**University of Manchester**

**School of Computer Science**

# COMP61232: Mobile Comms

## B5: Error control

**Barry Cheetham**



## Error correction & detection

- Mobile systems these days generally transmit & receive packets.
- Try to achieve error free packet transmission by:
  - (i) Error detection & retransmission (ARQ)
  - (ii) Forward error correction (FEC)
  - (iii) A combination of (i) & (ii)



## Error detection & ARQ

- ARQ is used for error correction when an error is detected.
  - A specific request may be made for a re-transmission.
  - Or failure to send an 'ack' may be trigger for re-transmission.
- Error detection with ARQ effective on **wired links**.
  - Ethernet uses only this mechanism.
- On radio, bit-errors occur more frequently.
  - The many re-transmissions that may be required with error detection & ARQ could be too expensive.
  - Radio channel resources more precious & limited than wired.
  - Phy layer synchronising preamble much longer for radio ( $\approx$ 180 $\mu$s for 802.11b WLANs & $\approx$ 6.4 $\mu$s for 10MHz Ethernet )
  - Re-transmission packets more expensive to send by radio.



## 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' (re-transmission) if bit-errors remain.



## Block & convolutional coding

- Block codes used for both error detection & correction.
  - Require 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 codes generally used for bit-error correction.
  - Coding can start as soon as a few bits are available
  - Can go on uninterrupted, in principle for ever.
  - Decoder can start producing its error-corrected bit-stream once $\approx$ 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, because $1 \oplus 0 \oplus 1 \oplus 0 = 0$
- 1011 has odd parity, because $1 \oplus 0 \oplus 1 \oplus 1 = 1$
- Transmitter appends 'parity-bit' to 4-bit number:
  - 10100 or 10111
  - Makes parity always even.
- 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 no. 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?

17/03/14 Comp61242: B5 7

## Hamming distance

- Hamming distance between two binary numbers is number of bits that are different.
- Hamming distance between '7-bit' numbers B & C is 4
- Obtained by xor-ing & counting the number of '1's:

| A | B | $A \oplus B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

B 1100001
C: 1010010
A xor B 0110011

- Clearly, 4 bit-errors would be necessary to convert A to B.

17/03/14 Comp61242: B5 8

## Relevance to block codes for error detection & correction

- Assume we have 5 numbers chosen so that Hamming distance between any two of them is $\geq 3$ .
  - A 0000000
  - B 1100001
  - C 1010010
  - D 0110011
  - E 0110100
- If B is transmitted & one bit-error occurs, the damaged number, B*, cannot be another valid number.
- We know there has been an error.
- Two bit-errors also detected
- 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.

17/03/14 Comp61242: B5 9

## 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 $\leq$ (d-1)/2.

17/03/14 Comp61242: B5 10

## Hamming codes

- Assume you are sending 'm-bit' messages.
- Introduce r 'check-bits' chosen 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.

17/03/14 Comp61242: B5 11

## 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 $P_2 = B_3 \oplus B_1 \oplus B_0$

17/03/14 Comp61242: B5 12

## At receiver

- Calculate ‘receiver parities’
  - $R0 = B3 \oplus B2 \oplus B1 \oplus P0$ (miss out B0)
  - $R1 = B3 \oplus B2 \oplus B0 \oplus P1$ (miss out B1)
  - $R2 = B3 \oplus B1 \oplus B0 \oplus P2$ (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).



## Interleaving

- Radio links often suffer bursts of errors
- Transmitting a block by column spread then out in time.
- Can ensure only single-bit error per ASCII char.

| 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’ number 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'.



## '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

$$\begin{array}{rl}
p(x) = & 1.x^7 + 0.x^6 + 0.x^5 + 1.x^4 + 1.x^3 + 0.x^2 + 0.x + 1 \\
q(x) = & 1.x^7 + 1.x^6 + 1.x^5 + 0.x^4 + 0.x^3 + 0.x^2 + 1.x + 1 \\
\hline
\text{'Sum'} = & 0.x^7 + 1.x^6 + 1.x^5 + 1.x^4 + 1.x^3 + 0.x^2 + 1.x + 0 \\
\equiv & 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0
\end{array}$$

- 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 '+'

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

- Result is $x^7 + x^4 + x^3 + 1$ with remainder $x^2 + x$

  10011001 with remainder 110. **CRC is 110.**



## 'Division' using binary numbers

```
              1 0 0 1 1 0 0 1
1 0 0 0 1 ) 1 0 0 1 0 0 0 0 1 0 0 1
            1 0 0 0 1  xor
            0 0 1 1 0
              0 0 0 0 0  xor
              0 1 1 0 0
                0 0 0 0 0  xor
                1 1 0 0 0
                1 0 0 0 1  xor
                  1 0 0 1 1
                  1 0 0 0 1  xor
                  etc.
```



## Comparision

- Confirms that '100100001001' 'divided' by '10001' in polynomial division is '10011001' with rem 0.
  - Same result as we obtained by direct polynomial division,
- Different from ordinary division.
  - Methodology is similar (& simpler)
  - Similarities & differences with ordinary arithmetic are interesting,
  - Suggest an easy way of programming the polynomial 'division'.



## Practical CRC

- Same 'division' performed at receiver
  - If we get different remainder, 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 CRC.
- Any combination that 'adds' 'multiple' of G(x) not detected.
- 'Multiple' of G(x) means 'product' of G(x) & any other poly.
- Assume we transmit a sequence of bits represented by T(x).
- Effect of errors is to 'add' an error polynomial E(x).
- Instead of T(x) we receive T(x) $\oplus$ E(x)
- Remainder will now be $R_T(x) \oplus R_E(x)$
  where $R_T$ (x) is remainder for T(x) & $R_E$ (x) is for E(x).
- If E(x) is 'multiple' of G(x), $R_E(x)$ =0 & does not change CRC.

17/03/14 Comp61242: B5 25

## Choice of G(x)

- See references on CRC
- In practice, a number of standard ones are in general use.

17/03/14 Comp61242: B5 26

## 'Burst' errors & standard generators

- G(x) of order r causes all error 'bursts' of length $\le$ 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: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

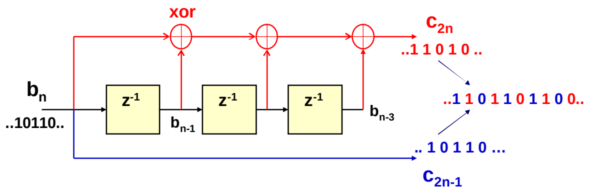
17/03/14 Comp61242: B5 27

## MATLAB code for CRC-8-ATM

```
function check=CRC8(xa);
%  xa is array of bits to be transmitted (column vector)
% Generates 8-bit CRC check with g(x) = x^8 + x^2 +x + 1
      xae = [xa;0;0;0;0;0;0;0;0];  % Append 8 zeros to bit-stream
      g8x = [1;0;0;0;0;0;1;1;1] ;   % Generator polynomial
      xsa=xae(1:9);
      for i=1:length(xa)
          if xsa(1) = = g8x(1), xsa = xor(xsa,g8x);  end;
             xsa(1:8)=xsa(2:9);
             if i<length(xa)  xsa(9)=xae(i+9); end;
      end;
 check = xsa(1:8);            % 8 bit CRC column vector
 return;
```

17/03/14 Comp61242: B5 28

## 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 $z^{-1}$ box is just a 'delay by one bit' box.

17/03/14 Comp61242: B5 29

## Tabulate to check this out

| $b_n$ | $b_{n-1}$ | $b_{n-2}$ | $b_{n-3}$ | $b_n$ | $b_n \oplus b_{n-1} \oplus b_{n-2} \oplus b_{n-3}$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |

Interleave outputs, lower part first:

..1 1 0 1 1 0 1 1 0 0..

17/03/14 Comp61242: B5 30

## Strategy for decoder

- Encoder generates 'valid' sequences:
  - 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.

17/03/14 Comp61242: B5 31

## 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?

17/03/14 Comp61242: B5 32

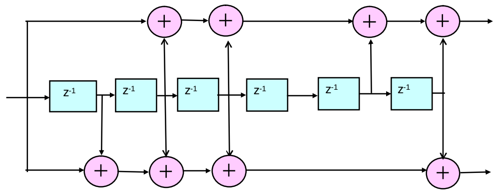
## Some terms

- '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.

17/03/14 Comp61242: B5 33

## A standard half rate convolutional coder



- Described by 2 generator functions '1111001' & '1011011'
  - These 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 )

17/03/14 Comp61242: B5 34

## 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;
```

17/03/14 Comp61242: B5 35

## Advantages of FEC for cellular

- Use of FEC in cellular mobile systems increases energy efficiency & effectiveness of spatial multiplexing
- 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 frequency re-use problem & reduces power consumption.

17/03/14 Comp61242: B5 36

## Minimum free distance

- (171,133) coder is non-systematic as message bits not seen in coded bit-stream.
- Non-systematic coders are generally more powerful.
- 'Minimum free distance' (dfree) for this half rate coder is 10.
- Equivalent of minimum Hamming distance for block codes.
- Minimum Hamming distance between coded bit-streams for any different message sequences of arbitrary length (>K).
- Error bursts of length (dfree-1)/2 bits can always be corrected.
- Errors in 4 coded bits within a short segment will be corrected
- Longer error bursts may be corrected but this is not guaranteed.
- Once these errors have been forgotten, further segments containing 4 bit-errors or less can be corrected.
- Works if segments with errors are not too close together.



## IEEE802.11 frame

- IEEE802.11 'mac protocol data unit' (MPDU) has 256 control bits, the body & a 32 bit 'CRC' check (FCS):

| 16 bits | 16 | 48 | 48 | 48 | 16 | 48 | 16 | variable | 32 |
|---|---|---|---|---|---|---|---|---|---|
| [redacted] | | | | | | | | Body | [redacted] |

MPDU convolutionally (or LDPC) coded & passed down to phy layer.

'SIGNAL' header (with FEC), & preamble for synchronisation prefixed.

Preamble & 'signal' sent at 1(802.11b&g) or 6Mb/s. Coded MPDU up to 72Mb/s.

| PLCP preamble | SIGNAL (½FEC) | Coded MPDU |
|---|---|---|



## Error control in 802.11

- At some bit-rates, 'half rate' FEC coder is used.
  - At others 3/4 rate and 2/3 rate used.
  - Half rate means that FEC doubles number of bits.
- Scrambling & interleaving applied to randomize bit-errors
- Receiver has 'soft-decision' Viterbi FEC decoder.
  - Used for both 'SIGNAL' & 'coded MPDU.
- 'Soft decision' allows 2 dB decrease in SNR with same error rate as hard.
- If too many bit-errors to be corrected, CRC will fail & packet will be rejected.



## Conclusions & learning outcomes

- Roles of error correction & detection in fixed & mobile networks.
- Both are used in IEEE802.11 standard & mobile telephony.
- Hamming distance relevant to error detection & correction.
- Differences between block codes & convolutional (tree) 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.
- Need 'soft decision Viterbi decoder' for max-likelihood decoding.
- Standard IEEE convolutional coder is easily implemented.
- Minimum free distance specifies correction power of conv coders.



## Problems & discussion points

1. Why is a checksum that adds '1's not a good idea?
2. Why do IEE802.11 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. For links without FEC, could you introduce FEC in an application?
5. Can you improve the naïve decoder for the 'rolling parity' coder?
6. How are CRC bits used for proving integrity in WEP security and why they are not really good at this task?
7. Parity check is a form of CRC. What is its generator polynomial?
8. 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?
9. Calculate the polynomial 'sum' of 100111 and 111001.



## Problems & discussion points(cont)

10. Find the remainder when 101000 is polynomal divided by 111.
11. Sketch a (117,155) convolutional coder.
12. Calculate output of the (171,133) coder when the input is 11011
13. What is meant by a 'burst' of bit-errors?