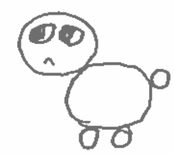
# TURBO CODING

① Consider the task of communicating a message using a communication method that corrupts 5% of the bits you send, e.g.

10101 $\xrightarrow{\text{might end up as}}$ 11101

corrupted



How can you ensure that the recipient will understand what you sent?

② One approach is to send multiple redundant copies of the message:

10101 10101 10101 ⟶ 10100 11101 10101

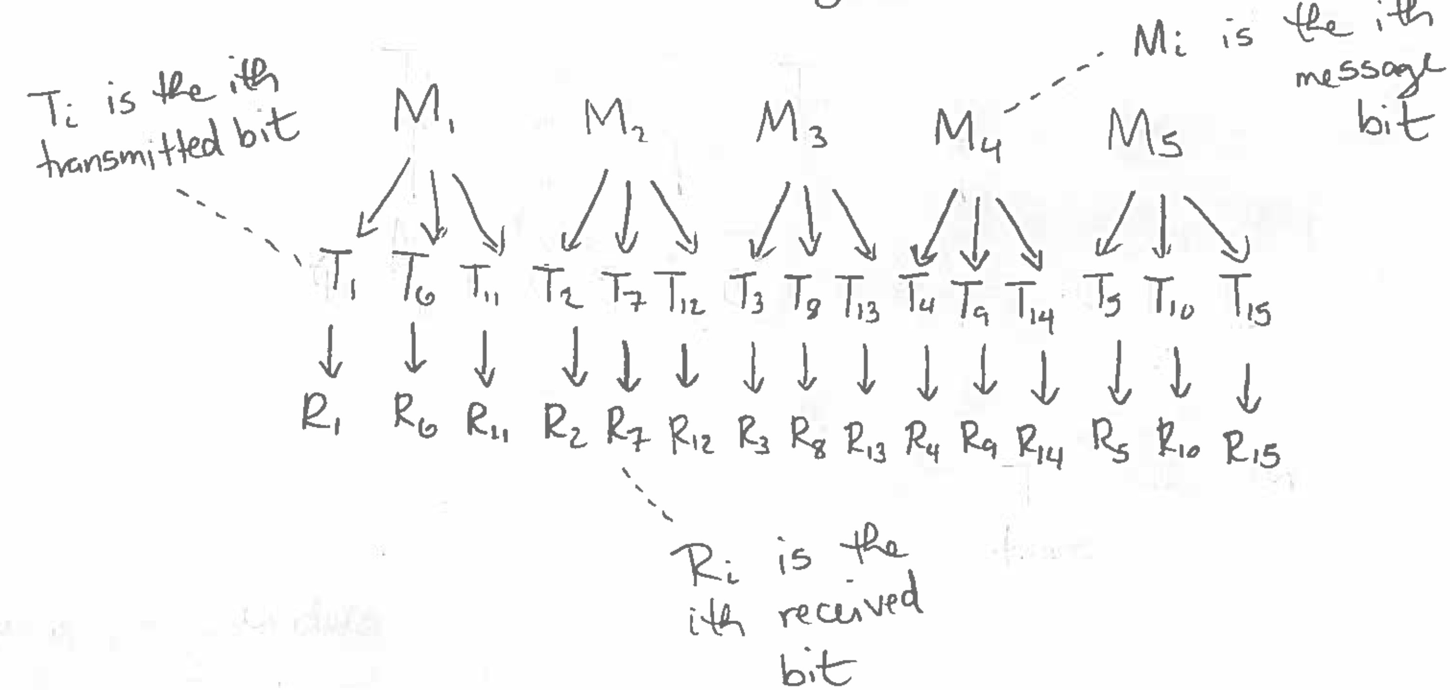
Corrupted

probably 10101 is the true message, since it falls "between" 10100 and 11101

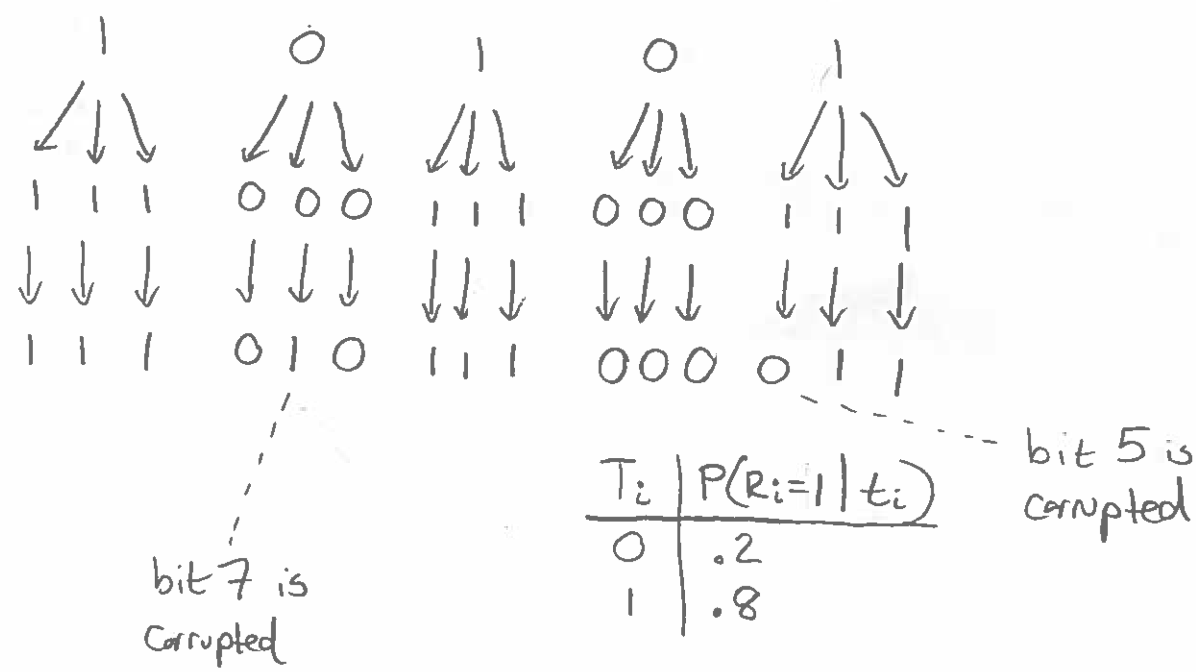
If you send enough redundant copies, then hopefully we can reconstruct the message.

# TURBO CODING

(3) How can we reconstruct the most probable original message? Model it as a Bayesian network!



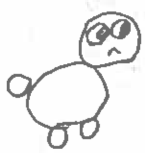

e.g.




| $T_i$ | $P(R_i=1 \mid t_i)$ |
|---|---|
| 0 | .2 |
| 1 | .8 |

4) The recipient observes <u>only</u> the received bits $R_1, \ldots, R_{15}$. To infer the most probable original message, the recipient computes $\underset{m_1, \ldots, m_5}{\operatorname{argmax}} P(m_1, \ldots, m_5 \mid r_1, \ldots, r_{15})$ using the Bayesian network.

5) But this may not be the most efficient way to encode the original message. Suppose we have a 3-bit message and our bandwidth only allows us to send 6 bits total. If we simply send two redundant copies of the message, what if a bit gets corrupted?



$$101 \; 101 \rightsquigarrow \overset{\substack{\text{corrupted}\\ \downarrow}}{1}11 \; 101$$

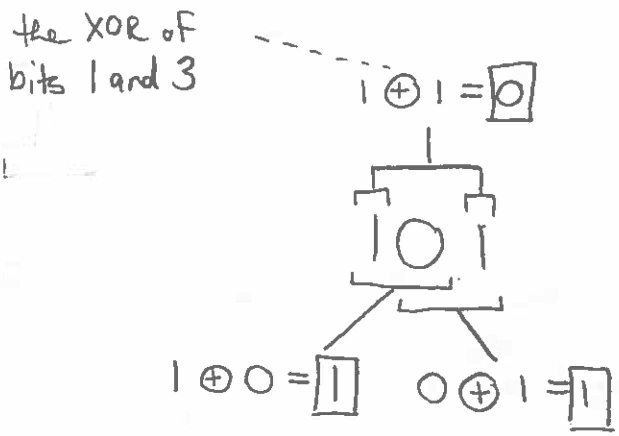
The received bits are equally likely given an alternative message:

$$111 \; 111 \rightsquigarrow 111 \; 1\overset{\substack{\text{corrupted}\\ \downarrow}}{0}1$$

So if the recipient receives the transmission 111 101, there's no way to recover the original message.

# Turbo Coding

⑥ What if we use our extra 3 bits in a more clever way?




Then:

XOR of bits 1 and 2
XOR of bits 2 and 3
XOR of bits 1 and 3

101 110 ⟶ 111 110 (corrupted)

But the message for 111 (with the same corruption) is:

111 000 ⟶ 101 000

Notice that the received bits 111 110 is closer to transmission 101 110 and received bits 101 000 is closer to transmission 111 000.

# TURBO CODING

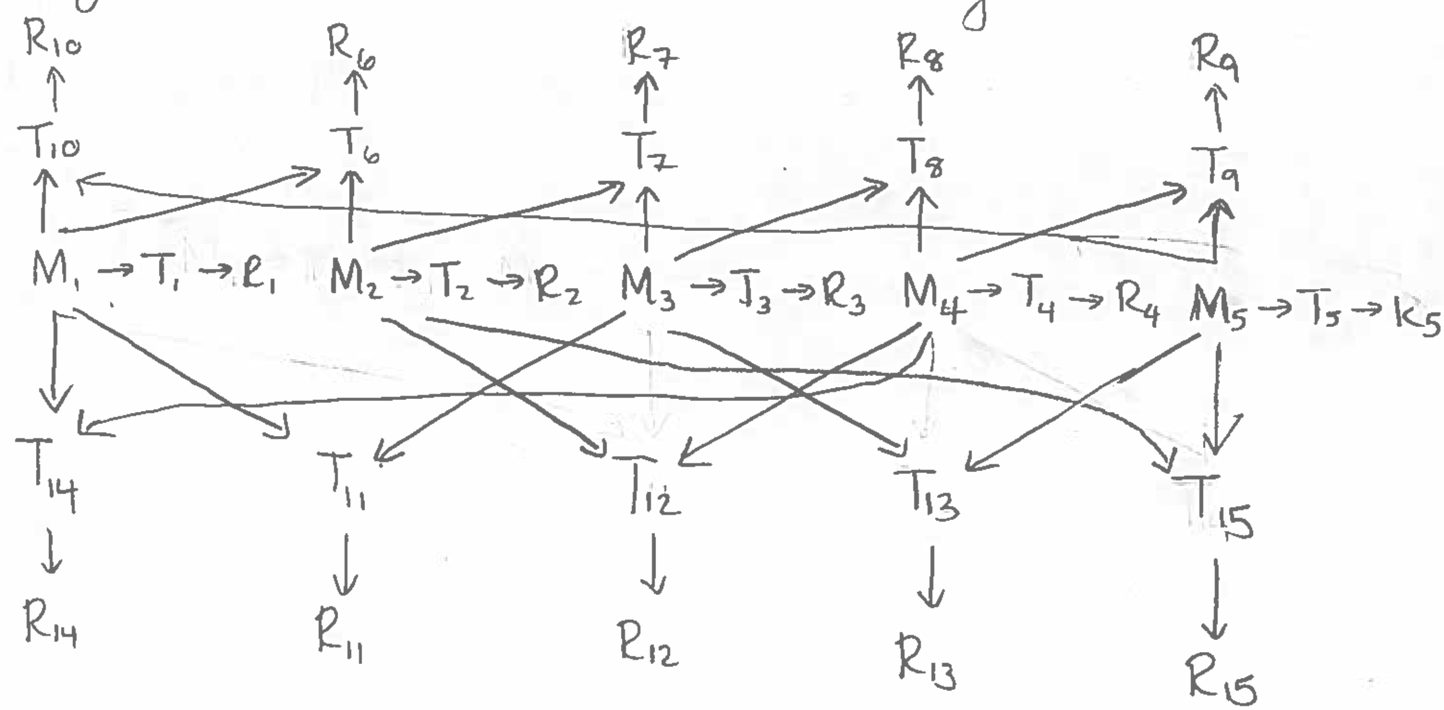
7) For an n-bit message, we can transmit as follows:

- let $M_1, \dots, M_n$ be the original message bits
- let Transmission $T_i = M_i$ for $1 \le i \le n$.
- let transmission $T_{n+i} = M_i \oplus M_{(i+1)\%n}$ for $1 \le i \le n$
- let transmission $T_{2n+i} = M_i \oplus M_{(i+2)\%n}$ for $1 \le i \le n$

e.g. if the original message is 10100, then the transmission is:

10100 11101 01110

8) Again, we can model this as a Bayesian network:




And again, to infer the most probable original message, the recipient computes $\underset{m_1,\dots,m_5}{\operatorname{argmax}} P(m_1,\dots,m_5 \mid r_1,\dots,r_{15})$.

# TURBO CODING

(9) If we code up these Bayesian networks (let's call them RedundantNet and ParityNet), then we see the following statistics:

If the corruption probability is 0.10%:

| | P(orig message \| received bits) in RedundantNet | in ParityNet |
|---|---|---|
| no corrupted bits | .993 | .999 |
| 1 corrupted bit | .895 | .997 |
| 2 corrupted bits | .807 | .981 |

(10) We can make ParityNet arbitrarily reliable by increasing the number of parity bits transmitted.

A very popular architecture that uses these ideas for encoding and decoding is called "Turbo Coding", which is a core technology behind 3G and 4G communications standards.

A seminal paper by Robert McEliece et al, called "Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm" describes the connections between turbo coding and Bayesian network inference.