

Communication Networks: Technology & Protocols



Stavros Tripakis (stavros@eecs)

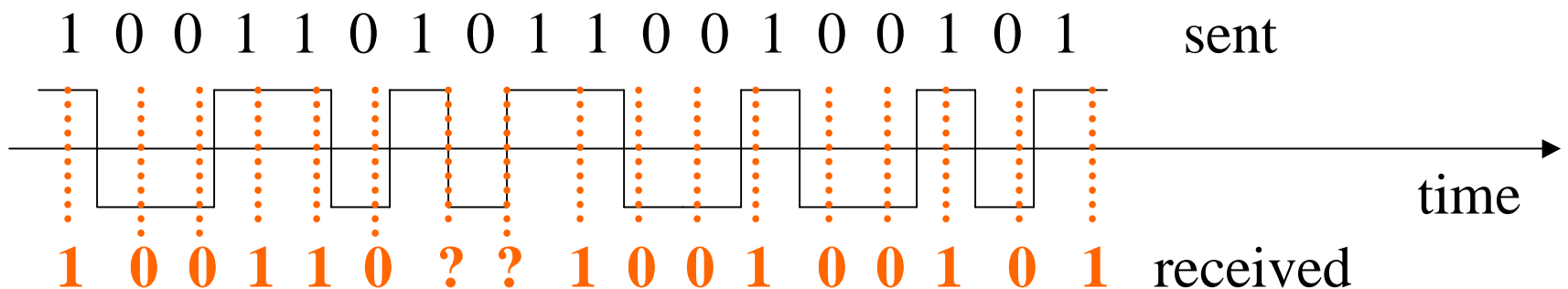
Lecture 24
October 20

Error detection & correction

- Detect (and possibly correct) errors in transmission introduced by physical layer.
- Error detection happens at different layers:
 - At data-link: **bit-error** detection/correction: check whether all bits in a packet are received correctly. If yes, deliver packet to higher layer; otherwise, discard packet (or, if possible, correct it and deliver it).
 - At network or transport layers: **packet-error** detection: check whether packet is correct. If yes, deliver to higher layer, otherwise, discard.

Where are bit-errors and where do they come from ?

- Bit inversion, e.g., send 100, receive 101.
- Lost/added bits, e.g., send 100, receive 1000.
- Causes of errors:
 - Noise: send signal of -5 Volts, receive signal of +1 Volts.
 - De-synchronization of sender/receiver clocks:



- In wireless channels: multipath, shadowing, etc.

Error-detection/correction codes

- **Block codes:** memory-less codes.
 - For each “block” of (data) bits M , add some redundant bits $R = f(M)$, and transmit MR (codeword).
- **Convolutional codes:** codes with memory.
 - $R = f(M, s)$, where s is the state of the code (depends on previous blocks).
- **Criteria for choosing a code:**
 - How much redundancy (overhead) does it add ?
 - How many errors can it detect/correct ?
 - How expensive is it to compute ?

Simple solution: send multiple copies



- Each packet is sent k times, $k > 1$.
- Error detection (not all copies the same).
- Error correction (majority rule).
- Reduces PER: packet error rate.
- Too much redundancy: channel capacity divided by k .
- Can do better using more clever codes.

Parity codes

■ Simple parity:

■ For each k bits, add 1 parity bit:

■ Even/Odd parity: some of 1s must be even/odd.

■ E.g.: 1001 --- even parity ---> 1001**0**

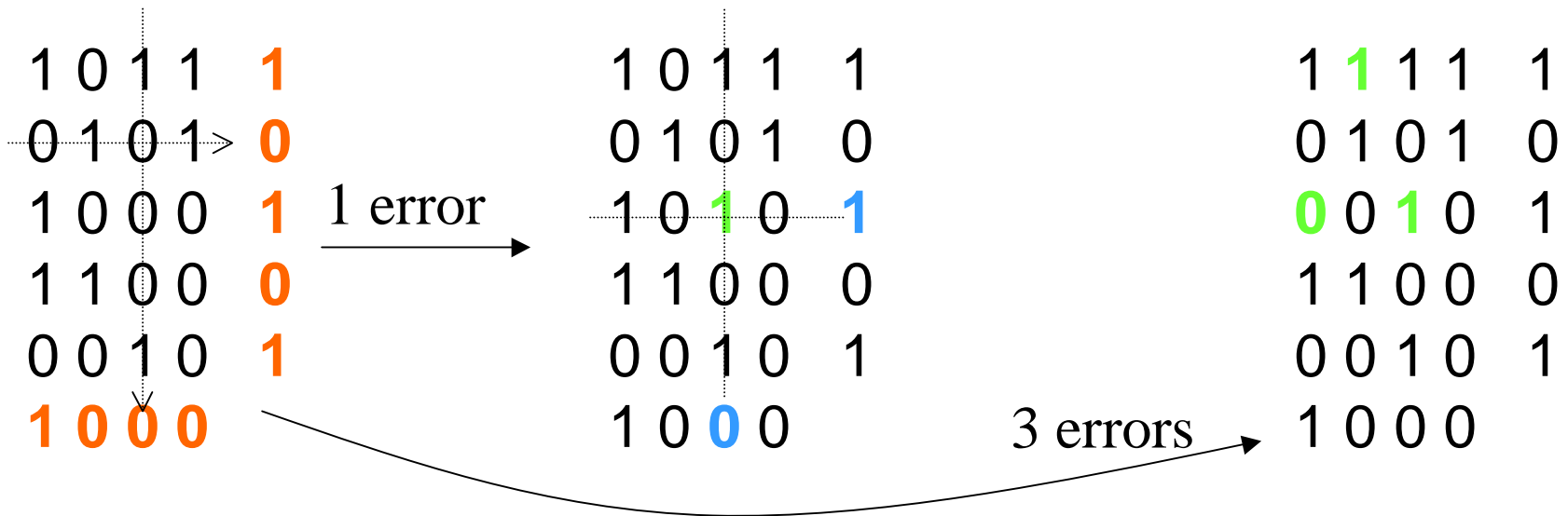
■ Detects an odd number of bit-errors, e.g.:

■ 1001 ---> 10010 --- channel ---> 10**1**10 (1 error)
(receiver knows there is an error).

■ 1001 ---> 10010 --- channel ---> **0**1010 (2 errors)
(receiver detects no error).

Parity codes

■ Two-dimensional parity:



- Detection of 1,2,3-bit errors.
- Correction of 1 bit-error per row/column.

Hamming codes

- Bit-words of length n : points in n -dimensional boolean space.
- *Hamming distance* between two words: the number of bits in which the words differ, e.g. $|1001, 1000| = 1$, $|1001, 1100| = 2$.
- If valid codewords have minimum distance d , then:
 - If less than d bit-errors per block, error detection.
 - If less than $d/2$ bit-errors per block, also correction.

Hamming codes

- What is the minimum amount of redundancy ?
- Hamming: can always correct 1 bit-error per codeword of length n , with $\log_2(n+1)$ redundant bits.
- Idea: if codeword is $b(1) b(2) \dots b(n)$, then:
 - $b(1), b(2), b(4), b(8), \dots, b(2^k)$: parity bits.
 - All other bits : data bits.
 - Parity bit $b(2^i)$ checks the parity of all bits $b(j)$, s.t. j written in base-2 has 1 in position i .
 - With k data bits, it is called an (n,k) code.

Hamming codes: example

- Example: data = 0010. Find the (7,4) codeword.
 - There are $7-4=3$ parity bits.
 - The codeword is: $b_1 b_2 b_3 b_4 b_5 b_6 b_7$, where $b_3=0$, $b_5=0$, $b_6=1$, $b_7=0$.
 - Parity bit b_1 checks b_3 (011), b_5 (101), b_7 (111). Therefore, $b_1 = 0$ (assuming even parity).
 - Parity bit b_2 checks b_3 (011), b_6 (110), b_7 (111). Therefore, $b_2 = 1$.
 - Parity bit b_4 checks b_5 (101), b_6 (110), b_7 (111). Therefore, $b_4 = 1$.
- Final codeword: 0101010.

Hamming codes: example (cont'd)

- Final codeword: 0101010.
- Channel corrupts codeword: 0101**1**10.
- Receiver can tell which bit is inverted:
 - $b_1 = 0$ is the parity bit of 0, **1**, 0 : wrong.
 - $b_2 = 1$ is the parity bit of 0, **1**, 0 : correct.
 - $b_4 = 1$ is the parity bit of 1, **1**, 0 : wrong.
 - The inverted bit is the intersection of the bits checked by b_1 and those checked by b_4 , that is, $\{b_3, b_5, b_7\}$ and $\{b_5, b_6, b_7\} = \{b_5, b_7\}$.
 - The inverted bit is not b_7 , since this is checked by b_2 (so, b_2 would be wrong, but it is correct).

Cyclic redundancy check (CRC)

■ Idea:

- There is a generator G , known to both sender and receiver.
- If M is the data to be sent, sender transmits codeword $(M, M \bmod G)$.
- Receiver gets (X, Y) and checks whether
$$X \bmod G = Y.$$

If yes, OK, otherwise discard packet.
- If G is properly chosen, then if errors occur, it should be improbable that they result in such X and Y that $X \bmod G = Y$.

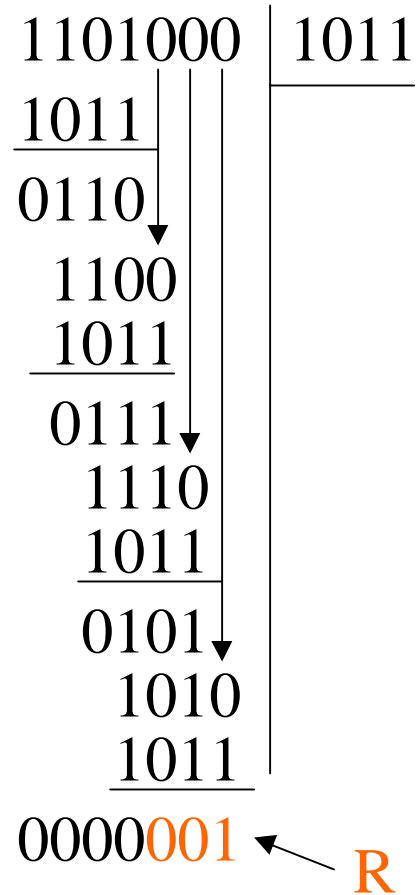
Cyclic redundancy check (CRC)

■ In reality:

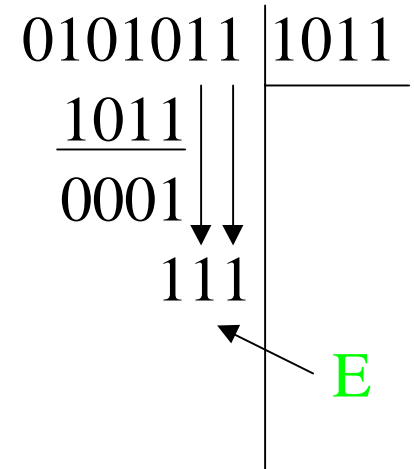
- M, G, etc, are base-2 numbers (bit-strings).
- Operations are done modulo 2 without carry.
E.g., $1+1 = 0+0 = 0$, $1+0 = 0+1 = 1$.
- There are r redundancy bits.
- Sender transmits $T = M \times 2^r + R$, where:
 $R = (M \times 2^r) \bmod G$ (remainder).
- Receiver gets T' (possibly corrupted) and checks whether $E = T' \bmod G = 0$. If yes, it assumes no error.
- In some codes (error-correcting codes or ECC), the 1s in E tell where errors occurred.

CRC : example

- Let $M=1101$, $G=1011$, $r=3$.
- Sender computes R:
long division of 1101000
by 1011.
- Sender transmits
 $1101000+R = 1101001 = T$.
- Suppose channel corrupts
string: $0101011 = T'$.
- Receiver computes
 $E = T' \bmod G = 111$.
- E not 0, so an error
occurred.



sender



receiver

CRC : properties

- CRC based on the theory of finite fields.
- Bit-words can be seen as polynomials:
e.g., 1001 : $x^3 + 1$, $x^4 + x^2 + 2x$: 10110
- It can be shown that CRC can detect :
 - All 1-bit errors.
 - Almost all 2-bit errors, if $G(x)$ has a factor with at least three terms, e.g., $G(x) = (x+1)(x^2+x+1) = x^3 + 2x^2 + 2x + 1$: 1111.
 - Any odd number of errors, if $G(x)$ has a factor $x+1$.
 - All bursts of up to m errors, if $G(x)$ is of degree m .
 - Longer burst errors with probability $1-2^{-m}$, if bursts are randomly distributed.

CRC : properties

- Generators are standardized:
 - CRC-m means a generator polynomial of degree m, that is, a bit-word of length m+1.
 - CRC-8 : 100000111 (or $x^8 + x^2 + x + 1$).
 - CRC-10: 11000001111.
 - CRC-12: 100000001101.
 - CRC-16, CRC-32, CRC-CCITT.
- Ethernet, FDDI use CRC-32, ATM uses CRC-8 and CRC-10 (bit-error rates in fibers are extremely small, e.g., 10^{-14}).
- Implemented in hardware using a shift register of r bits and XOR gates (addition/subtraction modulo 2 w/o carry is XOR).

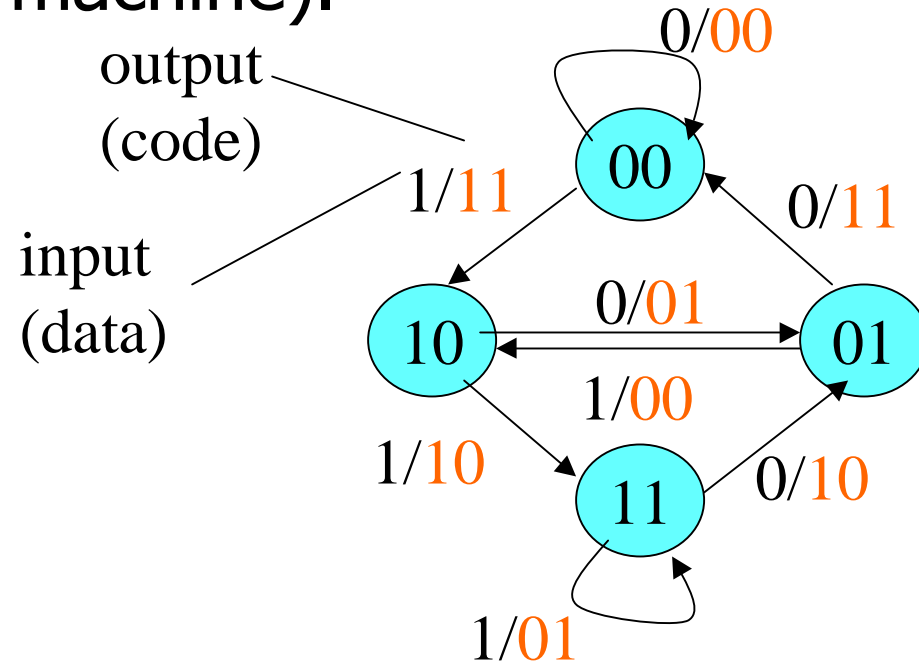
Other block codes

- Permutation codes: aimed at detecting error **bursts**. E.g., *interleaved* Hamming codes:
 - m consecutive codewords (each of length n) are written in an $m \times n$ matrix.
 - Matrix is transmitted column-wise instead of row-wise.
 - A burst of up to m bit-errors will appear as 1-bit error per m -bit column.
- Turbo codes: permutations + multiple encoders/decoders that "cooperate".

Convolutional codes

- Codes with memory (e.g., represented by an FSM, a finite-state machine).
- Encoder:

E.g.: Input = 010111
Output = 001101001001

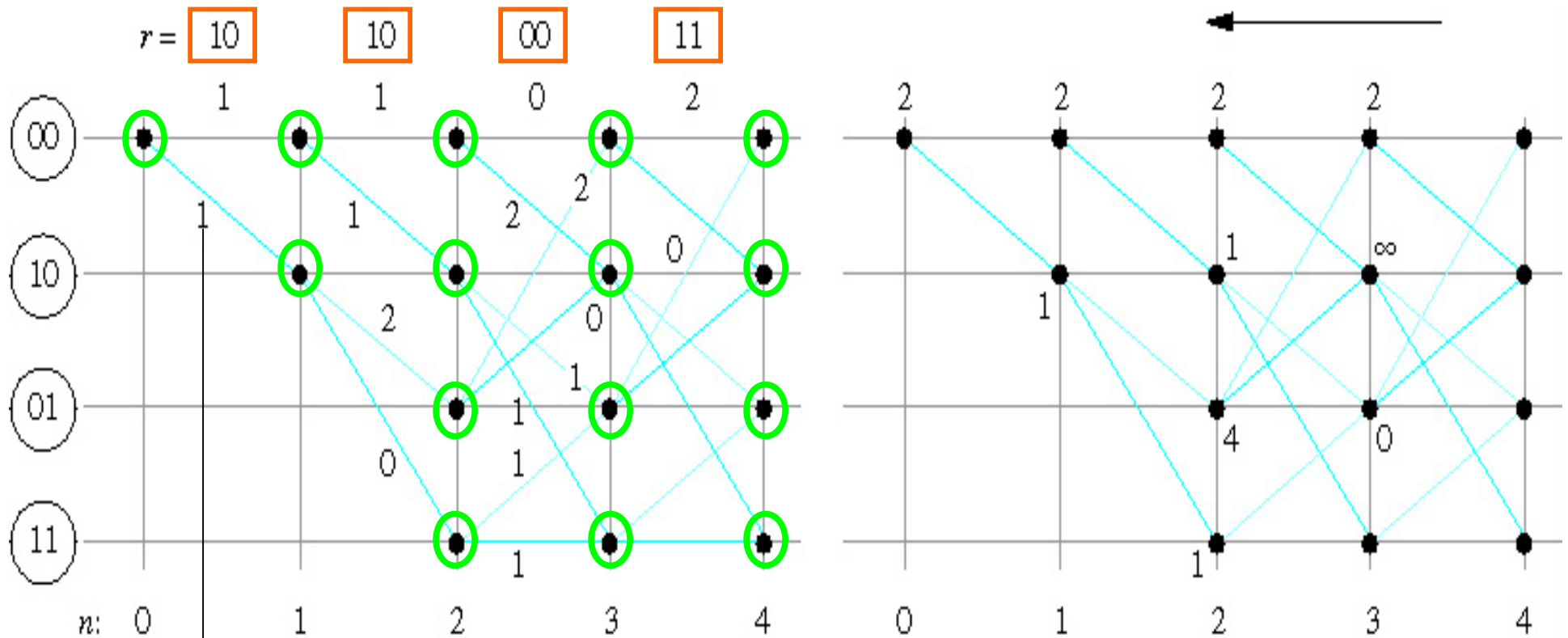


- Code *rate* : 1/2 (2 output bits per each input bit).
- Implemented as a circuit.

Convolutional codes

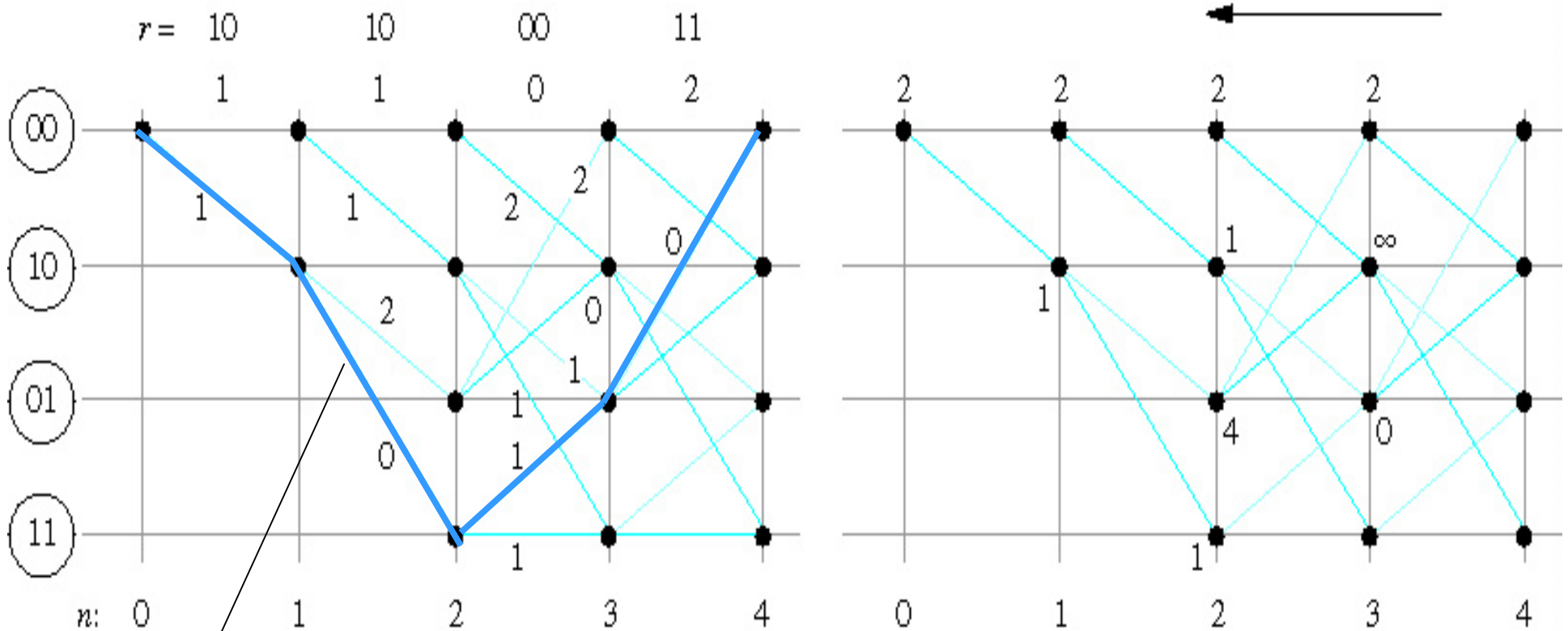
- Decoder:
 - Knows the FSM of the encoder.
 - Keeps track of the possible states the encoder might be in, the likelihood of each state and the history to get to that state.
 - Upon receiving a bit-word, updates its information according to the current states/likelihoods, and the FSM. E.g., it knows that if encoder was at state 00, then it couldn't have sent 01, so an error must have occurred.
 - At the end, it chooses the most-probable history.
 - Implementation: *pruning* is used to remove useless histories and keep complexity low.

Convolutional codes



Penalty : how many bits would have to be corrupted, for this step to be possible.

Convolutional codes



At the end, choose the path with the smallest total penalty.

Internet checksum

- Used in IP, TCP, UDP.
- Idea:
 - Sender adds up data by 32-bit words and puts the result into the checksum field.
 - Receiver adds up data and compares result to the checksum: if different, data (or checksum) must have been corrupted.
- Weaker detection properties.
- Easy to implement in software.
- Adequate, since almost all errors caught already at data link layer.