Chapter 2. Synchronization

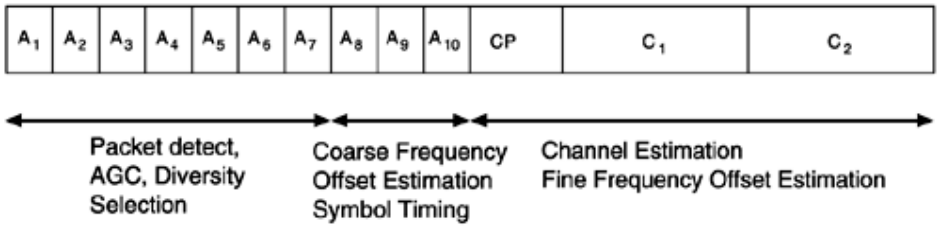
IN THIS CHAPTER

- Timing Estimation
- Frequency Synchronization
- Channel Estimation
- Clear Channel Assessment
- Signal Quality
- Bibliography

Synchronization is an essential task for any digital communication system. Without accurate synchronization algorithms, it is not possible to reliably receive the transmitted data. From the digital baseband algorithm design engineer's perspective, synchronization algorithms are the major design problem that has to be solved to build a successful product.

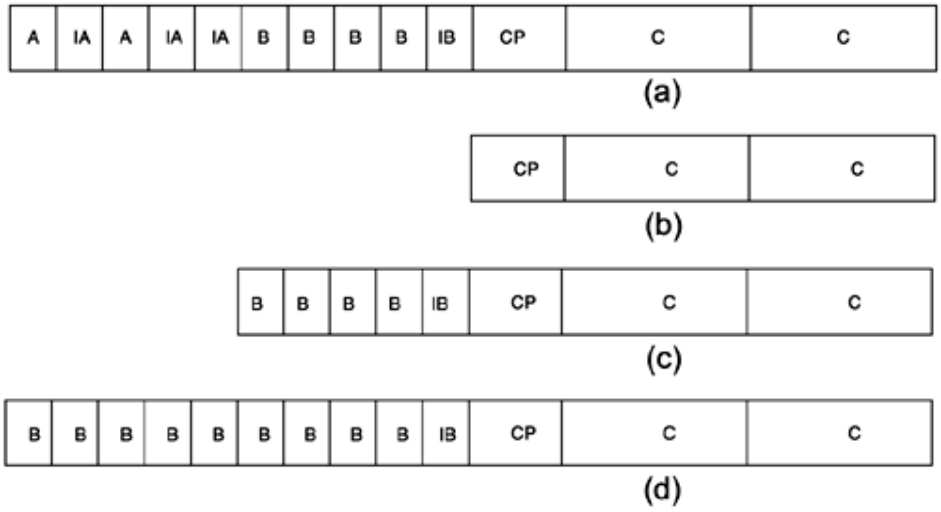
OFDM is used for both broadcast type systems and packet switched networks, like WLANs. These two systems require somewhat different approach to the synchronization problem. Broadcast systems transmit data continuously, so a typical receiver, like European Digital Audio Broadcasting (DAB) or Digital Video Broadcasting (DVB) system receivers, can initially spend a relatively long time to acquire the signal and then switch to tracking mode. Speth et al. [24] and [25] analyze synchronization techniques for broadcast receivers. On the other hand, WLAN systems typically have to use so called "single-shot" synchronization; that is, the synchronization has to be acquired during a very short time after the start of the packet. This requirement comes from the packet switched nature of WLAN systems and also from the high data rates used. To achieve good system throughput, it is mandatory to keep the receiver training information overhead to the minimum. To facilitate the single-shot synchronization, current WLAN standards include a preamble, like the IEEE 802.11a [9] preamble shown in Figure 2.4 or the various HiperLAN/2 [7] preambles shown in Figure 2.7, in the start of the packet. The length and the contents of the preamble have been carefully designed to provide enough information for good synchronization performance without any unnecessary overhead.

Figure 2.4. The IEEE 802.11a standard preamble.[*]



[*] *From IEEE Std. 802.11a-1999. Copyright 1999 IEEE. All rights reserved.*

Figure 2.7. Different preambles of the HiperLAN/2 standard.[*]



The OFDM signal waveform makes most of the synchronization algorithms designed for single carrier systems unusable, thus the algorithm design problem has to be approached from the OFDM perspective. This distinction is especially visible on the sensitivity difference to various synchronization errors between single carrier and OFDM systems. The frequency domain nature of OFDM also allows the effect of several synchronization errors to be explained with the aid of the properties of the Discrete Fourier Transform (DFT). Another main distinction with single carrier systems is that many of the OFDM synchronization functions can be performed either in time- or frequency-domain. This flexibility is not available in single carrier systems. The trade-offs on how to perform the synchronization algorithms are usually either higher performance versus reduced computational complexity.

The scope of this book is WLAN systems; hence this chapter concentrates on WLAN synchronization algorithms. Occasionally we comment on differences between WLAN and broadcast system synchronization algorithms.

The order of the algorithms described in this chapter follows to some extent the order of how an actual receiver would perform the synchronization. The main assumption usually made when WLAN systems are designed is that the channel impulse response does not change significantly during one data burst. This assumption is justified by the quite short time duration of transmitted packets, usually a couple milliseconds at maximum and that the transmitter and receiver in most applications move very slowly relatively to each other. Under this assumption most of the synchronization for WLAN receivers is done during the preamble and need not be changed during the packet.

Timing Estimation

Timing estimation consists of two main tasks: packet synchronization and symbol synchronization.

The IEEE 802.11 MAC protocol is essentially a random access network, so the receiver does not know exactly when a packet starts. The first task of the receiver is to detect the start of an incoming packet. HiperLAN/2 medium access architecture is a hybrid of both random access and time scheduled networks. Thus HiperLAN/2 receivers also have to be able to reliably detect the start of incoming packets, without prior knowledge.

Broadcast systems naturally do not require packet detection algorithms, because transmission is always on. However, for a packet oriented network architecture, finding the packets is obviously of central importance for high network performance.

Packet Detection

Packet detection is the task of finding an approximate estimate of the start of the preamble of an incoming data packet. As such it is the first synchronization algorithm that is performed, so the rest of the synchronization process is dependent on good packet detection performance. Generally packet detection can be described as a binary hypothesis test. The test consists of two complementary statements about a parameter of interest. These statements are called the null hypothesis, H_0, and the alternative hypothesis, H_1. In the packet detection test, the hypotheses assert whether a packet is present or not. The test is set up as shown below.

$$H_0 : \ Packet\ not\ present$$

$$H_1 : \ Packet\ present$$

The actual test is usually of the form that tests whether a decision variable m_n exceeds a predefined threshold *Th.* The packet detection case is shown below.

$$H_0 : m_n < Th \Rightarrow Packet\ not\ present$$

$$H_1 : m_n \geq Th \Rightarrow Packet\ present$$

The performance of the packet detection algorithm can be summarized with two probabilities: probability of detection P_D and probability of false alarm P_{FA}. P_D is the probability of detecting a packet when it is truly present, thus high P_D is a desirable quality for the test.P_{FA} is the probability that the test incorrectly decides that a packet is present, when actually there is none, thus P_{FA} should be as small as possible. In general, increasing P_D increases P_{FA} and decreasing P_{FA} decreases P_D, hence the algorithm designer must settle for a some balanced compromise between the two conflicting goals. The general hypothesis test problem is discussed in several statistical inference and detection theory books [3, 10].

Generally it can be said that a false alarm is a less severe error than not detecting a packet at all. The reason is that after a false alarm, the receiver will try to synchronize to nonexistent packet and will detect its error at the first received data integrity check. On the another hand, not detecting a packet always results in lost data. A false alarm can also result in lost data, in case an actual data packet starts during the time the receiver has not yet detected its mistake. In this case, the receiver will not be able to catch that packet. The probability of this occurring depends on several issues like the network load and the time it takes for the receiver to detect its mistake. In conclusion, a little higher P_{FA} can be tolerated to guarantee good P_D. In the next sections, we introduce several approaches to the packet detection test design problem.

Received Signal Energy Detection

The simplest algorithm for finding the start edge of the incoming packet is to measure the received signal energy. When there is no packet being received, the received signal r_n consists only of noise $r_n = w_n$. When the packet starts, the received energy is increased by the signal component $r_n = s_n + w_n$, thus the packet can be detected as a change in the received energy level. The decision variable m_n is then the received signal energy accumulated over some window of length L to reduce sensitivity to large individual noise samples.

Equation 2.1

$$m_n = \sum_{k=0}^{L-1} r_{n-k} r_{n-k}^* = \sum_{k=0}^{L-1} |r_{n-k}|^2$$

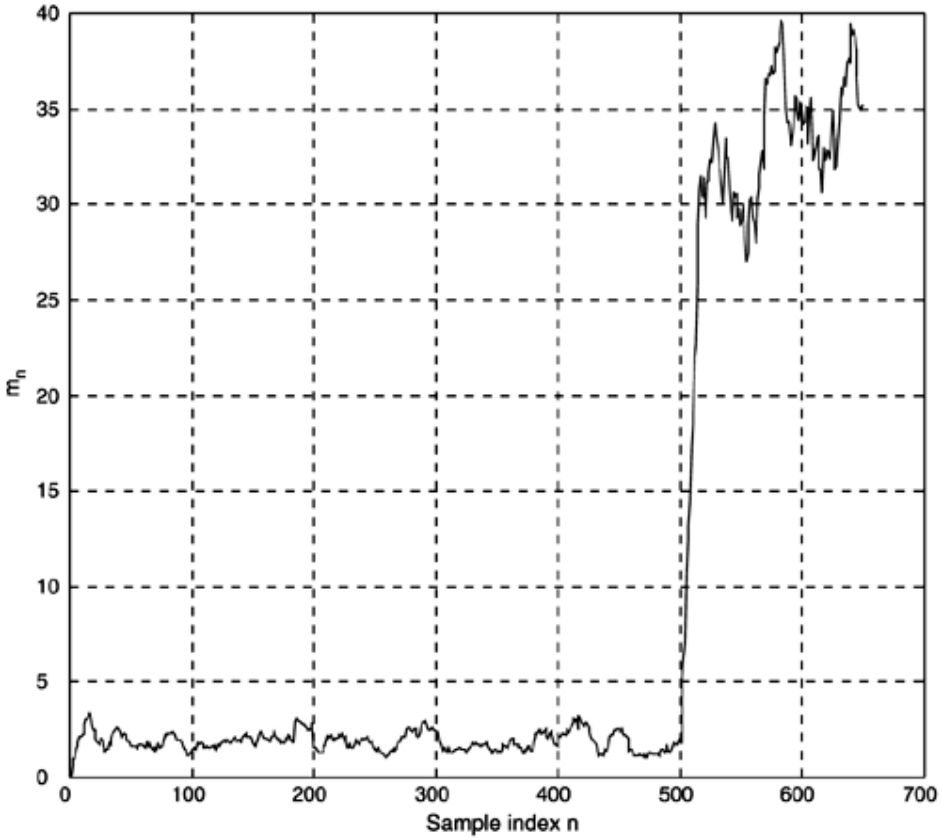
Calculation of m_n can be simplified by noting that it is a moving sum of the received signal energy. This type of sum is also called a sliding window. The rationale for the name is that at every time instant *n,* one new value enters the sum and one old value is disgarded. This structure can be used to simplify the computation of Equation 2.1. Equation 2.2 shows how to calcute the moving sum recursively.

Equation 2.2

$$m_{n+1} = m_n + |r_{n+1}|^2 - |r_{n-L+1}|^2$$

Thus the number of complex multiplications is reduced to one per received sample; however, more memory is required to store all the values of $|r_n|^2$ inside the window. The response of this algorithm is shown in Figure 2.1. The figure shows the value m_n for IEEE 802.11a packet with 10dB Signal to Noise Ration (SNR) and sliding window length $L = 32$. The true start of the packet is at $n = 500$, thus in this case the threshold could be set between 10 and 25. The value of the threshold defines the P_D and P_{FA} of the test.

Figure 2.1. Received signal energy based packet detection algorithm.

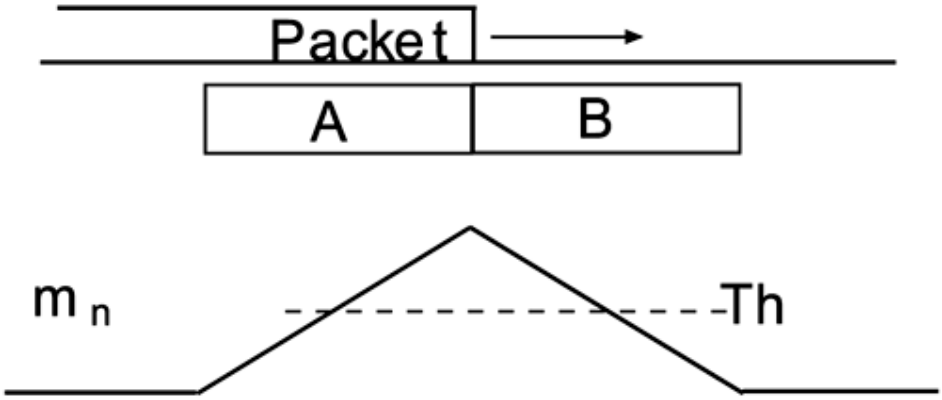


This simple method suffers from a significant drawback; namely, the value of the threshold depends on the received signal energy. When the receiver is searching for an incoming packet, the received signal consists of only noise. The level of the noise power is generally unknown and can change when the receiver adjusts its Radio Frequency (RF) amplifier settings or if unwanted interferers go on and off in the same band as the desired system. When a wanted packet is incoming, its received signal strength depends on the power setting of the transmitter and on the total path loss from the transmitter to the receiver. All these factors make it quite difficult to set a fixed threshold, which could be used to decide when an incoming packet starts. The next section describes an improvement to the algorithm that alleviates the threshold value selection problem.

Double Sliding Window Packet Detection

The double sliding window packet detection algorithm calculates two consecutive sliding windows of the received energy. The basic principle is to form the decision variable m_n as a ratio of the total energy contained inside the two windows. Figure 2.2 shows the windows *A* and *B* and the response of m_n to a received packet. In Figure 2.2, the *A* and *B* windows are considered stationary relative to the packet that slides over them to the right. It can be seen that when only noise is received the response is flat, since both windows contain ideally the same amount of noise energy. When the packet edge starts to cover the *A* window, the energy in the *A* window gets higher until the point where *A* is totally contained inside the start of the packet. This point is the peak of the triangle shaped m_n and the position of the packet in Figure 2.2 corresponds to this sample index *n*. After this point *B* window starts to also collect signal energy, and when it is also completely inside the received packet, the response of m_n is flat again. Thus the response of m_n can be thought of as a differentiator, in that its value is large when the input energy level changes rapidly. The packet detection is declared when m_n crosses over the threshold value *Th*.

Figure 2.2. The response of the double sliding window packet detection algorithm.



Equation 2.3 shows the calculation of the *A* window value and Equation 2.4 the calculatisssson for *B* window.

Equation 2.3

$$a_n = \sum_{m=0}^{M-1} r_{n-m} r_{n-m}^* = \sum_{m=0}^{M-1} |r_{n-m}|^2$$

Equation 2.4

$$b_n = \sum_{l=1}^{L} r_{n+l} r_{n+l}^* = \sum_{l=0}^{L} |r_{n+l}|^2$$

Both a_n and b_n are again sliding windows, thus the computation can be simplified in the same recursive manner as for the energy detection window. Then the decision variable is formed by dividing the value of the a_n by b_n

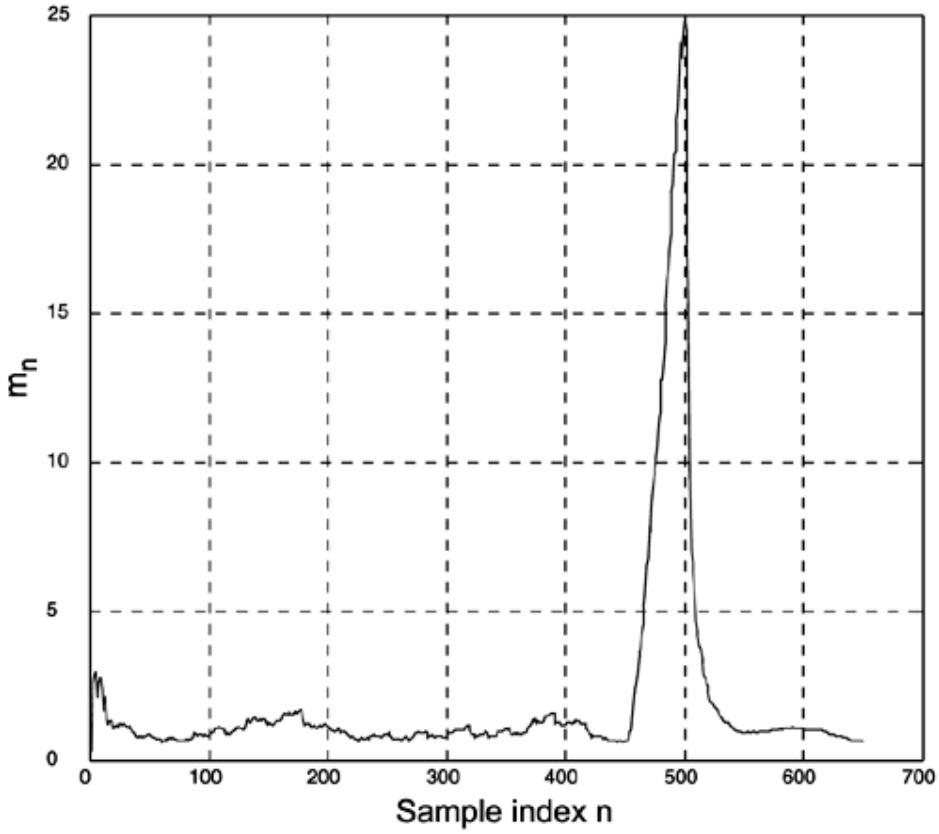
Equation 2.5

$$m_n = \frac{a_n}{b_n}$$

A simulated response of the variable m_n is shown in Figure 2.3. The figure is again for the IEEE 802.11a preamble with 10dB SNR. The figure clearly shows how the value of m_n does *not* depend on the total received power. After the peak, the response levels off to the same value as before the peak, although the received energy level is much higher. An additional benefit of this approach is that, at the peak point of m_n the value of a_n contains the sum of signal energy *S* and noise energy *N* and the b_n value is equal to noise

energy *N,* thus the value of m_n at the peak point can be used to estimate the received SNR. Equation 2.6 is the ratio a_n and b_n at the peak point and Equation 2.7 is the SNR estimate calculated from the ratio.

Figure 2.3. Double sliding window packet detection.



Equation 2.6

$$m_{peak} = \frac{a_{peak}}{b_{peak}} = \frac{S+N}{N} = \frac{S}{N} + 1$$

Equation 2.7

$$\widehat{SNR} = m_{peak} - 1$$

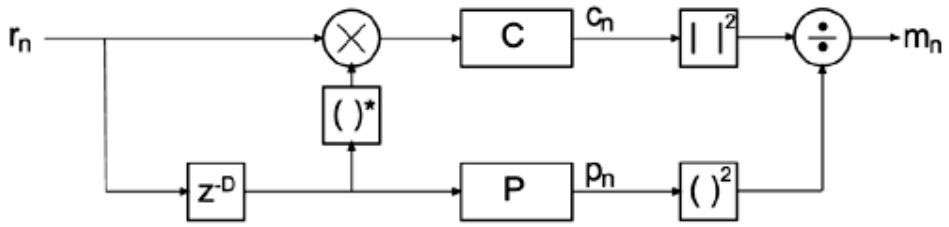
Using the double sliding window algorithm is a good approach, if the receiver does not have additional information about the received data. However, if the receiver does have additional information, more can be done as is explained in the following section.

Using the Structure of the Preamble for Packet Detection

A general communications system engineering principle is that the receiver should use all the available information to its advantage. In the packet detection algorithm, this means that the known structure of the preamble should be incorporated into the algorithm. The preambles of both the IEEE 802.11a and HiperLAN/2 systems have been designed to help the detection of the start edge of the packet. The IEEE 802.11a standard gives guidelines on how to use the various segments of the preamble to perform the necessary synchronization functions. The preamble structure and the guidelines are illustrated in Figure 2.4. The parts from A_1 to A_{10} are short training symbols, that are all identical and 16 samples long. *CP* is a 32-sample cyclic prefix that protects the long training symbols C_1 and C_2 from intersymbol interference (ISI) caused by the short training symbols. The long training symbols are identical 64 samples long OFDM symbols. However, the guidelines are not binding requirements of the standard. The design engineer has the freedom to use any other available method or develop new algorithms.

The structure of the WLAN preamble enables the receiver to use a very simple and efficient algorithm to detect the packet. The following approach was presented in Schimdl and Cox [21] for acquiring symbol timing, but the general method is applicable to packet detection. Overall the method resembles the double sliding window algorithm introduced in the previous section, but it takes advantage of the periodicity of the short training symbols at the start of the preamble. This approach is called the delay and correlate algorithm. The name and the relation to the double sliding window method is illustrated in Figure 2.5, which shows the signal flow structure of the delay and correlate algorithm. The Figure shows two sliding windows *C* and *P*. The *C* window is a crosscorrelation between the received signal and a delayed version of the received signal, hence the name delay and correlate. The delay z^{-D} is equal to the period of the start of the preamble; for example, IEEE 802.11a has $D = 16$, the period of the short training symbols. The *P* window calculates the received signal energy during the crosscorrelation window. The value of the *P* window is used to normalize the decision statistic, so that it is not dependent on absolute received power level.

Figure 2.5. Signal flow structure of the delay and correlate algorithm.



The value c_n is calculated according to Equation 2.8, and the value of p_n according to Equation 2.9.

Equation 2.8

$$c_n = \sum_{k=0}^{L-1} r_{n+k} r_{n+k+D}^{*}$$

Equation 2.9

$$p_n = \sum_{k=0}^{L-1} r_{n+k+D} r_{n+k+D}^{*} = \sum_{k=0}^{L-1} \left| r_{n+k+D} \right|^2$$

Then the decision statistic m_n is calculated from Equation 2.10

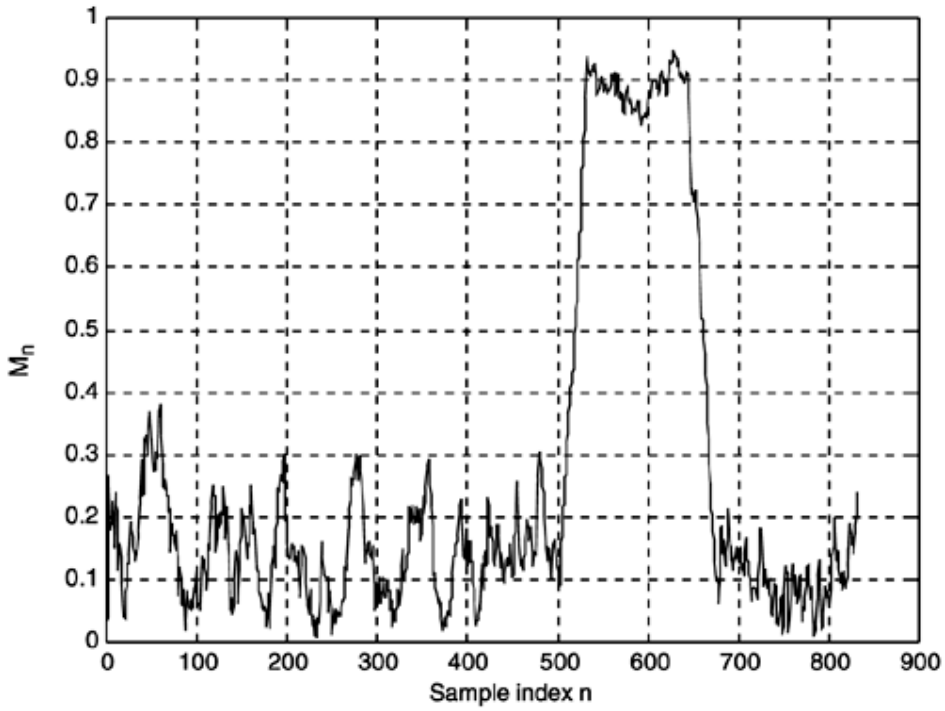
Equation 2.10

$$m_n = \frac{|c_n|^2}{(p_n)^2}$$

c_n and p_n are again sliding windows, so the general recursive procedure can be used to reduce computational workload.

Figure 2.6 shows an example of the decision statistic m_n for IEEE 802.11a preamble in 10dB SNR. The overall response is restricted between [0, 1] and the step at the start of the packet is very clear. The difference of the response to the double sliding window response in Figure 2.3 can be explained with the behavior of the c_n value. When the received signal consists of only noise, the output c_n of the delayed crosscorrelation is zero-mean random variable, since the crosscorrelation of noise samples is zero. This explains the low level of m_n before the start of the packet. Once the start of the packet is received, c_n is a crosscorrelation of the identical short training symbols, which causes m_n to jump quickly to its maximum value. This jump gives a quite good estimate of the start of the packet.

Figure 2.6. Response of the delay and correlate packet detection.



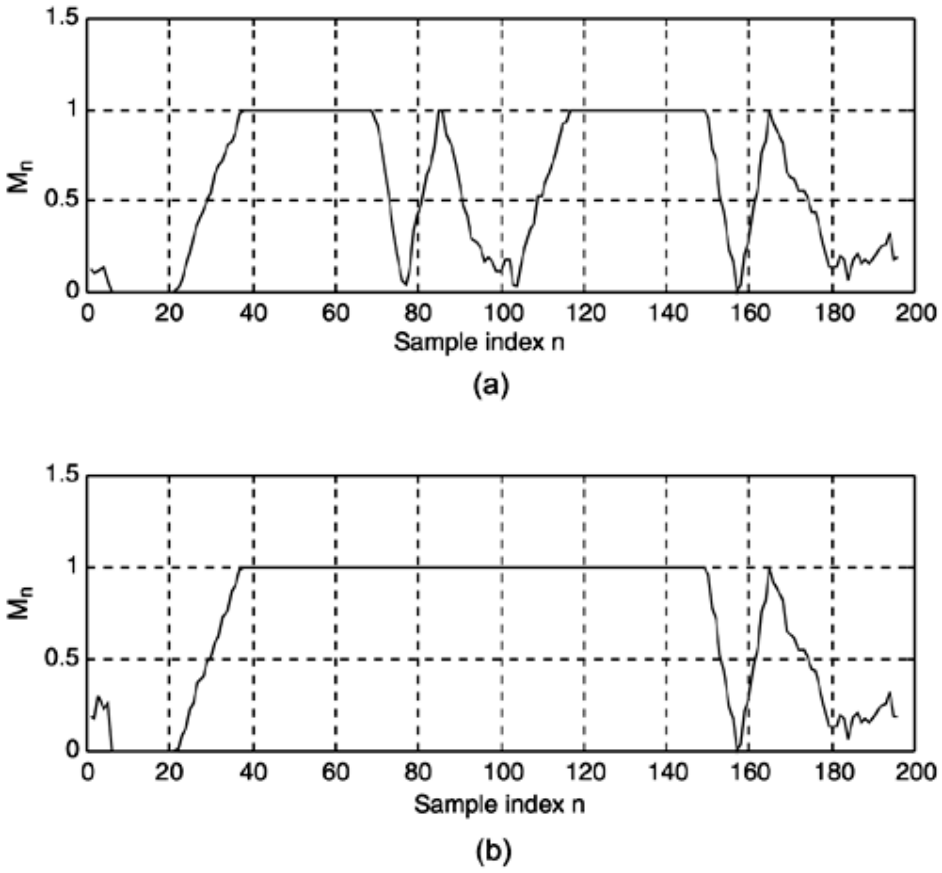
Exercise 1 *The simulation tool implements the delay- and correlate packet detection algorithm. Write a Matlab script for the double sliding window algorithm and compare its performance with the original algorithm.*

HiperLAN/2 Preambles

HiperLAN/2 has been designed with several different preambles, in contrast to the IEEE 802.11a that has only one preamble. The rationale for HiperLAN/2's several preambles is the different MAC architecture of HiperLAN/2. The centralized control approach of HiperLAN/2 enables some optimizations of the preamble design; namely, the preamble length is variable. This reduces overhead for some packets, which in turn increases system capacity. The different preamble structures of HiperLAN/2 are shown in Figure 2.7. The letters inside the figure designate different waveforms that are used during the preamble, the *I* preceding a letter means that the signal is inverted; for example *IA* is equal to *–A*. Preamble (a) is used for broadcast packets from the Access Point (AP); it is a full-length structure, since all the stations have to be able to receive the packet. The preamble (b) is a downlink preamble; it does not contain the short training symbols, because the all the stations know the starting point of the packet and frequency synchronization information. Preamble (c) is intented for generel uplink use. Preamble (d) can be used instead of (c), if the AP supports it. The main reason for the longer (d) preamble is a possibility to use switch antenna diversity, this technique is discussed in more detail in Chapter 4, "Antenna Diversity." The longer short training symbol section allows the receiver to measure the received signal power for two antennas before the long training symbols, and make a decision of which antenna to use.

The different patterns for the short training symbols, when used with the delay and correlate algorithm, allow the receiver to distinguish between the different types of bursts. The response of m_n to preambles (c) and (d) is the same as for the IEEE 802.11a preamble, and is shown in Figure 2.8 (b). The response to preamble (a), however, is a zig-zag curve as the windows slide through multiple sections A, IA, B, and BI, this is shown in Figure 2.8 (a). This zig-zag response designates the broadcast packets that every station should decode. However, if a station receives a flat response, and it is not expecting a packet, decoding is not necessary.

Figure 2.8. Response of delay and correlate to HiperLAN/2 preambles (a) and (d).



Symbol Timing

Symbol timing refers to the task of finding the precise moment of when individual OFDM symbols start and end. The symbol timing result defines the DFT window; i.e., the set of samples used to calculate DFT of each received OFDM symbol. The DFT result is then used to demodulate the subcarriers of the symbol. The approach to symbol timing is different for WLAN and broadcast OFDM systems. A WLAN receiver cannot afford to spend any time beyond the preamble to find the symbol timing, whereas a broadcast transmission receiver can spend several symbols to acquire an accurate symbol timing estimate.

Symbol Timing for a WLAN Receiver

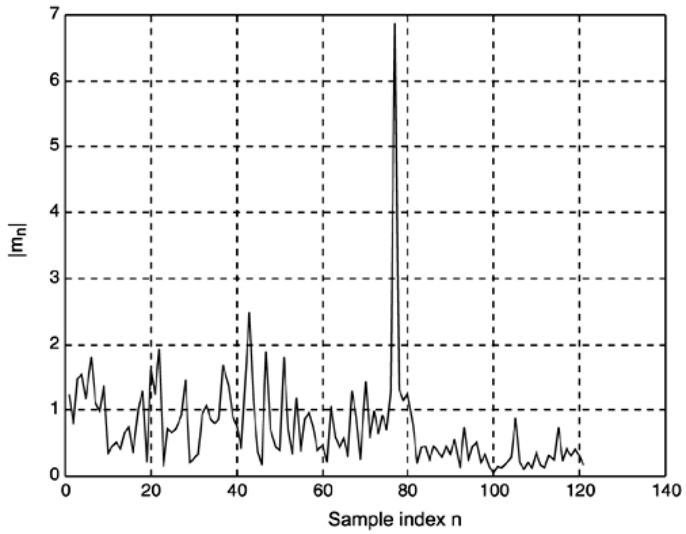
WLAN receivers have knowledge of the preamble available to them, which enables the receiver to use simple crosscorrelation based symbol timing algorithm. After the packet detector has provided an estimate of the start edge of the packet, the symbol timing algorithm refines the estimate to sample level precision. The refinement is performed by calculating the crosscorrelation of the received signal r_n and a known reference t_k; for example, the end of the short training symbols or the start of the long training symbols, to find the symbol timing estimate. Equation 2.11 shows how to calculate the crosscorrelation. The value of *n* that corresponds to maximum absolute value of the crosscorrelation is the symbol timing estimate.

Equation 2.11

$$\hat{t}_s = \arg\max_n \left| \sum_{k=0}^{L-1} r_{n+k} t_k^* \right|^2$$

In Equation 2.11, the length *L* of the crosscorrelation determines the performance of the algorithm. Larger values improve performance, but also increase the amount of computation required. In hardware implementations, it is possible to use only the sign of the reference and received signals, effectively quantizing them to one-bit accuracy. This greatly simplifies the hardware implementation, since no actual multiplications are necessary. Figure 2.9 shows the output of the crosscorrelator that uses the first 64 samples of the long training symbols of the IEEE 802.11a standard as the reference signal. The simulation was run in Additive White Gaussian Noise (AWGN) channel with 10dB SNR. The high peak at $n = 77$ clearly shows the correct symbol timing point.

Figure 2.9. Response of the symbol timing crosscorrelator.

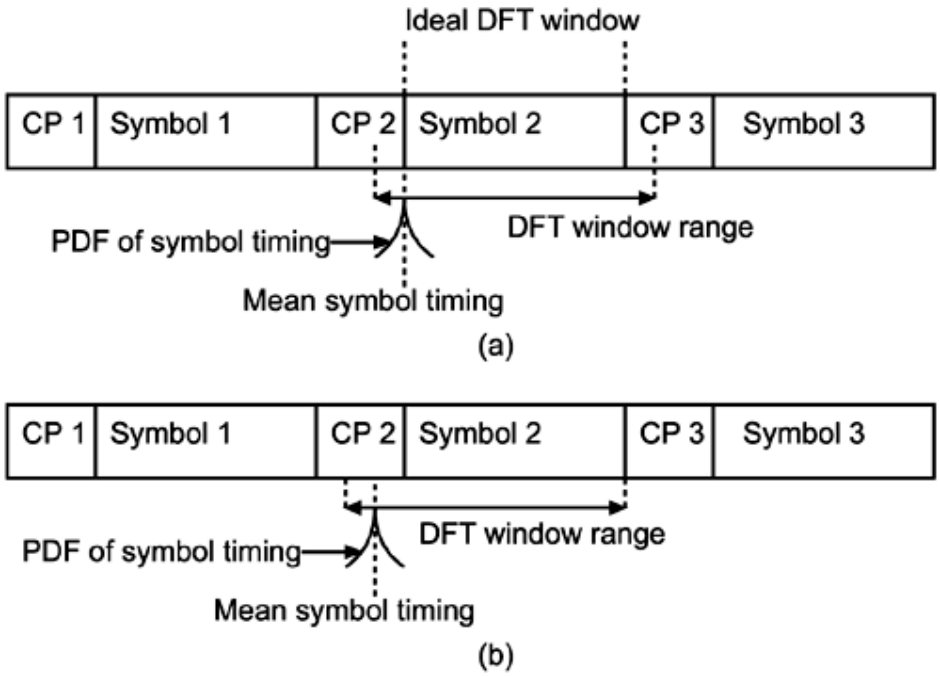


Optimization of Symbol Timing in a Multipath Channel

The performance of the symbol timing algorithm directly influences the effective multipath tolerance of a OFDM system. Characteristics of multipath channels were introduced in Chapter 1, "Background and WLAN Overview." An OFDM receiver achieves maximum multipath tolerance when symbol timing is fixed to the first sample of an OFDM symbol. Figure 2.10 shows three consecutive OFDM symbols, their respective cyclic prefixes (CP), and the effect of symbol timing on the DFT window. The ideal timing for symbol 2 is shown as the ideal DFT window in Figure 2.10 (a). In this case the DFT window does not contain any samples from the CP, hence the maximum length of the channel impulse response (CIR) that does not cause intersymbol interference (ISI) is equal to the CP length. This is the maximum possible ISI free multipath length for an OFDM system. In practice, it is impossible to fix the symbol timing point perfectly to the first sample of the OFDM symbol. There will always be some variability in the symbol timing estimate around its mean value. In addition to the ideal symbol timing, Figure 2.10 (a) shows the probability density function (PDF) of a realistic symbol timing estimate with the mean value equal to the ideal symbol timing. The PDF shows how the actual symbol timing estimate can be before or after the ideal value. The effect on the DFT window is shown as the DFT window range; that is, the range within which the DFT window will fall. When the symbol timing point is estimated before the ideal value, the start of the DFT window will contain samples from the CP and the last samples of the symbol are not used at all. This case does not cause serious problems, because the CP is equal to the last samples of the symbol the circular convolution property of DFT is still satisfied as shown in Equation 1.74. It is possible that some ISI is caused by the early symbol timing estimate, if the CIR is long enough to reach the first samples of the DFT window. For a properly designed OFDM system, the amount of this interference is negligible, because the last taps of the CIR are small. Next consider the case when the symbol timing estimate is after the ideal value. In this case, the start of the DFT window will be after the first sample of the symbol and the last samples are taken from the beginning of the CP of the next symbol. For example, in Figure 2.10 (a), this would mean the the DFT window for symbol 2 would contain samples from the CP 3. When this happens, significant ISI is created by the samples form CP of the next symbol. Additionally the circular convolution property required for the orthogonality of the subcarriers is no longer true, hence intercarrier interference is generated. The end result of a late symbol timing estimate is a significant performance loss. Fortunately, there is a simple solution for this problem. Since early symbol timing does not create significant problems, the mean value of the symbol timing point can be shifted inside the CP. This is shown in Figure 2.10 (b). After the shift, the PDF of the symbol timing estimate is entirely contained inside the CP 2, hence the DFT range is also within CP 2 and symbol 2. This means that the circular convolution is preserved and no ISI is caused by the samples from CP 3. The optimal amount of the shift depends on the OFDM system paremeters and the performance of the symbol timing algorithm. A rule of thumb for an IEEE 802.11a system is 4-6 samples. The drawback of the shift of the mean symbol timing

point is reduced multipath tolerance of the OFDM system. The shift effectively shortens the CP by the amount of the shift, because some samples of the CP are always used for the DFT window. This means that a CIR shorter that the CP can cause ISI; however, as was mentioned earlier, the last taps of the CIR are quite small, so the effect on performance is not serious. Nevertheless the OFDM system designer should keep the symbol timing in mind and put some samples in to the CP in addition to the maximum expected CIR length.

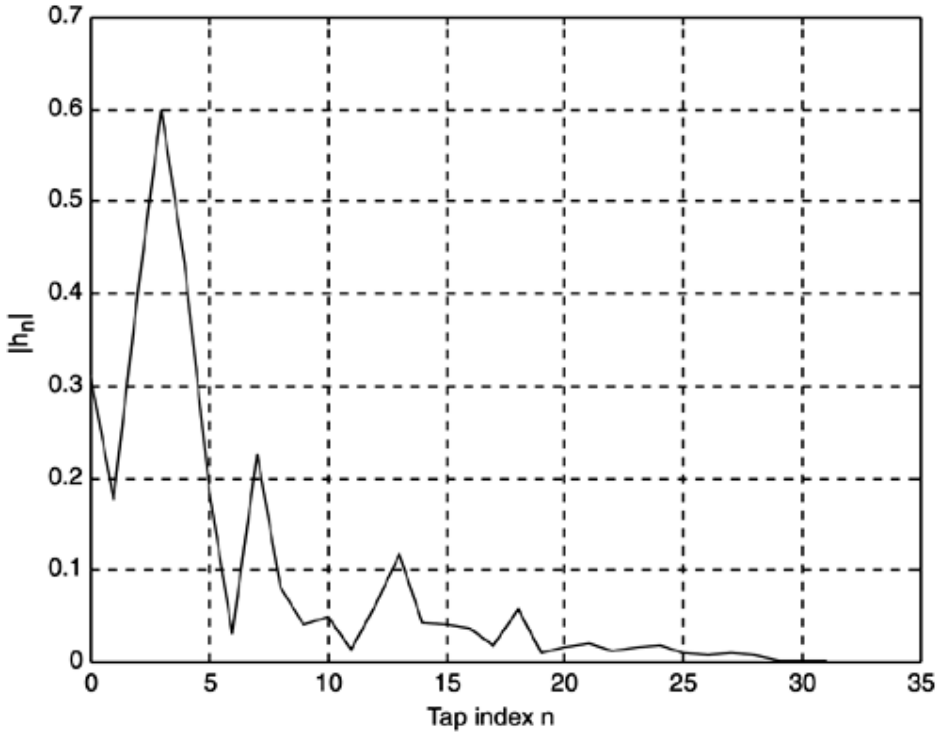
Figure 2.10. Effect of variance in symbol timing.



Exercise 2 *The simulation tool allows to change the amount of the symbol timing shift. Experiment with different values with a multipath channel model, and observe the performance difference.*

In a multipath environment, the symbol timing can be further optimized, if an estimate of the multipath taps h_n of the channel impulse response is available to the receiver. This approach is discussed in more detail in [15]. Figure 2.11 shows a sample of a channel impulse response. In the sample, the strongest tap occurs several taps later than the first one. The crosscorrelation symbol timing estimator would pick the strongest tap 3 as the timing point; however, this will mean that the taps 0 to 2 will not contribute to the signal energy inside the DFT window. This means that some signal energy is lost due to the symbol timing estimate. This loss can be recovered by changing the timing point such that the energy of h_n inside the DFT window is maximized. For the CIR sample in Figure 2.11, this would mean shifting the timing point to the left to cover the taps that occur before the strongest tap.

Figure 2.11. A sample of a channel impulse response.



Continuous Transmission System Symbol Timing

The different symbol timing algorithms for continuous transmission receivers can coarsely be divided into data-aided and nondata-aided methods. To facilitate data-aided methods broadcast OFDM systems like DVB, periodically insert known training symbols into the transmitted signal. These symbols are called *pilot symbols*. The nondata-aided methods use the cyclic prefix structure of OFDM waveform to perform symbol timing. In this sense, they are similar to the preamble structure packet detection algorithm; the algorithm presented in Schimdl and Cox [21] was developed for symbol timing, but the approach could also be used packet detection. Several authors have investigated symbol timing based on the cyclic prefix [1, 2]. Okada et al. [16] develop a technique for jointly estimating the symbol timing and frequency offset of the received signal. The underlying theme in all these techniques is the use of correlation to detect the repetition of the cyclic prefix in the received signal.

Sample Clock Tracking

A quite different timing estimation problem is tracking the sampling clock frequency. The oscillators used to generate the Digital to Analog Converter (DAC) and Analog to Digital Converter (ADC) sampling instants at the transmitter and receiver will never have exactly the same period. Thus the sampling instants slowly shift relative to each other. The problem has been analyzed by several authors [6, 11, 17, 19].

The sampling clock error has two main effects: A slow shift of the symbol timing point, which rotates subcarriers, and a loss of SNR due to the intercarrier interference (ICI) generated by the slightly incorrect sampling instants, which causes loss of the orthogonality of the subcarriers. In [23] the normalized sampling error is defined as

Equation 2.12

$$t_\Delta = \frac{T' - T}{T}$$

where *T* and T' are the transmitter and receiver sampling periods, respectively. Then the overall effect, after DFT, on the received subcarriers $R_{l,k}$ is shown as

Equation 2.13

$$R_{l,k} = e^{j2\pi k t_\Delta l \frac{T_s}{T_u}} X_{l,k} \operatorname{sinc}(\pi k t_\Delta) H_{l,k} + W_{l,k} + N_{t_\Delta}(l,k)$$

where *l* is the OFDM symbol index, *k* is the subcarrier index, T_s and T_u are the duration of the total OFDM symbol and the useful data portion, $W_{l,k}$ is additive white noise and the last term $N_{t_\Delta}(l,k)$ is the additional interference due to the sampling frequency offset. The power of the last term is approximated by

Equation 2.14

$$P_{t_\Delta} \approx \frac{\pi^2}{3}(k t_\Delta)^2$$

hence the degradation grows as the square of the product of the offset t_Δ and the subcarrier index *k*. This means that the outermost subcarriers are most severely affected. The degradation can also be expressed directly as SNR loss in decibels. The following approximation [19] is derived

Equation 2.15

$$D_n \approx 10 \log_{10}\left(1 + \frac{\pi^2}{3}\frac{E_s}{N_0}(k t_\Delta)^2\right)$$

WLAN OFDM systems typically have relatively small number of subcarriers and quite small t_Δ, hence $kt_\Delta \ll 1$, so the interference caused by sampling frequency offset can usually be ignored. Equation 2.13 also shows the more significant problem caused by the offset, namely the term

Equation 2.16

$$e^{j2\pi k t_\Delta l \frac{T_s}{T_u}}$$

This term shows the amount of rotation angle experienced by the different subcarriers as an OFDM signal is received. The angle depends on both the subcarrier index *k* and the OFDM symbol index *l.* Hence the angle is largest for the outermost subcarriers and increases with consecutive OFDM symbols. The term t_Δ is usually quite small, but as *l* increases, the rotation will eventually be so large that correct demodulation is no longer possible. This necessitates tracking the sampling frequency offset.

Estimating the Sampling Frequency Error

The majority of the published approaches to estimating the sampling frequency offset rely on *pilot subcarriers.* The pilot subcarriers are used to transmit known data, called *pilot symbols,* that the receiver can use to perform synchronization functions. Sampling frequency error estimation algorithms usually assume the pilots to be symmetrically distributed around a middle subcarrier. The following method was presented in Speth et al. [25] and Fechtel [6]. The pilot subcarriers are divided into two sets; C_1 corresponds to pilots on negative subcarriers, and C_2 to pilots on positive subcarriers. The sampling frequency offset is estimated by using the knowledge of the linear relationship between the phase rotation caused by the offset and the pilot subcarrier index. The received pilot subcarriers, in a simplified form, are

Equation 2.17

$$R_{l,k} = H_k P_{l,k} e^{j2\pi k t_\Delta l \frac{T_s}{T_u}}$$

Then calculate the rotation of the pilots from one symbol to the next

Equation 2.18

$$Z_{l,k} = R_{l,k} R^*_{l-1,k}$$

Equation 2.19

$$= H_k P_{l,k} e^{j2\pi k t_\Delta l \frac{T_s}{T_u}} \left(H_k P_{l-1,k} e^{j2\pi k t_\Delta l \frac{T_s}{T_u}} \right)^*$$

Equation 2.20

$$= |H_k|^2 |P_{l,k}|^2 e^{j2\pi k t_\Delta l \frac{T_s}{T_u}} e^{-j2\pi k t_\Delta (l-1) \frac{T_s}{T_u}}$$

Equation 2.21

$$= |H_k|^2 |P_{l,k}|^2 e^{j2\pi k t_\Delta l \frac{T_s}{T_u}}$$

Next calculate the cumulative phases of $Z_{l,k}$ for the two sets C_1 and C_2 as

Equation 2.22

$$\phi_{1,l} = \angle \left[\sum_{k \in C_1} Z_{l,k} \right]$$

Equation 2.23

$$\phi_{2,l} = \angle \left[\sum_{k \in C2} Z_{l,k} \right]$$

The sampling frequency offset is then estimated by

Equation 2.24

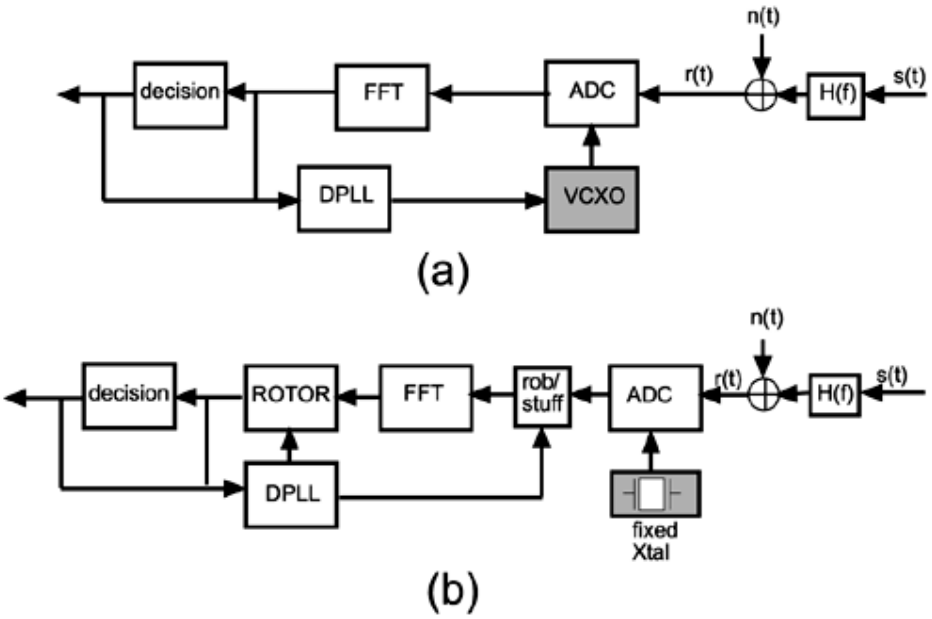
$$\hat{t}_\Delta = \frac{1}{2\pi} \frac{T_u}{T_s} \frac{1}{\min_{k \in C_2}(k) + \max_{k \in C_2}(k)} \left(\phi_{2,l} - \phi_{1,l} \right)$$

where the normalization factor $\min_{k \in C_2}(k) + \max_{k \in C_2}(k)$ assumes that the pilot indexes *k* are evenly distributed.

Correcting the Sampling Frequency Error

The rotation caused by the sampling frequency offset can be corrected with two main approaches. Namely, the problem can be corrected at its source by adjusting the sampling frequency of the receiver DAC—the receiver structure is shown on top of Figure 2.12. Secondly, the rotation can be corrected after the DFT processing by derotating the subcarriers, the lower receiver structure in Figure 2.12. Both of these solutions are analyzed in Pollet et al. [19], where the first method is referred to as synchronized sampling and the second as non-synchronized sampling.

Figure 2.12. Adapted from Pollet et. al. [19]: Receiver structures for correcting sampling frequency error. © 1994 IEEE.



The adjustment of the clock of the ADC naturally perfectly removes the sampling frequency offset, provided the estimate of the offset is accurate. In this sense, it is the optimal way of performing the correction. However, during recent years, the trend in receiver design has been towards all digital receivers, that do not attempt to adjust the sampling clock. This is illustrated in the lower receiver structure in Figure 2.12 by the fixed crystal that controls the ADC. The rationale for this trend is the desire to simplify the analog part of the receiver, as the analog components are relatively costly compared to digital gates. Hence by using a fixed crystal, instead of a controllable one, the number of analog components can be reduced, and as a result the cost of the receiver can be decreased. The non-synchronized sampling receiver in Figure 2.12 (b) shows an additional block named "rob/stuff," just after the ADC. This block is required, because the drift in sampling instant will eventually be larger than the sampling period. When this happens, the "rob/stuff" block will either "stuff" a duplicate sample or "rob" one sample from the signal, depending on whether the receiver clock is faster or slower than the transmitter clock. This process prevents the receiver sampling instant from drifting so much that the symbol timing would be incorrect. The "ROTOR" block performs the required phase corrections with the information provided by the Digital Phase-Locked Loop (DPLL) that estimates the sampling frequency error.

Frequency Synchronization

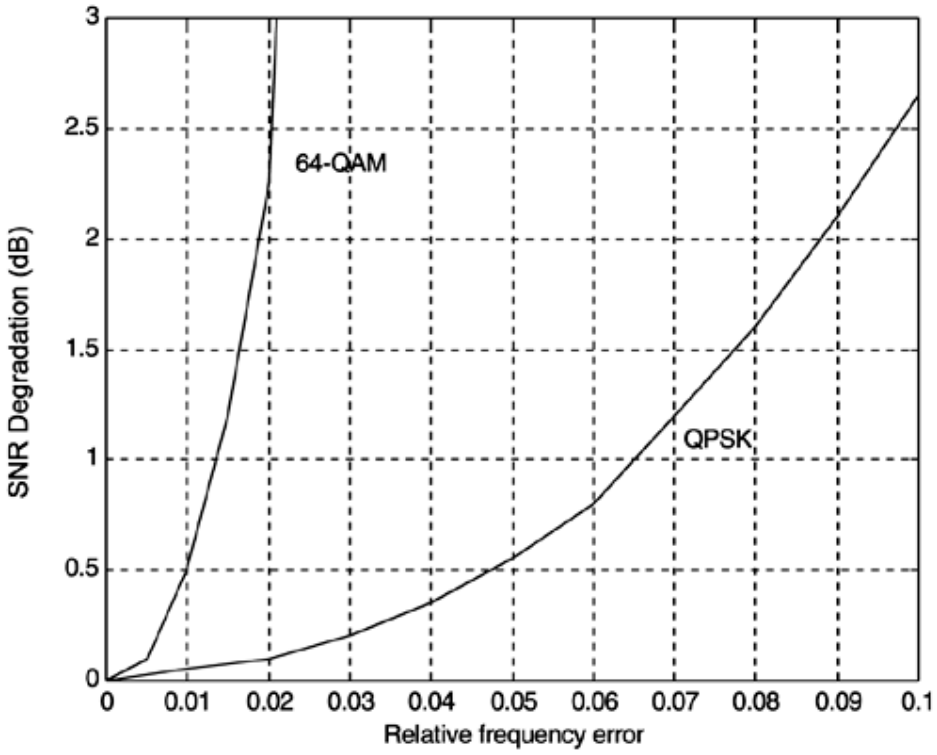
One of the main drawbacks of OFDM is its sensitivity to carrier frequency offset. The effect of carrier frequency error on SNR was introduced in Chapter 1, where the main differences between multicarrier and singlecarrier systems were discussed. The degradation is caused by two main phenomena: reduction of amplitude of the desired subcarrier and ICI caused by neighboring carriers. The amplitude loss occurs because the desired subcarrier is no longer sampled at the peak of the sinc-function of DFT. The sinc-function is defined as $\mathrm{sinc}(x) = \frac{\sin(x)}{x}$. Adjacent carriers cause interference, because they are not sampled at the zero-crossings of they sinc-functions. The overall effect on SNR is analyzed in Pollet et al. [18], and for relatively small frequency errors, the degradation in dB was approximated by

Equation 2.25

$$SNR_{Loss} = \frac{10}{3\ln 10}(\pi T f_{\Delta})^2 \frac{E_s}{N_0} \text{dB}$$

where f_{Δ} is the frequency error as a fraction of the subcarrier spacing and T is the sampling period. The performance effect varies strongly with the modulation used; naturally, constellations with fewer points can tolerate larger frequency errors than large constellations. Figure 2.13 illustrates the effect for QPSK and 64-QAM constellations. The figure shows that the 64QAM, the largest constellation in IEEE802.11a, cannot tolerate more than 1% error in the carrier frequency for a negligible SNR loss of 0.5dB, whereas QPSK modulation can tolerate up to 5% error for the same SNR loss.

Figure 2.13. Symbol error rate (SER) degradation due to frequency offset at *SER* = 10^{-4}. © 1995 IEEE. Adapted from Daffara and Adami [5].



The analysis above would seem to indicate that large constellations are exceedingly difficult to use with an OFDM system. However, keep in mind that a large constellation automatically implies higher operating SNR than with a small constellation. This directly improves the performance of the frequency error estimators, as is shown in Equations 2.41 and 2.57.

In Hsieh and Wei [8], the various algorithms that have been developed to estimate carrier frequency offsets in OFDM systems are divided into three types:

Type 1	Data-aided algorithms; these methods are based on special training information embedded into the transmitted signal.
Type 2	Nondata-aided algorithms that analyze the received signal in frequency domain.
Type 3	Cyclic prefix based algorithms that use the inherit structure of the OFDM signal provided by the cyclic prefix.

For WLAN applications, type 1 is the most important. The preamble allows the receiver to use efficient maximum likelihood algorithms to estimate and correct for the frequency offset, before the actual information portion of the packet starts. The algorithms belonging to types 2 and 3 are better suited for broadcast or continuous transmission type OFDM systems. Owing to the nature of this book, we will focus on data-aided frequency synchronization algorithms in this section.

Time Domain Approach for Frequency Synchronization

We first derive a data-aided maximum-likelihood estimator that operates on the received time domain signal. This method has been presented in several papers in slightly varying forms [2, 21]. The training information required is at least two consecutive repeated symbols. As a note, the preamble of the WLAN standards satisfies this requirement for both the short and long training symbols.

Let the transmitted signal be s_n, then the complex baseband model of the passband signal y_n is

Equation 2.26

$$y_n = s_n e^{j2\pi f_{tx} n T_s}$$

where f_{tx} is the transmitter carrier frequency. After the receiver downconverts the signal with a carrier frequency f_{rx}, the received complex baseband signal r_n, ignoring noise for the moment, is

Equation 2.27

$$r_n = s_n e^{j2\pi f_{tx} n T_s} e^{-j2\pi f_{rx} n T_s}$$

Equation 2.28

$$= s_n e^{j2\pi (f_{tx} - f_{rx}) n T_s}$$

Equation 2.29

$$= s_n e^{j2\pi f_\Delta n T_s}$$

where $f_\Delta = f_{tx} - f_{rx}$ is the difference between the transmitter and receiver carrier frequencies. Let *D* be the delay between the identical samples of the two repeated symbols. Then the frequency offset estimator is developed as follows, starting with an intermediate variable *z*

Equation 2.30

$$z = \sum_{n=0}^{L-1} r_n r_{n+D}^*$$

Equation 2.31

$$= \sum_{n=0}^{L-1} s_n e^{j2\pi f_\Delta n T_s} \left(s_{n+D} e^{j2\pi f_\Delta (n+D) T_s} \right)^*$$

Equation 2.32

$$= \sum_{n=0}^{L-1} s_n s_{n+D}^* e^{j2\pi f_\Delta n T_s} e^{-j2\pi f_\Delta (n+D) T_s}$$

Equation 2.33

$$= e^{-j2\pi f_\Delta D T_s} \sum_{n=0}^{L-1} |s_n|^2$$

Equation 2.33 is a sum of complex variables with an angle proportional to the frequency offset. Finally, the frequency error estimator is formed as

Equation 2.34

$$\hat{f}_{\Delta} = -\frac{1}{2\pi DT_s}\angle z$$

where the $\angle$z operator takes the angle of its argument.

Properties of the Time Domain Frequency Synchronization Algorithm

An important feature of the present method is its operating range. The operating range defines how large frequency offset can be estimated. The range is directly related to the length of the repeated symbols. The angle of z is of the form $-2\pi f_{\Delta} DT_s$, which is unambiguously defined only in the range $[-\pi, \pi)$. Thus if the absolute value of the frequency error is larger than the following limit

Equation 2.35

$$|f_{\Delta}| \geq \frac{\pi}{2\pi DT_s} = \frac{1}{2DT_s}$$

the estimate will be incorrect, since z has rotated an angle larger than ¼. This maximum allowable frequency error is usually normalized with the subcarrier spacing f_s. If the delay D is equal to the symbol length, then

Equation 2.36

$$\frac{1}{2DT_s} = \frac{1}{2} f_s$$

Thus the frequency error can be at most a half of the subcarrier spacing. It should be noted that if the repeated symbols include a cyclic prefix, the delay is longer than the symbol length, and hence the range of the estimator is reduced.

As an example we can calculate the value of this limit for the IEEE 802.11a system for both the short and long training symbols. For the short training symbols, the sample time is 50ns, and the delay D = 16. Thus the maximum frequency error that can be estimated is

Equation 2.37

$$f_{\Delta \max} = \frac{1}{2DT_s}$$

Equation 2.38

$$= \frac{1}{2 \cdot 16 \cdot 50 \cdot 10^{-9}}$$
$$= 625kHz$$

This should be compared with the maximum possible frequency error in an IEEE802.11a system. The carrier frequency is approximately 5.3 GHz, and the standard specifies a maximum oscillator error of 20 parts per million (ppm). Thus if the transmitter and receiver clocks have the maximum allowed error, but with opposite signs, the total observed error will be 40ppm. This amounts to a frequency error of $f_\Delta = 40 \cdot 10^{-6} \cdot 5.3 \cdot 10^9 = 212kHz$. Hence the maximum possible frequency error is well within the range of the algorithm. Now consider the long training symbols. The only significant difference is that the delay $D = 64$ is four times longer. Hence the range is

Equation 2.39

$$f_{\Delta \max} = \frac{1}{4 \cdot 2DT_s}$$

Equation 2.40

$$= 156.25kHz$$

Observe that this is less than the maximum possible error defined in the standard. Thus this estimator would not be reliable if only the long training symbols were used.

Beek et al. [2] show that in an AWGN channel the estimator $\hat{f}_\Delta$ is a maximum-likelihood estimate of the frequency offset. Additionally, under the same AWGN assumption, Schimdl and Cox [21] do an analysis of the performance of the algorithm and show that at high SNR the variance $\sigma^2_{\hat{f}_\Delta}$ of the estimator is proportional to

Equation 2.41

$$\sigma^2_{\hat{f}_\Delta} \sim \frac{1}{L \cdot SNR}$$

Hence the more samples in the sum, the better the quality of the estimator will be, as one would expect.

Post DFT Approach to Frequency Error Estimation

Frequency offset estimation can also be performed after the DFT processing [14]. As with the time domain algorithm, at least two consecutive repeated symbols are required. The preamble structure of WLAN standards has this property during the short and long training symbols, thus allowing the receiver to perform the frequency error estimation also after the DFT processing. Following the analysis in Moose [14], a maximum-likelihood frequency error estimator is derived as follows. The received signal during two repeated symbols is, ignoring noise for convenience,

Equation 2.42

$$r_n = \frac{1}{N}\left[\sum_{k=-K}^{K} X_k H_k e^{\frac{j2\pi n(k+f_\Delta)}{N}}\right] \qquad n = 0,1,\ldots,2N-1$$

where X_k are the transmitted data symbols,H_k is the channel frequency response for subcarrier *k, K* is the total number of subcarriers, and f_Δ is the fractional frequency error normalized to the subcarrier spacing. Then calculating the DFT for the first received symbol, the value of *k*th subcarrier is

Equation 2.43

$$R_{1,k} = \sum_{n=0}^{N-1} r_n e^{\frac{-j2\pi kn}{N}} \qquad k = 0,1,\ldots,N-1$$

and the DFT of the second symbol is

Equation 2.44

$$R_{2,k} = \sum_{n=N}^{2N-1} r_n e^{\frac{-j2\pi kn}{N}}$$

Equation 2.45

$$= \sum_{n=0}^{N-1} r_{n+N} e^{\frac{-j2\pi kn}{N}} \qquad k = 0,1,\ldots,N-1$$

Now the crucial observation, from Equation 2.42, is that

Equation 2.46

$$r_{n+N} = \frac{1}{N}\left[\sum_{k=-K}^{K} X_k H_k e^{\frac{j2\pi(n+N)(k+f_\Delta)}{N}}\right]$$

Equation 2.47

$$= \frac{1}{N}\left[\sum_{k=-K}^{K} X_k H_k e^{\frac{j2\pi n(k+f_\Delta)}{N}} e^{\frac{j2\pi(k+f_\Delta)N}{N}}\right]$$

Equation 2.48

$$= \frac{1}{N}\left[\sum_{k=-K}^{K} X_k H_k e^{\frac{j2\pi n(k+f_\Delta)}{N}} e^{j2\pi(k+f_\Delta)}\right]$$

Equation 2.49

$$= \frac{1}{N}\left[\sum_{k=-K}^{K} X_k H_k e^{\frac{j2\pi n(k+f_\Delta)}{N}}\right] e^{j2\pi f_\Delta}$$

Equation 2.50

$$= r_n e^{j2\pi f_\Delta}$$

since $e^{j2\pi k} = 1$. Equation 2.50 implies that

Equation 2.51

$$R_{2,k} = R_{1,k} e^{j2\pi f_\Delta}$$

Thus every subcarrier experiences the same phase shift that is proportional to the frequency offset. Hence the frequency error can be estimated from this phase shift. Using an intermediate variable *z* again

Equation 2.52

$$z = \sum_{k=-K}^{K} R_{1,k} R_{2,k}^{*}$$

Equation 2.53

$$= \sum_{k=-K}^{K} R_{1,k} \left(R_{1k} e^{j2\pi f_{\Delta}} \right)^{*}$$

Equation 2.54

$$= e^{-j2\pi f_{\Delta}} \sum_{k=-K}^{K} R_{1,k} R_{1,k}^{*}$$

Equation 2.55

$$= e^{-j2\pi f_{\Delta}} \sum_{k=-K}^{K} \left| R_{1,k} \right|^{2}$$

Thus *z* is a complex variable, for which the angle is defined by the frequency error. Finally the estimator is

Equation 2.56

$$\hat{f}_{\Delta} = -\frac{1}{2\pi} \angle z$$

which is quite similar in form to the time domain version.

Properties of the Post DFT Frequency Error Estimation Algorithm

The estimator uses the angle of the complex variable as the basis for the estimate, thus the same $[-\pi, \pi)$ unique range applies as for the time domain estimator. The frequency error f_Δ was defined as a fractional value normalized to the subcarrier spacing, hence the maximum estimable frequency error is again a half of the subcarrier spacing. Otherwise the angle of z will rotate over the $\pm\pi$ limit. This implies that the frequency range is the same for both the time domain and frequency domain estimators.

A somewhat counterintuitive property of this estimator is that the ICI introduced by calculating the DFT of the signal with frequency offset is actually useful information for the estimator. Thus there is no performance penalty compared to the time domain algorithm. In fact, the variance $\sigma^2_{\hat{f}_\Delta}$ is shown [14] to be proportional to

Equation 2.57

$$\sigma^2_{\hat{f}_\Delta} \sim \frac{1}{L \cdot SNR}$$

which is the same result as for the time domain algorithm.

Exercise 3 *The simulation tool implements the time domain frequency error estimation algorithm. Write a Matlab script for the frequency domain algorithm and compare the performance of the algorithms.*

Comments on Frequency Error Estimation Algorithms

Figure 2.4 suggests a two-step frequency estimation process with a coarse frequency estimate performed from the short training symbols and a fine frequency synchronization from the long training symbols. The rationale for this can be explained with the estimation range of the time domain algorithm and the post DFT algorithm. As was shown in Equation 2.38, either estimator could not reliably estimate the frequency error only from the long training symbols. The two-step process would then proceed by first acquiring a coarse estimate of the frequency error from the short training symbols, and then correcting the long training symbols with this estimate. The accuracy of the coarse estimate should easily be better than the 156.25kHz range of the estimator during the long training symbols. Hence a second estimation step could be done from the long training symbols to improve the estimate. Whether the second step is necessary depends on the accuracy of the first estimate. If enough data samples are used to calculate the first estimate from the short symbols, a satisfactory accuracy can usually be reached. Hence the second estimation would be unnecessary.

The main disadvantage of frequency domain estimation is that the DFT has to be calculated for both repeated symbols. Compared to the time domain estimator, the DFT operations mean additional computations without any advantages. Thus the time domain method is preferable for a WLAN receiver, which in general has very little time to complete all the necessary synchronization functions during the preamble.

Alternative Techniques for Frequency Error Estimation

Hsieh and Wei [8] present a technique to remedy the ±0.5 subcarrier estimation range. The idea is to take advantage of the correlation of the channel frequency response between adjacent subcarriers. The algorithm first uses training data to estimate the channel frequency response. The next step is to calculate

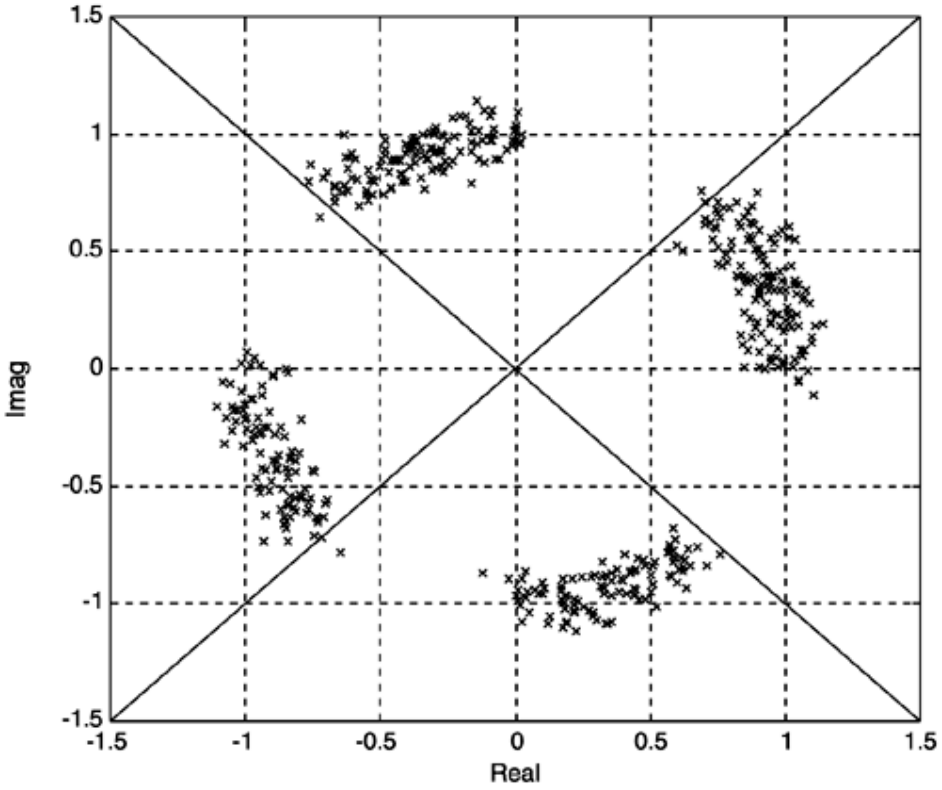
the autocorrelation function of the channel estimate. Since the frequency offset shifts the received signal in frequency, the autocorrelation function will have its maximum at the lag corresponding to the frequency offset. The authors show that in an OFDM system with 1,024 subcarriers frequency offsets up to ±200 subcarrier spacings can be estimated, which is a significant improvement from the ±0.5 subcarrier spacing limit.

Traditional single carrier receivers usually employ some kind of phase locked loop to track the carrier phase. The main difficulty in using phase locked loops for OFDM carrier phase tracking is the nature of the OFDM waveform. It is not easy to design a phase error detector, a main component of a phase locked loop, for an OFDM system. A carrier frequency tracking technique for OFDM that resembles single carrier frequency recovery loop has been proposed [13]. The proposed method is composed of two steps: acquisition and tracking. The acquisition step is performed from known synchronization symbols. Tracking mode is a frequency control loop that has been modified for OFDM waveform. The main advantage of this method is its quite large acquisition range, compared to the two estimators presented above. This advantage is significant for broadcast systems like DVB and DAB, which place subcarriers very close to each other; for example, the minimum subcarrier spacing for DAB is 1 kHz. Thus a half subcarrier spacing acquisition range of the estimator practically renders the presented time domain and post DFT methods useless. In the paper, the crucial phase error detection is performed by detecting and remodulating the received data, and then comparing the remodulated signal to the received signal. A similar technique is also investigated in Daffara and Chouly [4]. The structure of the phase error detector of the previous two papers is simplified [5] by exploiting the cyclic prefix to detect the phase error.

Carrier Phase Tracking

Frequency estimation is not a perfect process, so there is always some residual frequency error. The SNR loss due to the ICI generated should not be a problem if the estimator has been designed to reduce the frequency error below the limit required for a negligible performance loss for the used modulation. The main problem of the residual frequency offset is constellation rotation. The analysis of the post DFT frequency error estimator also shows that the constellation rotation is the same for all subcarriers. To illustrate the effect in an IEEE 802.11a system, Figure 2.14 shows how much a QPSK constellation rotates during 10 OFDM symbols with a 3kHz frequency error. This error corresponds to only 1% of the subcarrier spacing, thus the effect on SNR is negligible as shown earlier. The figure shows that after only 10 symbols, the constellation points have just rotated over the decision boundaries shown as solid lines, thus correct demodulation is no longer possible.

Figure 2.14. Constellation rotation with 3kHz frequency error during 10 symbols.



This effect forces the receiver to track the carrier phase while data symbols are received.

Data-Aided Carrier Phase Tracking

The simplest method is data-aided tracking of the carrier phase. IEEE 802.11a and HiperLAN/2 include four predefined subcarriers among the transmitted data. These special subcarriers are referred to as *pilot* subcarriers. The main purpose of these pilots is exactly to help the receiver to track the carrier phase. After the DFT of the *n*th received symbol, the pilot subcarriers $R_{n,k}$ are equal to the product of the channel frequency response H_k and the known pilot symbol $P_{n,k}$, rotated by the residual frequency error.

Equation 2.58

$$R_{nk} = H_k P_{nk} e^{j2\pi n f_\Delta}$$

Assuming an estimate $\hat{H}_k$ of the channel frequency response is available, the phase estimate is

Equation 2.59

$$\hat{\Phi}_n = \angle \left[\sum_{k=1}^{N_p} R_{n,k} \left(\hat{H}_k P_{n,k} \right)^* \right]$$

Equation 2.60

$$= \angle \left[\sum_{k=1}^{N_p} H_k P_{n,k} e^{j2\pi n f_\Delta} \left(\hat{H}_k P_{n,k} \right)^* \right]$$

If we assume that the channel estimate is perfectly accurate, we get the estimator

Equation 2.61

$$\hat{\Phi}_n = \angle \left[\sum_{k=1}^{N_p} |H_k|^2 |P_{n,k}|^2 e^{j2\pi n f_\Delta} \right]$$

Equation 2.62

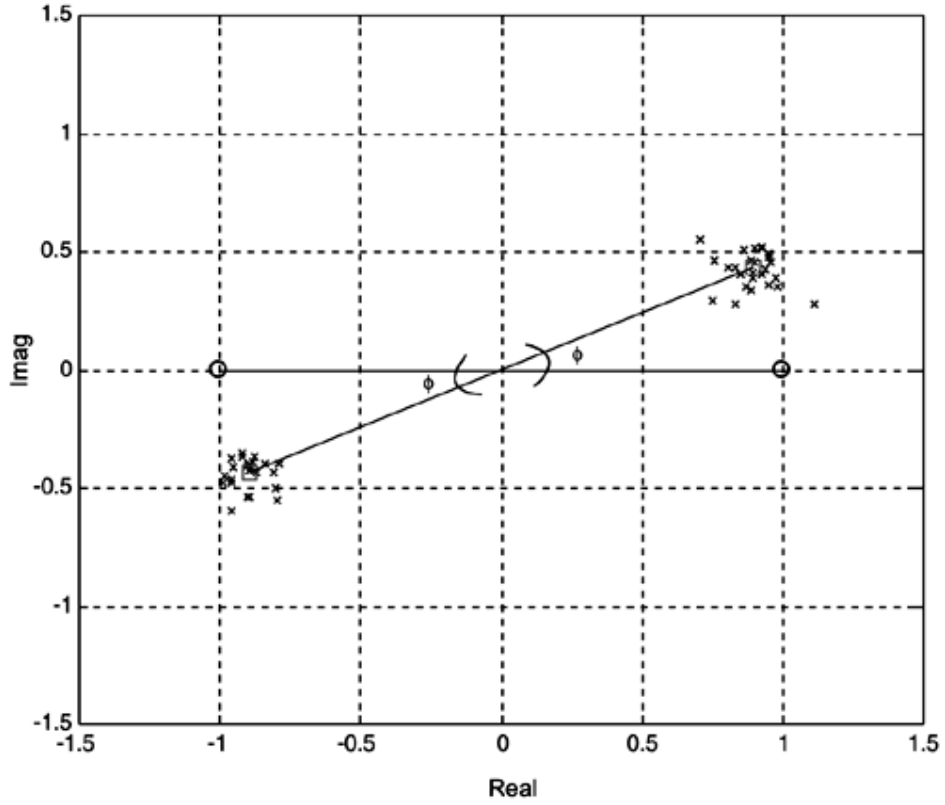
$$= \angle \left[e^{j2\pi n f_\Delta} \sum_{k=1}^{N_p} |H_k|^2 \right]$$

In this case we need not worry about the [–π, π) phase range, because the pilot data are known, thus the phase ambiguity is automatically resolved correctly. Note that in practice the channel estimates are not perfectly accurate, thus they contribute to the noise in the estimate.

Nondata-Aided Carrier Phase Tracking

The phase error can be estimated without the help of the pilot symbols. Recall that the phase error resulting from the frequency offset is identical for all the subcarriers; this property (identical frequency offset) allows the development of a nondata-aided estimator for the phase error. Figure 2.15 shows how the data subcarriers have rotated for a single BPSK modulated OFDM symbol after the channel effect has been corrected. The angle Φ shown in the figure is the result of the frequency offset.

Figure 2.15. BPSK single symbol rotation.



The angle Φ can be estimated without any knowledge of the data by performing hard decisions $\hat{X}_{n,k}$ on the received data symbols $R_{n,k} = H_k X_{nk} e^{j2\pi n f_\Delta}$ after they are corrected for the channel effect. Then the angle between a hard decision and the corresponding received symbol is used as an estimator.

Equation 2.63

$$\hat{\Phi} = \angle\left[\sum_{k=-K}^{K} R_{n,k}\left(\hat{X}_k \hat{H}_k\right)^*\right]$$

Equation 2.64

$$= \angle\left[\sum_{k=-K}^{K} H_k \hat{H}_k^* X_{n,k} \hat{X}_{n,k}^* e^{j2\pi n f_\Delta}\right]$$

Equation 2.65

$$= \angle\left[\sum_{k=-K}^{K}|H_k|^2|X_{n,k}|^2 e^{j2\pi n f_\Delta}\right]$$

In Equation 2.63-2.65, we have assumed that all the hard decisions are correct and that the channel estimates are perfect. In reality, neither condition is true, but if the number of data carriers is large the effect of incorrect hard decisions will not have a significant impact.

It should be noted that the total rotation angle increases from symbol to symbol, hence it is necessary to compensate this angle before the hard decisions are performed. Without the compensation the rotation angle will eventually cross the decision boundaries of the constellation, and then all the hard decisions will be incorrect. As a result, the phase estimate will also be worthless. This sounds like a circular argument: How can the angle be compensated before it has been estimated? A simple tactic is to keep track of the total rotation up to the previous symbol, and correct the data with this value, before the hard decisions are performed. The rotation from symbol to symbol is quite small, thus the probability of incorrect hard decisions is not increased significantly. After the angle has been estimated, the total rotation angle is updated for the next symbol.

Exercise 4 *The simulation tool implements the data-aided carrier phase tracking algorithm. Write a Matlab script for the nondata-aided algorithm and compare the performance of the algorithms. Try to implement a method to combine both algorithms to achieve improved performance.*

Channel Estimation

Channel estimation is the task of estimating the frequency response of the radio channel the transmitted signal travels before reaching the receiver antenna. The impulse response of a time varying radio channel is usually [20] represented as a discrete time FIR filter

Equation 2.66

$$h(\tau;t) = \sum_n \alpha_n(t) e^{-j2\pi f_c \tau_n(t)} \delta(\tau - \tau_n(t))$$

WLAN applications generally assume that the channel is *quasistationary,* that is, the channel does not change during the data packet. With this assumption the time dependency in Equation 2.66 can be dropped

Equation 2.67

$$h(\tau) = \sum_n \alpha_n e^{-j2\pi f_c \tau_n(t)} \delta(\tau - \tau_n)$$

Then the discrete time frequency response of the channel is the Fourier transform of the channel impulse response

Equation 2.68

$$H_k = DFT\{h_n\}$$

Hence the channel estimation process outputs the $\hat{H}_k$, an estimate of the channel frequency response for each subcarrier.

Channel estimation is mandatory for OFDM systems that employ coherent modulation schemes. Otherwise correct demodulation would not be possible. Knowledge of the channel can also improve the performance OFDM with noncoherent modulation schemes, although the $\hat{H}_k$ are not needed for demodulation in this case. The improvement can be achieved when an error control code is used in the system, in which case the channel knowledge can help the code decoder do a better job.

Frequency Domain Approach for Channel Estimation

Frequency domain channel estimation can be performed with two main approaches. The first method uses training data transmitted on every subcarrier. This method is applicaple to WLAN systems. The second method uses correlation properties of the frequency response of a multipath channel, and training information that is transmitted on a subset of the subcarriers. This approach is used in broadcast OFDM systems.

Channel Estimation Using Training Data

The long training symbols in the WLAN preamble, Figure 2.4, facilitate an easy and efficient estimate of the channel frequency response for all the subcarriers. The contents of the two long training symbols are identical, so averaging them can be used to improve to quality of the channel estimate. DFT is a linear operation, hence the average can be calculated before the DFT. Then only one DFT operation is needed to calculate the channel estimate. After the DFT processing, the received training symbols $R_{1,k}$ and $R_{2,k}$ are a product of the training symbols X_k and the channel H_k plus additive noise $W_{l,k}$.

Equation 2.69

$$R_{l,k} = H_k X_k + W_{l,k}$$

Thus the channel estimate can be calculated as

Equation 2.70

$$\hat{H}_k = \frac{1}{2}(R_{l,k} + R_{2,k})X_k^*$$

Equation 2.71

$$= \frac{1}{2}(H_k X_k + W_{1,k} + H_k X_k + W_{2,k})X_k^*$$

Equation 2.72

$$= H_k |X_k|^2 + \frac{1}{2}(W_{1,k} + W_{2,k})X_k^*$$

Equation 2.73

$$= H_k + \frac{1}{2}(W_{1,k} + W_{2,k})X_k^*$$

where the training data amplitudes have been selected to be equal to one. The noise samples $W_{1,k}$ and $W_{2,k}$ are statistically independent, thus the variance of their sum divided by two is a half of the variance of the individual noise samples.

Channel Estimation Using Pilot Subcarriers and Interpolation

If the subcarriers are very close to each other, like in broadcast standards (e.g. DVB) the frequency domain correlation of the channel frequency response can be used to estimate the channel, even if training data is not available for all the subcarriers. The *coherence bandwidth* of the channel is usually approximated [20] as the inverse of the total length of the channel impulse response

Equation 2.74

$$(\Delta f) \approx \frac{1}{\tau_{\max}}$$

When the subcarrier spacing is much less than the coherence bandwidth, the channel frequency response on neighboring subcarriers will be practically identical. For example, the DVB standard takes advantage of this by not sending training information on all subcarriers, hence more data can be transmitted. The pilot subcarriers in DVB are divided into two sets. The first set is regular pilot subcarriers that are similar to the WLAN pilots subcarriers; known data is constantly transmitted on them. The second set is called *scattered* pilots; known data is transmitted only intermittently transmitted on the subcarrier. Then the channel estimation is performed by interpolating the frequency response of the subcarriers that do not have training information from the known subcarriers. We do not describe this algorithm in detail, a description of a method can be found in Speth et. al. [25].

Time Domain Approach for Channel Estimation

The channel estimation can also be performed using the time domain approach, before DFT processing of the training symbols. In this case, the channel impulse response, instead of the channel frequency response, is estimated. The following derivation of the channel estimator uses IEEE 802.11a standard training symbols as an example, but the same approach can be applied to any OFDM system that includes training symbols. The received time domain signal during the two long training symbols is

Equation 2.75

$$r_{1,n} = h * x_n + w_{1,n}$$

The time domain convolution can be expressed as a matrix vector multiplication. The circular convolution matrix is formed from the training data as

Equation 2.76

$$X = \begin{bmatrix} x_1 & x_{64} & x_{63} & \cdots & x_{64-L+2} \\ x_2 & x_1 & x_{64} & & x_{64-L+3} \\ \vdots & & \vdots & & \vdots \\ x_{63} & x_{62} & & & x_{64-L} \\ x_{64} & x_{63} & & \cdots & x_{64-L+1} \end{bmatrix}$$

The parameter L defines the maximum length of the impulse response that can be estimated, and X is in general a rectangular matrix. The channel impulse response vector is

Equation 2.77

$$h = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_L \end{bmatrix}$$

Then the convolution is expressed as

Equation 2.78

$$r_{l,n} = h * x_n + w_{l,n} = Xh + w_l$$

The channel impulse response estimate can then be formed by

Equation 2.79

$$\hat{h} = \frac{1}{2} X^{\dagger}(r_{1,n} + r_{2,n})$$

Equation 2.80

$$= \frac{1}{2} X^{\dagger}(Xh + w_1 + Xh + w_2)$$

Equation 2.81

$$= X^{\dagger}Xh + \frac{1}{2} X^{\dagger}(w_1 + w_2)$$

Equation 2.82

$$= h + \frac{1}{2} X^{\dagger}(w_1 + w_2)$$

where $X^{\dagger}$ denotes Moore-Penrose [22] generalized inverse of *X.* The channel frequency response estimate is then formed by calculating the DFT of the impulse response estimate

Equation 2.83

$$\hat{H} = DFT\left\{\hat{h}\right\}$$

Exercise 5 *The simulation tool implements the frequency domain channel estimation algorithm. Write a Matlab script for the time-domain algorithm and compare the performance of the algorithms.*

Exercise 6 *The simulation tool allows to run simulations with perfect synchronization or with partial or full synchronization. Experiment how the performance of the system degrades compared to perfect synchronization as the various synchronization algorithms are activated.*

Analysis of the Time Domain and Frequency Domain Approaches for Channel Estimation

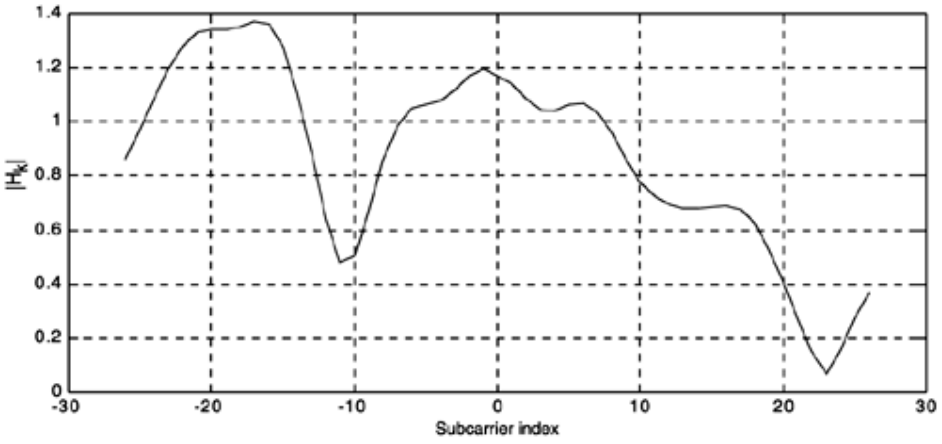
The advantage of the time domain approach is improved performance, when the maximum length of the impulse response can be limited to a number significantly less than the number of subcarriers. The rationale is that the frequency domain estimator has to simultaneously estimate all the subcarriers, whereas the time domain estimator needs to estimate only the taps of the impulse response. When the number of subcarriers is large compared to the number of channel taps, the signal energy used to estimate each H_k is significantly less than the signal energy used to estimate each h_n. In other words, it is easier to estimate fewer parameters given a fixed amount of data. For example, in the IEEE 802.11a system the number of subcarriers is 52, and the maximum length of the channel can be assumed to be less than the cyclic prefix length of 16 samples. Thus the frequency domain algorithm estimates more than three times the number of parameters than the time domain algorithm.

The drawback of the time domain method is that additional computations are required. The $\frac{1}{2} X^{\dagger}(r_{1,n} + r_{2,n})$ operation requires $64 \cdot L$ multiplications, which are not needed at all in the frequency domain estimator. This is the usual engineering trade-off; better performance usually implies higher costs in one form or another.

Enhancing the Channel Estimate

The correlation of the channel frequency response between different subcarriers can be used to further improve the quality of the channel estimate. Figure 2.16 shows an example of the channel amplitude response for a IEEE802.11a system. It can be seen that the neighboring subcarriers are highly correlated. We will not go to the details of this method, but refer to [12] for a full discussion. The paper shows that in some conditions performance very close to ideal channel estimation can be achieved. Unfortunately, this improvement does not come without a price; significantly more computations are required than to calculate the simple frequency domain channel estimator.

Figure 2.16. Example of a channel amplitude response.



Clear Channel Assessment

Clear Channel Assessment (CCA) is not strictly an OFDM synchronization algorithm, but a network synchronization method. However, it is closely related to packet detection problem and thus it is included in this chapter. The overall system performance of IEEE 802.11 WLAN is crucially dependent on high quality CCA algorithm, so the algorithm has to be carefully designed. The IEEE 802.11a standard has two different requirements for the CCA algorithm. The first is the detection probability when a preamble is available and the second is the detection probability when the preamble is not available. The requirement for the first case is that the CCA algorithm shall indicate a *Busy* channel with >90% probability within $4\mu s$ observation window, if a signal is received at -82 dBm level. For the second case, when the known preamble structure is not available, the requirement is relaxed by 20dB. Thus a signal detection probability of >90% within $4\mu s$ observation for a received signal level of -62dBm is required.

The first requirement can be approached using any of the packet detection algorithms presented in this chapter. The methods that use the known preamble structure are capable of reaching the performance level required in the standard. The only possible approach for the second requirement is to use the received signal energy detection method. The reason is the maximum allowed length of $4\mu s$ for the observation time. This is equal to the symbol length in the IEEE802.11a system. Thus, during the $4\mu s$ time, the receiver might receive one whole OFDM symbol or the end of one symbol and the start of the next one. In this case, there is no available signal structure that could be taken advantage of to improve the detection probability. The OFDM waveform itself resembles white noise, and inside $4\mu s$ windows it cannot be guaranteed that a whole OFDM symbol is received, which would allow to use the cyclic prefix in the CCA algorithm. As a conclusion, the receiver can only measure the total received energy and test whether it is above the specified limit.

Signal Quality

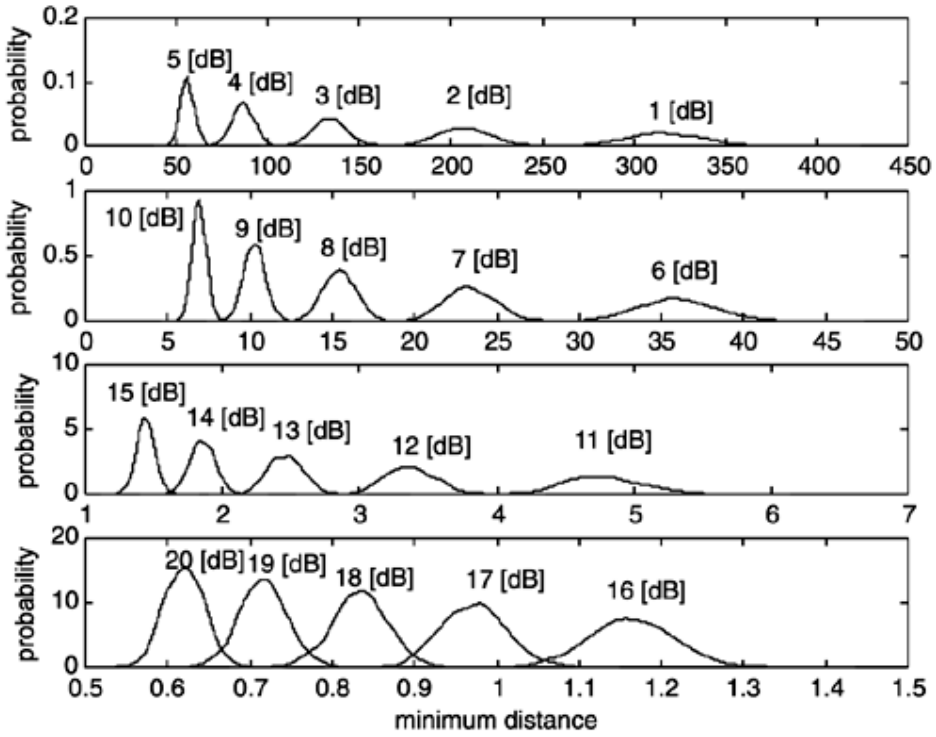
Signal quality estimation is not strictly a synchronization algorithm, but a network performance issue. WLAN systems define several data rates that can be used to transmit data; thus, to achieve the best overall network performance, data should be transmitted on the highest rate that can be reliably transmitted under the current channel conditions. For the transmitter to be able to decide what data rate to use, an accurate estimate of the quality of the channel has to be available. This is the task of the signal quality algorithm.

The double sliding window algorithm showed how its output could be used to estimate the received *SNR* in Equation 2.7. This is a quite simple method if the receiver implements the double sliding window detector. However, the form of the estimator does not take into account fading channel conditions. Only the total received signal energy is considered. In some cases, this behavior can lead to an overly optimistic estimate of the signal quality. For example, the total channel impulse response energy may be quite high, indicating high SNR, but the frequency response of the channel can have very deep nulls. In this case, the actual quality of the channel could be significantly less than the estimate given by double sliding window algorithm.

A quite different approach to the signal quality problem can be taken by using the convolutional error control code specified in the IEEE802.11a system. The convolutional error correcting code used in IEEE 802.11a standard is described in Chapter 3, "Modulation and Coding." The standard method to perform convolutional code decoding is the Viterbi algorithm that is described in Chapter 1, and in more detail in [20]. We will not describe the algorithm, but assume knowledge of its basic principles. Essentially the Viterbi algorithm calculates the distance between the maximum-likelihood transmitted code word and the received data. This distance can be used as an indication of the quality of the channel. When the channel is in good condition, the distance between the transmitted and received signal is small; this is indicated by the Viterbi algorithm as a small cumulative minimum distance at the end of the decoding process. When the channel is in a bad state, the distance between the transmitted and received signal is larger, and the cumulative minimum distance will be large. Figure 2.17 shows the probability density functions of the cumulative minimum distance for several different *SNRs*. The *SNR* estimator is formed by dividing the cumulative minimum distance into ranges according to the value of the PDFs. The *SNR* value

corresponding to the PDF that has the largest value at a given cumulative minimum distance is the estimate of the signal quality. For example, a cumulative minimum distance between 2.2 and 2.9 indicates a 13dB SNR. From Figure 2.17 it can be seen that for a large range of received *SNRs* a very good accuracy of 1dB resolution can be achieved.

Figure 2.17. Cumulative minimum distance in Viterbi decoding.



Bibliography

[1] J.-J. van de Beek, M. Sandell, M. Isaksson and P.O. Borjesson, *"Low-complex frame synchronization in OFDM systems,"* in Proceedings of IEEE Internatial Conference Universal Personal Communications, Toronto, Canada, Sept. 27–29, 1995, pp. 982–986

[2] J-J. van de Beek, M. Sandell, P. O. Börjesson, *"ML Estimation of Time and Frequency Offset in OFDM systems," IEEE Transactions on Signal Processing*, Vol. 45, No. 7, July 1997

[3] G. Casella, R. Berger, *"Statistical Inference,"* Duxbury Press, California, 1990

[4] F. Daffara, A. Chouly, *"Maximum Likelihood Frequency Detectors for Orthogonal Multicarrier Systems,"* Proceedings of IEEE International Conference on Communications, Geneva, Switzerland, May 22–26, 1993, pp.766–771

[5] F. Daffara, O. Adami, *"A New Frequency Detector for Orthogonal Multicarrier Transmission Techniques,"* Proceedings of IEEE Vehicular Technology Conference, Chicago, IL, July 25–28, 1995, pp.804–809

[6] S. A. Fechtel, *"OFDM Carrier and Sampling Frequency Synchronization and its Performance on Stationary and Mobile Channels," IEEE Transactions on Consumer Electronics*, Vol. 46, No. 3, August 2000, pp. 438–441

[7] HiperLAN/2, *"Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer,"* ETSI TS 101 475 V1.2.1 (2000–11)