DSSS and FHSS
A comparison of two modulation schemes

Group 6: Urs Gerber, Gian-Luca Mateo, Niclas Scheuing, Roger Stebler

University of Berne

1 Protocol Introduction

Spread Spectrum is a procedure to spread a transmission over a wider frequency range. The goal is robustness and support for parallel transmissions. Refer to [ISS] for more details.

Both, *FHSS* and *DSSS*, require the sender and receiver to be time-synchronized.

1.1 Frequency Hopping Spread Spectrum

In *FHSS* the frequency band f is split into N sub-bands $f_1, ..., f_N$. Each sender gets a pseudo-random chip sequence $p_n \in [f_1, ..., f_N]^n$ which is known by the sender and the receiver only.

Slow hopping The sender will start by taking the first sub-band from its chip sequence p_n and transmit on this sub-band. After a defined dwell time, it takes the next sub-band from p_n and continue this procedure. The dwell time T_h is a multiple of the symbol time T_s. T_s is the time during which one symbol is broadcast.

Fast hopping In this case the symbol time T_s is a multiple of the dwell time T_h. During the transmission of one symbol, one or multiple hops are made. So one symbol is transmitted on multiple frequencies.

Interference FHSS is very robust under narrow-band interference. If one sub-channel is jammed, this will only affect part of the transmission. Any other user sending with the same FHSS scheme will appear as narrow-band interference. Broad-band interference can not be dealt with easily.

1.2 Direct Sequence Spread Spectrum

Each sender has a chip sequence $p_n \in \{-1, 1\}^n, \sum p_n = 1$ known by the sender and receiver only. The chip-rate R_c is a multiple of the signal-rate R_s.

Spreading Before sending the data sequence d_t, it is multiplied point-wise with the chip-pattern resulting in the transmission data $t_x = d_t p_n$. Image 7 shows an example of these steps. This multiplication is called *spreading* because the frequency of the original data R_s has been increased to the frequency of the chip pattern R_c. This can be seen on the right side of Figure 7.

Despreading is the inverse of the spreading operation. The received signal r_x is again point-wise multiplied with the chip pattern resulting in the despread data. Since $p_n p_n = [1, ..., 1]$, this will return the original data $d_r = t_x p_n = d_t p_n p_n = d_t$.

Chip pattern The length of the chip pattern T_c is either the symbol length T_s (short code) or a multiple of the T_s (long code).

When multiple users are transmitting on the same frequency range, their pattern have to be orthogonal.

Interference Narrow-band interference will be spread during the despreading and have little impact on the SNR. Broad-band signals uncorrelated to the chip pattern will not be affected by the despreading and thus have a higher impact on the SNR. See Figure 8.

2 Experiment Setup

In the following sections we introduce the general experiment setup as well as the various test scenarios and parameters that were used to obtain the final results.

2.1 Test Architecture

Our simulation of the spread spectrum mechanisms is based on a simple pipeline, where a *sender* is writing its data onto a *medium*. Various kinds of interference is generated by a *jammer* which adds its data to the same medium. Finally the *receiver* reads data from the medium and tries to decode the original bit sequence.

Sender The sender is responsible for spreading a random data sequence consisting of 0s and 1s using either the *DSSS* or *FHSS* spreading mechanism as explained in section 1. After spreading, the sender will modulate the spread data onto passband using the BPSK modulation scheme (using a phase-shift of $\Delta\phi = \pi/2$) with a carrier frequency f_c in the case of DSSS. With FHSS, the sender generates a set of $N = 8$ sub-carrier frequency bands of bandwidth Δf. The modulated signal is then passed to the *medium*

Medium After a signal is sent to the medium, we perform a Fast Fourier Transform (FFT) of the data and store the result. A jammer may then add an interference signal onto the original data, depending on the respective test scenario. Whenever a receiver reads data from the medium, we perform an inverse FFT and pass the data to the receiver.

Receiver The receiver first reads the data off the medium and applies the respective demodulation scheme according to the scheme that was used to send the data in the first place. If DSSS was used, the receiver simply demodulates the whole signal using BPSK and despreads the signal using the same pn-sequence that the sender used. In the case of FHSS the receiver splits the data into slices equal in length to a chip – depending on the channel number determined by the pn-sequence. Each chip is then demodulated using BPSK and decoded.

Jammer We used multiple approaches to designing our jammer. The most reliable method – the method we used in our results – was designing the jammer as just another DSSS user whose bandwidth we can control using the chipping rate. One of the most difficult aspects of this simulation was the power control of this jammer with respect to the signal power that is already on the medium. As a result, we always specify the power of the jammer as a power factor p_f relative to the total power of the signal to be jammed.

2.2 Testing Scenarios and Parameters

The used test metric was solely the relative bit error rate BER. This was computed as the number of bit errors per sent data sequence BE divided by the length of the data sequence.

$$BER = \frac{BE}{length(data)}$$

Each test was repeated 20 times, yielding an effective average over all test runs.

Throughout all test scenarios, the carrier frequency for all DSSS tests remained $f_c = 100$Hz. The channel configuration for FHSS is also constant, with the $N = 8$ sub-carriers starting at frequency $f_c = 100$Hz with a channel bandwidth of $\Delta f = 128$Hz, resulting in sub-carriers at $f_i = (100 + i \cdot 128)Hz, i = 1, .., N$ The symbol rate was fixed at $f_{sym} = 8$Hz. In each test, white Gaussian noise with an SNR of 10dB was added to the time domain signal to simulate interference of various origins.

Each test scenario was conducted with the same variable parameters consisting of the chipping rate f_{chip} and the length of the pn-sequence l_{pn}. For both FHSS and DSSS, we used $l_{pn} = (16, 64)$. Initial tests showed that these are sensible values. The chipping rates used in DSSS were $f_{chip}^{DSSS} = (16, 32, 64, 96)$Hz, in FHSS $f_{chip}^{FHSS} = (1, 4, 16, 32)$ respectively. Combined, the two paremeter sets l_{pn} and f_{chip} form eight curves in each plot of a test scenario.

Narrowband Interference In this scenario we tested the performance of both spreading schemes under the influence of narrowband interference. For DSSS, we jammed the $f_{jam} = 100$Hz carrier with a noise bandwidth of $\Delta f_{jam} = 8$Hz at various noise power factors p_f from 0 to 7 in 0.5 step increments. For FHSS, we successively jammed each sub-carrier channel, from 0 to all 8 channels with a noise bandwidth of $\Delta f_{jam} = 64$Hz, as we wanted to jam a whole sub-carrier channel. The power factor for each jam channel was set to $p_f = 2.5$.

Wideband Interference Wideband interference is the core weakness of both spread specrum techniques – a fact that we wanted to simulate. The DSSS test run was conducted using a jamming bandwidth of $\Delta f_{jam} = 200$Hz again targeted at $f_{jam} = 100$Hz. The jamming power factor was scaled from 1 to 15 to simulate various strengths of wideband interference. For FHSS, a noise bandwidth of $\Delta f_{jam} = 1000$Hz was used and the jammed frequency was moved close to the center of the sub-carriers $f_{jam} = 550$Hz. The noise power factor was scaled from 2 to 4 in 0.5 increments. We expected both schemes to handle wideband interference equally bad.

Varying Noise Bandwidth An interesting scenario represents the performance of the spreading schemes in presence of various noise bandwidths. With our jammer design, this kind of comparison was very hard to achieve since a change in bandwidth also meant a change in power. We had to also adapt the power factor for each tested bandwidth to obtain comparable results. For both DSSS and FHSS we used $\Delta f_{fam} = (8, 16, 32, 64, 128, 256)$Hz with a matching power factor $p_f = (1, 2, 2.5, 3, 3.5, 4)$. These power factors were obtained by visually matching in all conscience the noise power density to figures given in [ISS] and are by no means exact. The jammed frequency was $f_{jam} = 100$Hz for both schemes.

Multiple Users In this scenario we tested how well DSSS and FHSS handle multiple users on the same frequency band. In both cases, we scaled the number of users from 1 to 15. We expected FHSS to handle this situation better than DSSS, since additional FHSS users appear as narrowband interference. The sender power for each user was kept the same.

Varying strength of white Gaussian noise In this last scenario we tested the performance of DSSS and FHSS in presence of average white Gaussian noise of various strengths. The SNR was scaled from -2dB to -10dB in the case of FHSS and from -8dB to -30dB for DSSS. We did not expect any scheme to perform substantially better.

3 Results and Analysis

Using the test setup and cases shown above, we were able to gain the following insights.

3.1 DSSS

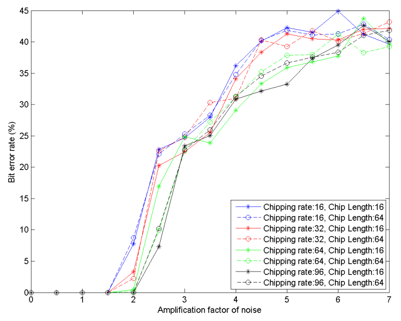
Performance in presence of narrowband and wideband interference
Figure 1a shows error rates of DSSS under the influence of narrowband noise. We observe that higher *chipping rates* - which lead to broader spreading - add robustness. This is due to the narrowband interference getting despread more strongly when employing higher chipping rates. The *chip sequence length* has little to no influence, which also matches our expectations, as the power spectrum

does not change with different chip length. Longer chip sequences do, hovewer, add support for more users.

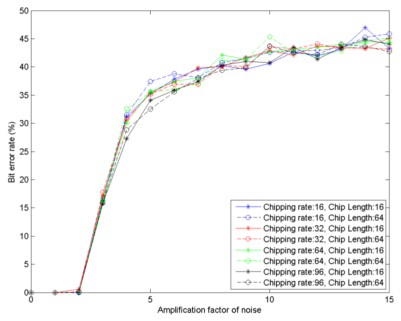
Figure 1b illustrates behaviour under the influence of wideband noise, which is nearly identical for all our different testing parameters. We would intuitively have expected low *chipping rates* to have better performance - due to their more narrow frequency spectrum and thus higher peak power - but this line of thought ignores that wideband noise is not correllated with the sent signal, and so does not get despread. Thus, it remains on the same power level after despreading, which results in the invariability of performance under changing parameters.

Due to our testing setup and methodology, direct comparison between the two scenarios is tedious. As our jammer implementation puts out a constant base power when measured over the whole spectrum, the SNRs tend to vary a lot for different scenarios. Because of that, we have chosen to regard narrowband and wideband noise as separate cases and do a comparison in a separate test case, where we tried to visually match the signal and noise spectra of the scenarios outlined in [ISS]. The results of that test are shown in 2a and discussed in the next section.

Fig. 1



(a) DSSS Narrowband noise: $\Delta f = 8\text{Hz}, f_{jam} = 100\text{Hz}$



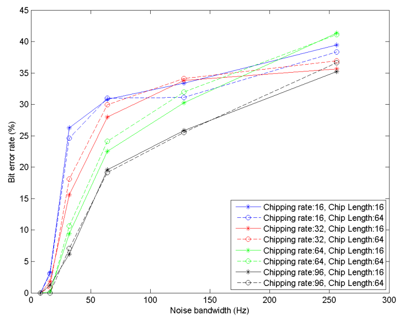
(b) DSSS Wideband noise: $\Delta f = 200\text{Hz}, f_{jam} = 100\text{Hz}$

Performance with varying interference bandwidth and white Gaussian noise

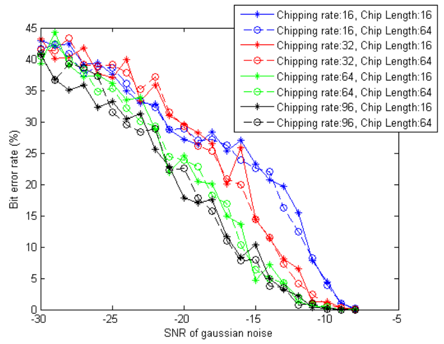
Figure 2a shows the results of the aforementioned test case. *Wideband noise* has a bigger impact on performance than *narrowband noise* with high *chipping rates* showing better performance than low ones. While we are very aware that the parameters for this configuration may seem arbitrary, we struggled to find reliable reference values. The result of this is that the results should be seen as a qualitative estimation of real-world performance of DSSS under presumably realistic levels and bandwidths of interference and not as exact values.

Figure 2b outlines performance of our DSSS implementation with varying levels of white Gaussian noise. The results widely match expected behaviour, with performance deteriorating with more added noise. Unexpectedly, performance was worse when employing low chipping rates. We suspect this may be due to the way MATLAB measures the SNR when using the *awgn* function.

Fig. 2



(a) DSSS: Various noise bandwidths and powers: $p_f = (1, 2, 2.5, 3, 3.5, 4)$



(b) DSSS: Various SNR of white Gaussian noise

Performance with multiple users

In Figure 3 we see that low *chipping rates* work considerably worse than high chipping rates, as the signal is spread less. This matches expected behaviour.

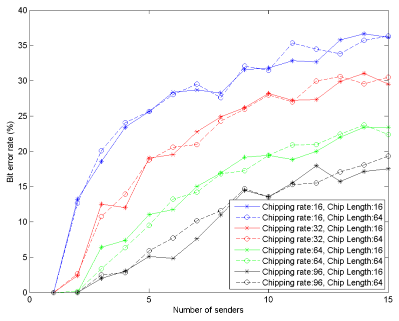


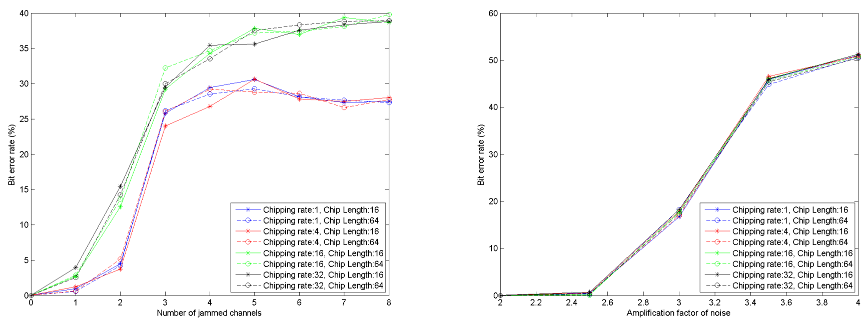
Fig. 3: DSSS: Multiuser

3.2 FHSS

Before going into the exact test cases, it is important to note that there is a problem with our implementation of fast frequency hopping, as it works badly with BPSK modulation. Various websites such as [wirelesscommunication.nl] emphasize the difficulty of using coherent data detection and that FSK is most often used. However, we followed our requirements and implemented BPSK.

Performance in presence of narrowband and wideband interference
The results widely match expected behaviour, with performance deteriorating the more channels are jammed and wideband noise affecting all channels - and thus all configurations - equally. Importantly, we were not able to create configurations without crosstalk between the different channels, so this will have influenced all results.

Fig. 4

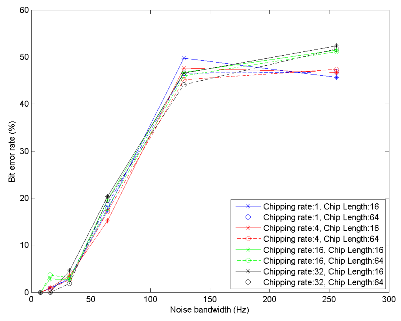


(a) FHSS: Narrowband noise: $\Delta f = 64\text{Hz}, p_f = 2.5$

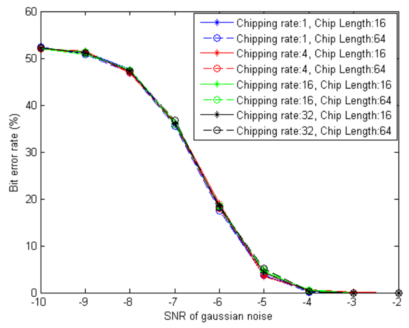
(b) FHSS: Wideband noise: $\Delta f = 1000\text{Hz}, f_{jam} = 550\text{Hz}$

Performance with varying interference bandwidth and white Gaussian noise
The results shown in Figure 5a again represent an estimation of performance under different - and presumably realistic - noise configurations. The results are similar to the ones from DSSS, with better performance under the influence of narrowband interference. Figure 5b shows a more streamlined result than DSSS, which is caused by the bandwidths and channel configurations being invariable under the different parameter configurations.

Fig. 5



(a) FHSS: Various noise bandwidth and power: $p_f = (1, 2, 2.5, 3, 3.5, 4)$



(b) FHSS: Various SNR of white Gaussian Noise

Performance with multiple users
The results shown match expected behaviour. The performance deteriorates quickly and bottoms at 40-50 % BER when the number of users approaches the number of channels.

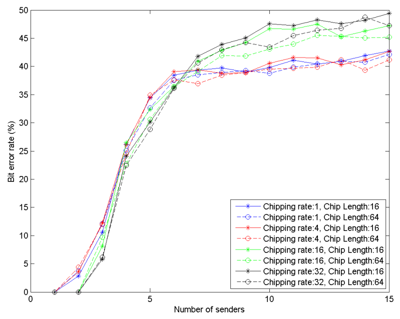


Fig. 6: Multiuser

4 Conclusion

Due to the different bandwidths both techniques use, a direct comparison is not straightforward. We can, however, state that DSSS reaches the same performance in multiuser scenarios as our FHSS implementation while using a chipping rate of 16. This corresponds to a bandwidth of 32 Hz, where FHSS employed 1024 Hz. Our DSSS implementation also performed better in our ”real-life noise” test (See “Performance with varying interference bandwidth and white

Gaussian noise") and was more robust to Gaussian noise. FHSS performed better in our narrowband interference test scenario and is probably more flexible to scale up to more users.

We could have rediscussed using FSK for fast-hopping in FHSS, as we think we could not properly show the potential of this technique with BPSK.

The implementation of the jammer has been challenging and though we tried several different approaches, the result is not fully satisfying. A way to jam a certain band using a certain SNR or power density spectrum would certainly be practical and would possibly yield better results. But this was outside the time frame of this project and we also lack the background in signal processing to properly implement it.

Further work could include support for simulation of free-space path loss, where the distance between the sender and the receiver would influence the quality of the signal. Additionally we could extend the free-space path loss by using a custom propagation coefficient to simulate more complex environments like buildings and other obstacles that prevent direct line of sight.

Another project could be to not only measure bit-error rates, but also introduce a packet-error rate. To support this we would first need to decide how a packet should be defined. One possibility would be to combine a fixed amount of bits to a packet or send a special bit sequence to announce the beginning of a new packet. Additionally, some sort of error correction scheme would be necessary to discuss this approach.

A Appendix

A.1 Protocol Introduction

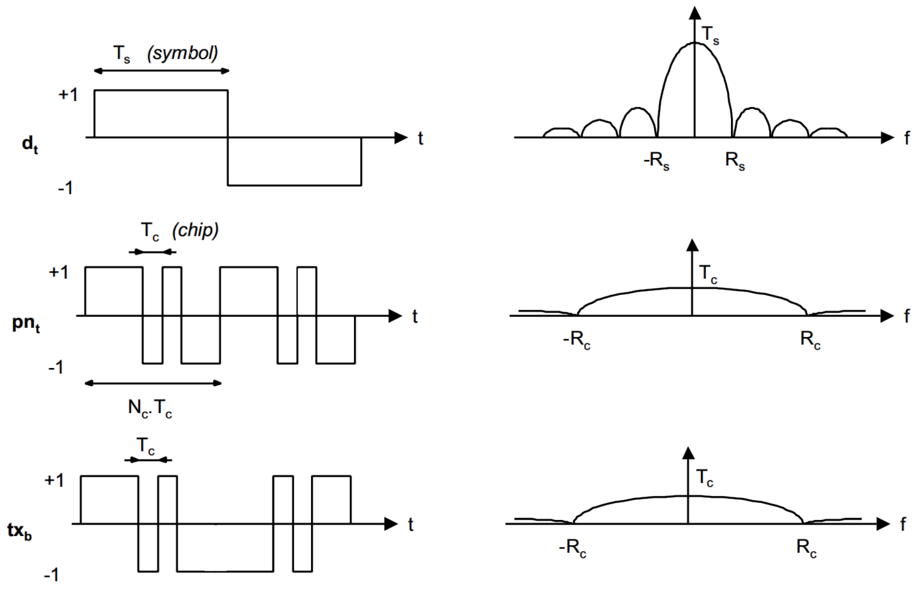


Fig. 7: The spreading steps of DSSS. Top: Data sequence d_t Middle: Chip pattern p_n Bottom: The point-wise multiplied transmission signal $t_x = d_t p_n$

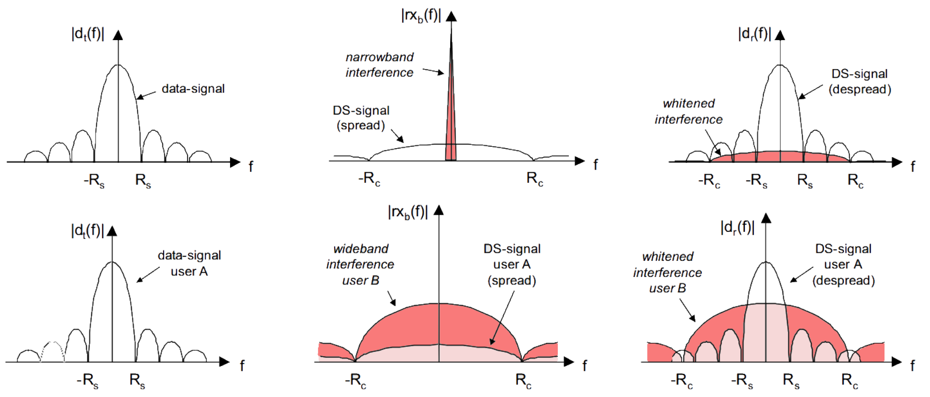


Fig. 8: DSSS with interference. Top: Narrow-band interference is spread during the despreading. Bottom: Broad-band interference is not affected by the despreading.

References

[ISS] Author: Jan De Nayerlaan Article: Spread Spectrum Year: (1999)

[wirelesscommunication.nl] Glas, Jack. ”Frequency Hopping.” Frequency Hopping. Wirelesscommunication.nl, n.d. Web. 07 Dec. 2014. http://www.wirelesscommunication.nl/reference/chaptr05/spreadsp/fh.htm