**Lab 5: 802.11a Image Transmission and Reception**

|  |  |
| --- | --- |
| **Author** | Name： 吉辰卿 Student ID: 11911303 |
| **Introduction:**  This experiment content is very rich: We spent four weeks, discussing how to use 802.11a to carry on the image data transmission and recovery problem. Now comes the third week: In this experiment, we mainly discuss how the receiver restores packet bits in 802.11a and restores the image sent by these packet bits in a certain order. Although this process is a bit cumbersome, when we understand this procedure carefully, it is very meaningful for us to understand the whole transmission process and the terminology in the protocol. So in this experiment, we used Teacher Wu to extract two sets of pre-recorded data, and recovered the sent images respectively by importing the two sets of data. In addition, we also found the position of the wrong packet detected by the receiver and the constellation diagram when the error occurred.  **Theoretical analysis:**   1. **How to decode the data field in 802.11a?**   In this experiment, we basically used the program in the previous experimental class. However, in this experiment, we are more concerned about how the packet is received and processed by the receiver to restore the image sent. This is done primarily in the ReceiverProc function in MATLAB, so I will focus on this process:  20220407154605  20220407154615  Figure 1. The design process of ReceiverProc Function design process  The more than 180 lines of code for accepting a package are shown above, and I'm going to summarize the process:   1. Get the required information in the PSDU   In this part, we mainly get the index range of training sequence and data related fields in the PSDU header in the package. By running the program, we can see that the index length of both STF and LTF is 160, while the index length of SIG field is 80.   1. Downsample the received signal   In this part, we also import pre-recorded data from the USRP, so we need to further underground sample the received signal. The important thing to note here is that during this downsampling process, we want to set the oversampling factor to 1.   1. Determine the minimum length of a received packet and use it as a base unit to determine whether a packet has been received   In this part, we will determine the length of the smallest packet received, which we set to 160, or 10 OFDM symbols. In the subsequent receiving process, we use this minimum length to judge whether a packet has been received.   1. Configure the CRC for MPDU and calculate the EVM   In this step, we need to configure the CRC check on the received packets to ensure that the CRC check on the receiving end is consistent with that on the transmitting end, so that the received packets can be correctly processed.   1. Receive cycle processing   This time comes the core part of our entire program, which is also mentioned in the screenshot of the programming flow above. In this part, we first detect the starting value of STF, that is, estimate the starting position of the whole packet. After detection, we get an index of the starting value. Then, we assume that the detected starting position is the starting position of STF, and conduct frequency deviation correction for THE STF/LTF/SIG field. After correction, we carry out symbolic synchronization for the LTF sequence. If the symbol synchronization fails at this time, it will prove that our previous detection of the STF starting value is wrong, and we need to go back to the detection again. At the time of redetection, we need to add the minimum length of a packet (160) to the result of the previous detection position at the beginning of this detection, and then re-run the previous step at the new position for the cycle detection. Finally, at some point, when the LTF symbols are synced into dogs, we think we have found the correct starting position of the STF. The flow code of the above cyclic detection is as follows:  **20220407163109**  **20220407163151**  Figure 2. Cyclic detection to correctly find the starting position of the STF  After completing the above steps, the STF with the correct starting position detected, together with the LTF and SIG fields following it, are calibrated for coarse and precise frequency offset. At the same time, because of the special properties of LTF, we can also use it for channel estimation and noise estimation. After that, we try to recover the L-SIG field. If the recovery fails, it proves that the search for the starting position of the packet is still some problems, and the search needs to continue in the loop to the front; If the SIG domain can be successfully recovered, we can parse the L-SIG field and then perform frequency offset correction for the entire data field. The MATLAB code for the above process is as follows:  **20220407165505**  **20220407165404**  Figure 3. Synchronize LTF confirmation and parse MCS information in SIG  After that, we create a non-HT object on the receiving side that also retrieves the index of the non-HT data field. We then use the results of the previous channel estimation to recover the bits in the PSDU data and the modulation symbols after equalization. While doing this, we can display the constellation of the currently received packet. After that, we remove the FCS from the MAC header and perform CRC checks. Finally, we process the MAC header information, parse the MAC header and get the packet number. At the same time, we can output the results that show the decoding and update the starting location of the next packet search. Note that if duplicate packets are detected, we can end the unpacking loop, indicating that we are done unpacking. The MATLAB code for this process is shown below:  **20220407172326**  **20220407172337**  **20220407172345**  Figure 4.Collate the received data packets and print the results  The functions used in all the above processes are summarized in the following flow chart:  20220407172945  20220407172955  20220407172959  Figure 5.The main function used by the receiver to process packets  **Lab results & Analysis：**  **Task1: Explain the functions of the following six subcomponents respectively**  **Result:**   1. **helperPacketDetect.m**   Data packets are detected to estimate the starting position of STF in data packets. The detection method is autocorrelation.   1. **wlanCoarseCFOEstimate.m**   After the starting position of STF sequence of packets is estimated in each cycle, perform the coarse frequency offset correction for estimated STF/LTF/SIG and determine their coarse frequency deviation.   1. **helperFrequencyOffset.m**   For STF/LTF/SIG, perform the coarse frequency offset correction, and for data fields, perform the precise frequency offset correction.   1. **helperSymbolTiming.m**   Perform the symbol synchronization for LTF.   1. **helperInterpretLSIG.m**   Parse the L-SIG field and recover the packet parameters such as MCS.   1. **wlanNonHTDataRecover.m**   Recover the PSDU bits and the modulation symbol after equalization based on the results of channel estimation.  **Task2: Image recover with Pre-Recorded data and further analyze the results.**  **Result:**  In this experiment, Mr. Wu pre-recorded two pieces of data from USRP for us, and now we analyze these two pieces of data respectively:   1. **rxWaveform.mat**   Firstly, Let's look at the search process for the starting position of STF in the theoretical analysis step, and the results are as follows:  20220407191002  Figure 6.The circular process of finding the starting position of an STF field  After that, let's examine the received constellation of each packet. We can see that, except for the 14th packet, the constellation of all packets looks like this:  **20220407191801**  Figure 7.The Constellation of received packets (Except for the 14th packet )  However, we found that the constellation of the 14th packet was quite different from that of other packets, as shown below:  20220407192043  Figure 8.The Constellation of received 14th packet packets  After that, let's arrange the sequence numbers of the packets we receive as follows:  20220407192412  We can see that the number of the 14th package is 255 instead of 6 as we should see.  Finally, we look at the restored image at the receiver and compare it with the image transmitted by the transmitter, as shown below:  20220407192403  Figure 9.The restored image compared with the original image  By comparison, we found that the restored image at the receiver had a large green strip more than the original image.  **Analysis:**  **Based on the above results, we can easily draw the conclusion that there is an error in unpacking the 14th packet with packet number 6.**  Let's briefly analyze these results: Figure 6 shows the process of correctly finding the starting position of STF in the process of understanding packets. As we can see from the above, each search moves a minimum pack length (160) to the right from the previous position (see the red dots in each image). When the second search does not meet the requirements, we move to the right by a minimum packet length and continue the search. When the starting position of the STF is found and the corresponding LTF symbol is synchronized correctly, we stop looking. Looking at the comparison between Fig. 7 and Fig. 8, the quality of the constellation in Fig. 8 is obviously lower than that in Fig. 7, which indicates that the receiver made a mistake in processing the 14th packet (packet number 6), resulting in the chaos of the constellation. Finally, when we look at the order of the received packet, we can see that the number of the 14th packet has changed to 255(it should have been 6), which again reflects the error of the receiver in processing the 14th packet. In fact, this error can be reflected in Fig. 9. From the Fig. 9, we can see that there is a lot of noise and interference in the picture after receiver unpacking and stitching, which is caused by a major error in the 14th packet unpacking.   1. **rxWaveform2.mat**   Firstly, Let's look at the search process for the starting position of STF in the theoretical analysis step, and the results are as follows:  20220407194435  Figure 10.The circular process of finding the starting position of an STF field  After that, let's examine the received constellation of each packet. We can see that, except for the 14th packet, the constellation of all packets looks like this:  20220407195451  Figure 11.The Constellation of received packets (Except for the 14th packet )  However, we found that the constellation of the 14th packet was quite different from that of other packets, as shown below:  20220407195554  Figure 12.The Constellation of received 14th packet packets  After that, let's arrange the sequence numbers of the packets we receive as follows:  20220407200355  We can see that the number of the 14th package is 255 instead of 6 as we should see.  From this figure we can see that it seems that the package numbers in this package sequence are all correct, but why is the constellation of the 14th package (the package number is 13) so confused? See our analysis below.  Finally, we look at the restored image at the receiver and compare it with the image transmitted by the transmitter, as shown below:  20220407201110  Figure 13.The restored image compared with the original image  By comparison, we found that compared with the original image, there are many irregular vertical lines from left to right in the restored image of the receiver, and the number of vertical lines is exactly 15.  **Analysis:**  **Based on the above results, we can easily draw the conclusion that：In the process of image unpacking based on rxWaveform2.mat data, some errors appeared in each packet, and the error in 14th packet (packet number 13) was more serious than other packets.**  Let's briefly analyze these results: From the comparison between Figure 11 and Figure 12, we can see that the constellation of the 14th received packet is chaotic compared to the other packets, but from the number of packets received in the packet sequence, it seems that there is nothing wrong with these packets. So why does this happen? The answer is in our receiving end's CRC check result, as follows:  20220407202516  20220407202510  Figure 14.The CRC result of the 14th packet is compared with that of other packets in rxWaveform2.mat(The CRC result is the same. Here, only the 13th packet is used as an example).  As can be seen from the figure above, not only the 14th packet, but other packets failed the CRC check at the receiving end, which proves that there are some problems in the 15 packets resolved by us. After the analysis, I was surprised to find that there were 15 thin green bars from left to right in the image recovered by the receiver, which represented the errors of each packet in the unpacking of 15 packets. From here, we can see that in the waveform data, the result of unpacking is that all the 15 packets have errors, which can be seen from the CRC check of the receiver and the image recovered by the receiver. However, the 14th packet (packet number 13) produced more serious errors than the other packets, which can be clearly seen in its constellation.  However, in rxWaveform. mat, why does the recovered image only have a green block, and this block is larger than the green bar in this waveform recovered image?  20220407204812  20220407204805  Figure 15.The CRC result of the 14th packet is compared with that of other packets in rxWaveform.mat(The CRC result is the same. Here, only the 13th packet is used as an example).  As can be seen from the figure above, in rxWaveform.mat, only the CRC check of the 14th packet fails, while the CRC check of the other packets is successful. This reflects the fact that other packets were unpacked without error, so there is no green bar in the restored image. In the 14th packet, not only the CRC check failed, but also the sequence number of the received packet and the corresponding constellation were wrong and chaotic. This indicates that in rxWaveform.mat, the unpacking of the 14th packet has a very serious error, which is much more serious than the error generated by each packet in the unpacking of rxWaveform2.mat. This can also be seen in the image recovered by the receiver. The green bar caused by this error is much thicker than the green bar recovered by rxWaveform2.mat. | |
| **Experience**  Through this lab, I have a deeper understanding of how packets are processed at the receiver in 802.11a. Through in-depth understanding of the ReceiverProc.m function, I have a further understanding of how the receiver decodes the packets in 802.11a image transmission：converts them into bits, and finally splicing them back into a complete image. Finally, by reading two groups of pre-recorded data from USRP, I find the errors in some packets of the receiver in the process of unpacking, and further verify the existence of these errors through the images recovered by the receiver.  Finally, I paste the screenshot of the class exercise in the last lab class:  20220407213141  Figure 16. Class exercise 1  20220407213207  Figure 17. Class exercise 2  20220407213230  Figure 18. Class exercise 3 | |
| **Score** | 自评分数：98 |