Mobile Systems

Lecture 5: Bit-error control

COMP28512

Steve Furber & Barry Cheetham

24 February 2015

Error correction & detection

- Mobile systems now transmit & receive packets.

1	0	0	1	..

- Bit-errors can occur due to noise affecting radio reception
- Try to achieve error free packet transmission by:
  - (i) Forward error correction (FEC)
  - (ii) Error detection & retransmission (ARQ)
  - (iii) A combination of (i) & (ii)

Forward error correction (FEC)

- Correction of bit-errors at receiver based on redundancy built into the transmission by.
  - appending 'check-bits', or
  - 'coding' to produce larger packets.
- 'Block coding' or 'convolutional coding' may be used.
- FEC decoder tries to correct any bit-errors.
- 'Error detection' can check whether all bit-errors have been corrected
- Can request 'ARQ' (retransmission) if bit-errors remain.

Block & convolutional coding

- Block codes used for both error detection & correction.
- Convolutional codes generally used for bit-error correction.

- Block-codes require the whole block of data to be available before it can be coded at the transmitter,
- Complete block of coded data must have been received before decoding can begin.

- Convolutional coding can start as soon as a few bits are available and can go on uninterrupted, in principle for ever.
- A convolutional decoder can start producing its error-corrected bit-stream once ≈ 50 bits have been received.
- Can go on decoding for as long as transmitter sends data.

Simplest block coding idea: parity

- 1010 has even parity, as number of ones is even.
  - Means that $1 \oplus 0 \oplus 1 \oplus 0 = 0$ ($\oplus$ = 'exclusive or' = 'xor')
- 1011 has odd parity, & $1 \oplus 0 \oplus 1 \oplus 1 = 1$
- Transmitter can append 'parity-bit' to 4-bit number to force parity to be always even: 1010**0** or 1011**1**.
- Receiver calculates parity using xor of 5 bits.
- If parity is odd, a bit-error must have occurred.
- Somewhere within the 5 bits.
- If parity-bit = 0, data may be correct.
- Or there may be an even number of bit-errors.

Odd parity

- Can use odd parity instead.
- Make number of 1's odd at transmitter
- Receiver calculates parity.
  - If parity is even, a bit-error must have occurred.
  - If parity is odd, data may be correct.
  - Or there may be an even no. of bit-errors.

- Question: Is it true that even parity detects an even number of bit-errors, & odd parity detects an odd number?

Hamming distance

- Hamming distance between two binary numbers is number of bits that are different.
- Obtained by xor-ing & counting the number of '1's:

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

C: 1 0 0 0 1 0 0 1
D: 1 0 1 0 1 0 1 0
C xor D: 0 0 1 0 0 0 1 1

- Hamming distance between C & D is 3
- Clearly, 3 bit-errors needed to convert C to D.

Relevance to error detection & correction

- Assume we have 5 numbers chosen so that Hamming distance between any two of them is ≥ 3 .

A 0000000
B 1100001
C 1010010
D 0110011
E 0110100

- If B is transmitted & **one** bit-error occurs, damaged number, B* say, cannot be another valid number.
- We can **detect** that there has been an error.
- If B* has one bit-error, its Hamming distance to B is 1.
- Distance to all other valid numbers at least 2.
- So B* can be **corrected** to B.
- If damaged number B* has 2 bit-errors, can still **detect** that bit-errors have occurred.
- But **cannot correct** them, as B* may **not** be closest to B .

Minimum Hamming distance

- Let min Hamming distance between any 2 numbers be d.
  - Error detection is possible if no. of bit-errors < d.
  - Error correction is possible if no. of bit-errors ≤ (d-1)/2.

Another example

- Assume four 4 codewords with min Hamming distance of 5:

A: 0000000000
B: 0000011111
C: 1111100000
D: 1111111111

- If we receive B with 2 bit-errors, it will be at distance of 2 from B & at least 3 from all other words.
- Taking shortest distance, can correct received number to B.
- For 3 or 4 bit-errors:
  - can detect that there are bit-errors,
  - trying to correct by finding shortest distance produces wrong code.

Hamming codes

- Assume you are sending 'm-bit' messages.
- Introduce r 'check-bits' chosen to make d = 3
- Allows detection of single & double bit-errors.
- Allows correction of a single bit-error.
- Bit-errors can occur in check-bits as well as message bits.
- Hamming codes of length 7 (m=4, r=3) given in textbooks.
- (Variations exist)
- Hamming codes are 'block codes'
- With m=4 & r=3, get (7,4) block code whose 'rate' is 4/7.

A (7,4) Hamming Code

- Let message bits to transmit be B_3, B_2, B_1, B_0.
- Add 3 extra bits P_2, P_1,P_0 to give

B_3	B_2	B_1	B_0	P_2	P_1	P_0

- Make B_3 B_2 B_1 P_0 have even parity - set $P_0 = B_3 \oplus B_2 \oplus B_1$ (miss out B_0)
- Make B_3 B_2 B_0 P_1 have even parity - set $P_1 = B_3 \oplus B_2 \oplus B_0$
- Make B_3 B_1 B_0 P_2 have even parity - set $P_2 = B_3 \oplus B_1 \oplus B_0$

At receiver

- Calculate 'receiver parities'
  - R0 = B3 ⊕ B2 ⊕ B1 ⊕ P0 (miss out B0)
  - R1 = B3 ⊕ B2 ⊕ B0 ⊕ P1 (miss out B1)
  - R2 = B3 ⊕ B1 ⊕ B0 ⊕ P2 (miss out B2)
- If no bit-errors, R0, R1 & R2 will be 0
- If just B0 in error, R1 & R2 will be 1.
- If just B1 in error, R0 & R2 will be 1
- If just B2 in error, R0 & R1 will be 1
- etc.

Syndrome table

R_2	R_1	R_0	Correction needed to
0	0	0	None
0	0	1	P_0
0	1	0	P_1
0	1	1	B_2
1	0	0	P_2
1	0	1	B_1
1	1	0	B_0
1	1	1	B_3

Another (7,4) Hamming Code

- Re-order & rename bits to give

B_3	B_2	B_1	P_2	B_0	P_1	P_0
A7	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**

Parity bits are now **A1**, **A2** & **A4**

Chosen to make the following have even parity:

A7, A5, A3, **A1** (1+2+4, 1+4, 1+2, 1)

A7,A6, A3, **A2** (2+1+4, 2+4, 2+1, 2)

A7,A6, A5, **A4** (4+1+2, 4+2, 4+1, 4)

Exercises

1. Work out the nice syndrome table for this code.
2. Why will 4 parity bits support up to 11 message bits?.
3. Design a (15,11) Hamming coder.
4. Design a (11, 7) coder. (This will not use every entry in syndrome table for single bit-error correction).
5. Why will r parity bits support up to $2^r - r - 1$ message bits?
   (With r parity bits, syndrome table has 2^r rows.
   It must be able to indicate a single error in:
   - one of the m message bits, or
   - one of the r parity bits, or
   - no bits at all

   This requires m + r + 1 rows. Therefore, $m + r + 1 \leq 2^r$.)

Interleaving

- burst errors
  - radio links often suffer bursts of errors
  - transmitting a block by column ensures only single-bit error per codeword row

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

Cyclic redundancy check (CRC)

- Block code for bit-error detection.
- Illustrate the concept as follows:
- Suppose we are transmitting a decimal number 139.
- Divide by 7 in integer arith & express remainder in binary.
- Gives 19 with remainder 6 or '110' in binary.
- Forget about the 19, but use '110' as check-bits.
- Same division done at receiver.
- If we get different remainder, a bit-error has occurred.
- Exactly 3 check-bits always produced.
- 'Generator' numbr 7 agreed in advance & carefully chosen.
- Not all combinations of bit-errors detectable by this method.
  - Any combination that adds or subtracts multiple of 7 not detected.

Real CRC checks & polynomials

- Can multiply by 10 before doing the division by 7.
- A decimal number may be expressed as a 'polynomial'

  $139 = 1 \times x^2 + 3 \times x^1 + 9 = p(x)$

- Binary numbers may be expressed as polynomials
  e.g. $10011001 = x^7 + x^4 + x^3 + 1 = p(x)$
- Real CRC checks do not use normal arithmetic
- They use different way of 'dividing' based on 'ex-or'.
- It is 'modulo 2' or Galois field arithmetic.
- The way we 'add', 'subtract', 'mult' & divide' is different.

'Summing'

- 'Sum' of 2 binary numbers is xor of each of their bits.
- Let P = 1 0 0 1 1 0 0 1 Q = 1 1 1 0 0 0 1 1
- Calculate 'sum' of P & Q as follows

$p(x) = 1.x^7 + 0.x^6 + 0.x^5 + 1.x^4 + 1.x^3 + 0.x^2 + 0.x + 1$
$q(x) = 1.x^7 + 1.x^6 + 1.x^5 + 0.x^4 + 0.x^3 + 0.x^2 + 1.x + 1$

'Sum' $= 0.x^7 + 1.x^6 + 1.x^5 + 1.x^4 + 1.x^3 + 0.x^2 + 1.x + 0$
$\equiv$ 0 1 1 1 1 0 1 0

- It's just the 'exclusive-or' of the bits.
- 'Subtract' is exactly same as 'sum'

'Division'

- Like long division except we use polynomials & 'xor' for '+'

$$
\begin{array}{l}
\qquad\qquad\quad \underline{x^7 + x^4 + x^3 + 1} \\
x^4 + 1 \;)\; x^{11} + x^8 \quad + x^3 + x^2 + x + 1 \\
\qquad\quad \underline{x^{11} \quad + x^7 \qquad\qquad\qquad \oplus} \\
\qquad\qquad x^8 + x^7 + x^3 + x^2 + x + 1 \\
\qquad\qquad \underline{x^8 \;+\; x^4 \qquad\qquad\qquad \oplus} \\
\qquad\qquad\quad x^7 + x^4 + x^3 + x^2 + x + 1 \\
\qquad\qquad\quad \underline{x^7 \quad + x^3 \qquad\qquad \oplus} \\
\qquad\qquad\qquad x^4 \quad + x^2 + x + 1 \\
\qquad\qquad\qquad \underline{x^4 \qquad\qquad + 1 \;\oplus} \\
\qquad\qquad\qquad\qquad x^2 + x
\end{array}
$$

- Result is $x^7 + x^4 + x^3 + 1$ with remainder $x^2 + x$
  10011001 with remainder 110. **CRC is 110.**

Practical CRC

- If same 'division' performed at receiver & we get different remainder, we know that a bit-error has occurred.
- G(x) known at transmitter & receiver & carefully chosen.
- Actually x^4+1 is not a good choice.
- In practice, for Mth order G(x), append M zeros to the data before the polynomial 'division'.
- Instead of $x^{11} + x^8 + \ldots$, divide $x^{15} + x^{12} + \ldots$ by G(x).
- Gives different remainder, same at transmitter & receiver.
- Result of 'division' is just discarded. Only need remainder!

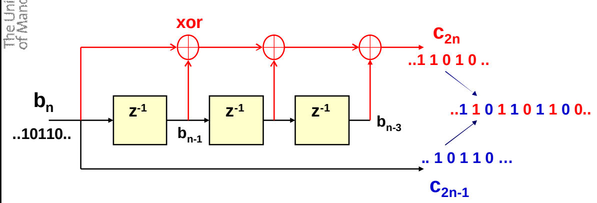
Limitations of CRC

- Not all combinations of bit-errors are detectable by a CRC.
- Any combination of bit-errors that 'adds' any 'multiple' of G(x) will not be detected.
- Try to make this unlikely by making order of G(x) large.

'Burst' errors & standard generators

- G(x) of order r causes all error 'bursts' of length $\leq$ r to be detected.
- Three standard generators are:
- CRC-8-ATM : $x^8 + x^2 + x + 1$ (100000111)
- CRC-16-IBM: $x^{16} + x^{15} + x^2 + 1$ (11000000000000101)
- CRC-32-IEEE:
  $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

A convolutional coder



- 'Rolling parity check' convolutional encoder with 'constraint length' 4.
- Half rate' coder: outputs of **lower** & **upper** branches interleaved.
- For each input bit b_n, **lower** branch reproduces b_n & **upper** branch produces parity check over b_n & 3 previous bits, i.e. $b_n \oplus b_{n-1} \oplus b_{n-2} \oplus b_{n-3}$.
- A $\mathbf{z^{-1}}$ box is just a 'delay by one bit' box.

Lecture 5 COMP28512 25

Strategy for decoder

- Encoder generates 'valid' sequences where each upper bit is 'xor' of current & 3 previous lower bits.
- Bit-errors can make the received bit-sequence 'invalid'.
- Select the valid sequence with minimum Hamming distance to the received sequence as the error corrected sequence.
- Can do this easily for short sequences of bits.
- With 8 bits, there would be 256 valid sequences of 16 bits.
- Can simple tabulate them.
- Method not feasible for longer sequences; e.g. 1024 bits
- 'Viterbi' algorithm performs selection in highly efficient way.

Lecture 5 COMP28512 26

Soft decision Viterbi decoder

- You now understand what a Viterbi decoder does.
- Conv coders are widely used esp. in mobile equipment.
- They may appear more complicated than block coders.
- But, thanks to Viterbi, the decoding is more efficient.
- Viterbi decoders use 'soft decisions'
- Instead of just '1' or '0' (hard decisions) can have.
  - 0.25 meaning 'probably 0',
  - 0.5 meaning 'don't know'
  - 0.75 could meaning 'probably 1.
- If we are expecting 0 or 4 volts for 0 & 1, then conversion to 'soft decision logic is obvious.
- Do you believe that soft decision decoding is better?

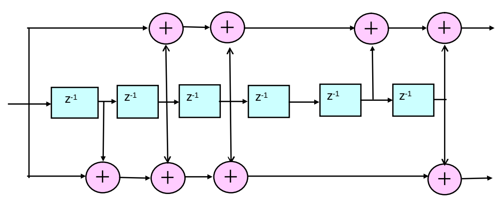
Lecture 5 COMP28512 27

Some terms

- 'Rolling parity' convolutional coder is:
  - 'Systematic' as original bit-stream appears in coder output.
  - Of 'constraint length' 4 as there are three z^{-1} delay boxes. (4 consec bits, current & 3 previous, available for computing outputs).
- 'Rolling parity' coder was presented for simplicity.
- Convolutional coder on next slide is widely used in practice.

Lecture 5 COMP28512 28

A standard half rate convolutional coder



- Described by 2 generator functions '1111001' & '1011011' which specify which bits are xor-ed together in lower & upper branches.
- Normally expressed in octal as '171' and '133' respectively.
- To convert to octal, split into groups of 3 starting from the right.
- 'Rolling parity' coder has generator functions: 1000 (10) & 1111 (17)

Lecture 5 COMP28512 29

Implementation of (171,133) coder

- Constraint length K=7 & each z^{-1} box requires a 'memory' variable.
- Call them X1, X2, …, X6, & initialise to zero.
- Append sequence of K-1 '0' bits to 'flush' z^{-1} memory to zero at end.
- Otherwise decoder will not correct some errors in final 6 data bits.
- This coder is used by IEEE802.11 standard

```
Inp = [1 1 1 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0]  %Test data
X1=0;X2=0;X3=0;X4=0;X5=0;X6=0;
for n=1:length(L)
    X=Inp(n);
   YU=xor(X,X2); YU=xor(YU,X3); YU=xor(YU,X5); YU=xor(YU,X6);
   YL = xor(X,X1); YL=xor(YL,X2); YL=xor(YL, X3); YL = xor(YL,X6);
   Y(2*n-1) = YL;  Y(2*n)=YU;
    X6=X5; X5=X4; X4=X3; X3=X2; X2=X1; X1=X;
end;
```

Lecture 5 COMP28512 30

Advantages of FEC for mobile systems

- Use of FEC in cellular mobile systems increases energy efficiency & effectiveness of spatial multiplexing by frequency re-use
- Transmitting at higher power is one way of making sure a signal is received with fewer errors.
- But high power signals carry further & cause interference over a wider range
- Makes re-use of frequency bands some distance away more difficult.
- Also quickly depletes a battery powered transmitter.
- Solution is to reduce transmission power & deal with resulting increase in bit-error rate using FEC.
- Solves cellular frequency re-use problem & reduces power consumption.

Lecture 5 COMP28512 31

Bit-rate & bandwidth

- Digital information conveyed by voltage pulses.
- e.g. pulses of amplitude +1 or 0 volts
  - binary signalling with 1 bit per pulse.
- Or of amplitudes 0, 1, 2 or 3 volts: 2 bits per pulse.
- Or of amplitudes 0, 1, 2, 3, 4, 5, 6, 7 volts: 3 bits per pulse.
- This is 'multi-level signalling'.
- Bandwidth efficiency attainable is 2 **pulses**/sec/Hz.
- If bandwidth is B Hz, can transmit up to 2B pulses/s.
- With binary signalling, bit-rate is 2B bits/s.
  - Bandwidth efficiency of 2 b/s per Hz.
- With 2 bits per pulse: 4 b/s per Hz
- With 3 bits per pulse: 6 b/s per Hz
- What is maximum possible?
- 'Channel capacity' depends on noise level.

Lecture 5 COMP28512 32

Shannon-Hartley Law

- Channel capacity: **$C = B \log_2(1 + S/N)$ bits/s**
- Max bit-rate achievable with arbitrarily small bit-error rate over channel affected by 'additive white Gaussian Noise'.
- Bandwidth B Hz.
- S/N = signal power / noise power ratio (not in dB),
- SNR = $10\log_{10}(S/R)$ is signal-to-noise ratio in dB.
- $C = B \log_{10}(1+S/N)/ \log_{10}(2) \approx 0.332 \log_{10}(1+S/N)$
- If S/N >>1, **$C \approx 0.332 \times B \times$ SNR in dB**.
- Valid when SNR = $10\log_{10}(S/N)$ >> 10 dB

- Exerc: What is C for 3kHz channel with 50dB SNR?
- Exerc: SNR needed to convey 54 Mb/s over 20MHz ?

Lecture 5 COMP28512 33

Conclusions & learning outcomes

- Roles of error correction & detection in mobile systems.
- Hamming distance relevant to error detection & correction.
- Differences between block codes & convolutional codes
- Understanding of Hamming codes & CRC checks.
- Use of polynomials to represent bit-streams.
- Polynomial 'sum', & 'division' defined & applied to CRC.
- Idea of convolutional coder illustrated by 'rolling parity' coder.
- Soft decision Viterbi decoder for max-likelihood decoding.
- Standard IEEE convolutional coder is easily implemented.
- Relationship between energy & bit-error rate mentioned.
- Shannon-Hartley Law gives channel capacity.

Lecture 5 COMP28512 34

Problems & discussion points

1. Why is a checksum that adds number of '1's not a good idea?
2. Why do packets need both error detection & correction?
3. Without FEC, how could 'soft decision' detection help you to correct a few bit-errors in a packet which failed its CRC at the receiver?
4. Parity check is simplest form of CRC. What is its generator polynomial?
5. Since we use even parity to check for an odd number of errors, can we use odd parity to detect an even number of errors?
6. Calculate the polynomial 'sum' of 100111 and 111001
7. Find the remainder when 101000 is 'divided' by 111.
8. Sketch a (117,155) convolutional coder.
9. Calculate the output of the (171,133) coder when the input is 11011
10. What is meant by a 'burst' of bit-errors?
11. Why does a 16-bit CRC detect all burst errors of length ≤ 16.

Lecture 5 COMP28512 35