Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.1:

In a Hamming code word a check bit toggles. Can the mistake be detected and corrected and if yes, how?

Solution:

When checking the parity of a certain check bit the wrong parity will be detected and the counter is incremented by the number of the particular check bit. Obviously other check bits will not fail because all other bits are correct. Note that the counter does already contain the number of the erroneous check bit. In other words: The normal algorithm of verifying bits considers data bits as well as check bits. No special care has to be taken.

Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.2:

The following Hamming code word is given: 01111001111. Create an error with as few changes as possible that can not be detected.

Solution:

Check bits are turquoise
toggled bits are red

```
12345678901    12345678901
01111001111    11111000011
```

For a minimum of toggled bits a data bit should be chosen which influences as few check bits as possible. Some data bits have only 2 check bits which care for them, e.g., bit number 9 (=8+1). So toggle 9, 8 and 1 and check whether 1 and 8 still result in an even parity (together with the data bits for which they are responsible).

Note that less than 3 bit changes are not possible, because error correcting codes have a distance of (2e +1) if e denotes the number of errors which can be corrected.

Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.2: (continued)

Now generate a bit error changing as many bits as possible.

Solution:

```
12345678901    12345678901         12345678901
01111001111    01111001111   result: 10000110000
```

Exercise 2.3:

A number of d bit error should be corrected. Explain why a distance of 2d is not sufficient for the code?

Solution:

The reason why the distance has to be 2d+1 and not only 2d is that for every incorrect code word there has to be a short and a long way to the next correct code word. In the case of a distance of 5 it is always possible to change 1 or 2 bits (namely those which are incorrect) in order to get to the next correct code word. Changing 3 bits however is not sensible because it would already be sufficient to change another 2 bits to obtain a valid symbol. However if the distance was e.g., 4, there would be two possibilities to change 2 bits in order to obtain a valid code word which would not be unique.

Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.4:

In the last lecture we have seen an estimation of how many redundant bits are necessary to detect and correct 1 bit errors. Now do the same estimation for 2 bit errors. It is not necessary to find a particular code, only a lower bound for the number of check bits is of interest.

How many bits are necessary to protect a 7 bits ASCII code against at most 2 toggled bits?

Solution:

The code should contain 2^m data bits and a yet unknown number of r check bits.
Remark: The code will contain 2^{m+r} symbols which are divided into 2^m valid symbols and $2^{m+r}-2^m$ invalid ones.

Either one- or two bit errors can occur:

1 bit errors:	For each of the valid code words an invalid one can be created by simply swapping one of the n bits.
2 bit errors:	Again one of the n bits can be toggled. For the second error bit (n-1) possibilities remain. At a first glance n(n-1) possibilities evolve. But since the order of the toggled bits does not matter the number of possibilities shrinks by 50%. This means n(n-1)/2 erroneous code words have to be considered for every valid code word.
No error:	Of course we must not forget to consider the valid code words.

Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.4: (continued)

Solution:

$$\left(\underbrace{n}_{\text{1 bit error}}+\underbrace{\frac{n(n-1)}{2}}_{\text{2 bit errors}}+\underbrace{1}_{\text{valid code word}}\right)2^m\leq 2^{r+m}$$

$$\left((m+r)+\frac{(m+r)(m+r-1)}{2}+1\right)2^m\leq 2^{r+m}$$

$$\frac{m+r+(m+r)^2}{2}+1\leq 2^r$$

$$\frac{7+7+(7+7)^2}{2}+1\leq 2^7$$

$$106\leq 128$$

In order to protect the 7 bit ASCII code against 2 simultaneous bit errors at last 7 check bits are necessary.

Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.5 (a):

For a given transmission channel there is an average of 1 error in 4000 bits. These single bit errors are statistically independent. A single packet consists of 128 bytes. It is either transmitted fully and correctly or not at all. The receiver can detect whether a packet was transmitted free of errors without any additional costs. If an error occurred the receiver asks the sender only once for retransmission. The request for retransmission is considered to be an ordinary packet of 128 bytes. If an error occurs in such a request it is treated as if no request was sent.

How high is the overall data rate in this scenario (in percent of the data rate that could theoretically be achieved if no error occurred)?

Solution: (a)

A bit is transmitted correctly with a probability of $\frac{3999}{4000}$

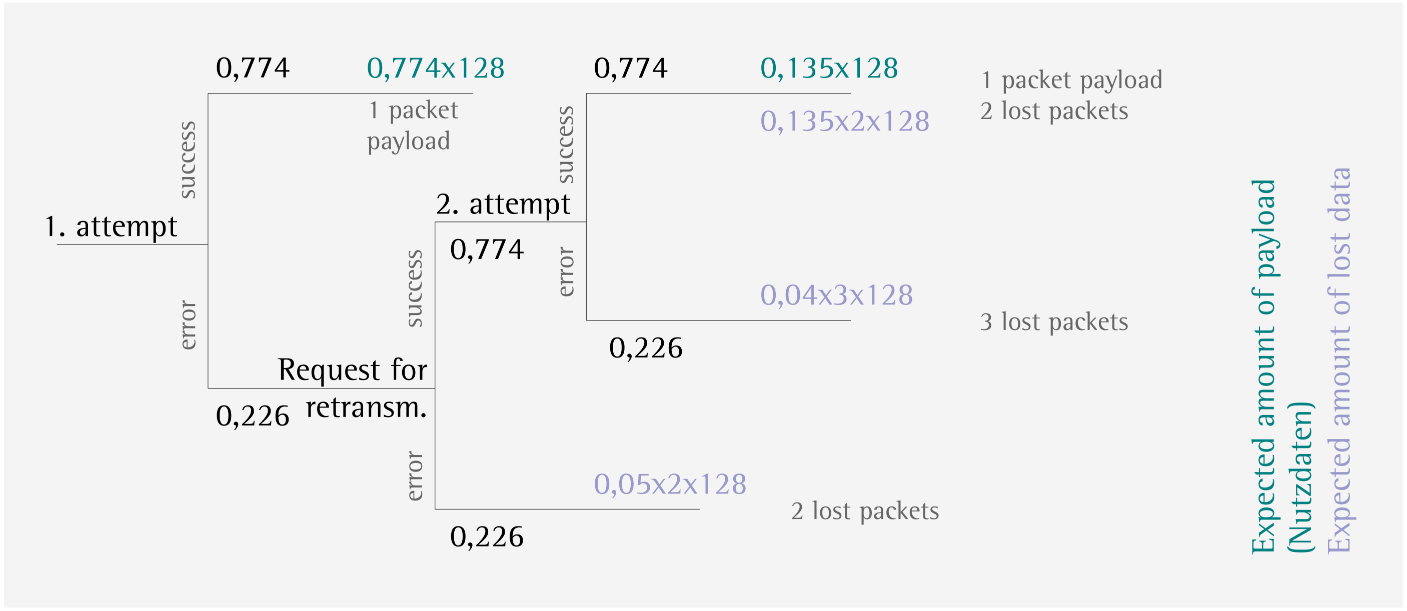
128x8=1024 Bit with a P' of $\left(\frac{3999}{4000}\right)^{1024} \approx 77{,}4\,\%$

An error occurs in all other cases with a P' of $1-\left(\frac{3999}{4000}\right)^{1024} \approx 22{,}6\,\%$

Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.5 (a):



In general we can expect:

0,774x128 + 0,135x128 = ca. 116 bytes of payload (expectation value)
0,135x2x128 + 0,04x3x128 + 0,05x2x128 = ca. 63 bytes lost data (expectation value)

Utilization of channel with useful data: $\frac{116}{116+63} \approx 64{,}8\,\%$

Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.5 (b):

To make things easier we assume that a bit error occurs only once per packet. Rather than asking the sender for retransmission we choose to employ forward error correction, e.g., using the Hamming code from the lecture (the code itself is not of importance here).

How high is the actual data rate (in percent) compared to the one that would theoretically be possible without forward error correction and if no errors occurred.

Solution (b)

In order to protect 1024 bits of data against 1 bit errors the following inequality has to hold:

$1024+r+1 <= 2^r$

for r =11 we obtain

$1024+11+1 <= 2048$

The throughput (with useful data) in this scenario is $\frac{1024-11}{1024} \approx 98{,}9\%$

Conclusion: Forward error correction can make more sense than using a protocol for error recovery when it comes to equally distributed errors, e.g., while doing radio transmissions.

Exercise Sensor Networks

Lecture 2: Error correction

Exercise 2.6: Explain why protecting n bits of data against single bit errors with forward error correction requires only O(log(n)) check bits while protecting a fixed number of c bits against a number of n toggled bits requires O(2^n) check bits. Hint: Think of the table consisting of valid and invalid code words and of the method in which bits a checked in the Hamming code.

Solution:

Protecting against single bit errors can e.g., be done with the Hamming code in which n check bits take care of O(2^n)* data bits. Adding only one more check bit is enough to double the number of data bits which can be protected.

However when protecting code words of a fixed length against n errors (toggled bits) each invalid code word that might occur has to appear in the code a well. You can imagine the code as a (long) list of all possible code words. Some of them are meaningful and some of them are marked as invalid. Each valid code word will be accompanied with a couple of others that are invalid. With an increasing number of bits the possibilities for errors increase exponentially because each error bit can be combined with the other error bits as seen in exercise 2.4.

* ([2^n-n] to be precise)

Exercise Sensor Networks

Exercise 2.7:

A sender wants to transmit the following 32 bits

```
1110  10110100        10111010
0101  01011011        11101101
1001  01010101    =   00000010
0110  10110110        00001110
```

Calculate inverse matrix on receiver side

```
1110  1000  <-+       --+
0101  0100    |  --+    |
1001  0010    |    |  <-+
0110  0001  --+  <-+
---------
1000  1001
0101  0100       <-+
0111  1010  <-+  --+
0011  0101  --+
---------
1000  1001
0010  1110  --+
0100  1111    |
0011  0101  <-+
---------
1000  1001
0100  1111
0010  1110
0001  1011
```

Exercise Sensor Networks

Exercise 2.7:

A sender wants to transmit the following 32 bits

```
1001  10111010     10110100
1111  11101101  =  01011011
1110  00000010     01010101
1011  00001110     10110110
```