cannot identify which frequency its is on (especially in the case with 2 frequencies in one set)
- Hence Acquisition is difficult as **synchronization** is the major task!

We have a solution!

- We have an internal buffer we save previous samples.
- Once the internal buffer overflows, it gives the set of samples to the acquisition
- We now know that only one frequency can be contained in the set of samples and proceed to acquisition and tracking