HARQ Protocol for Burst Transmission over FSO Turbulence Channels

Le Doan Hoang
Computer Communications Lab., The University of Aizu, Japan

Project Confirmation Seminar

Mar. 13, 2019

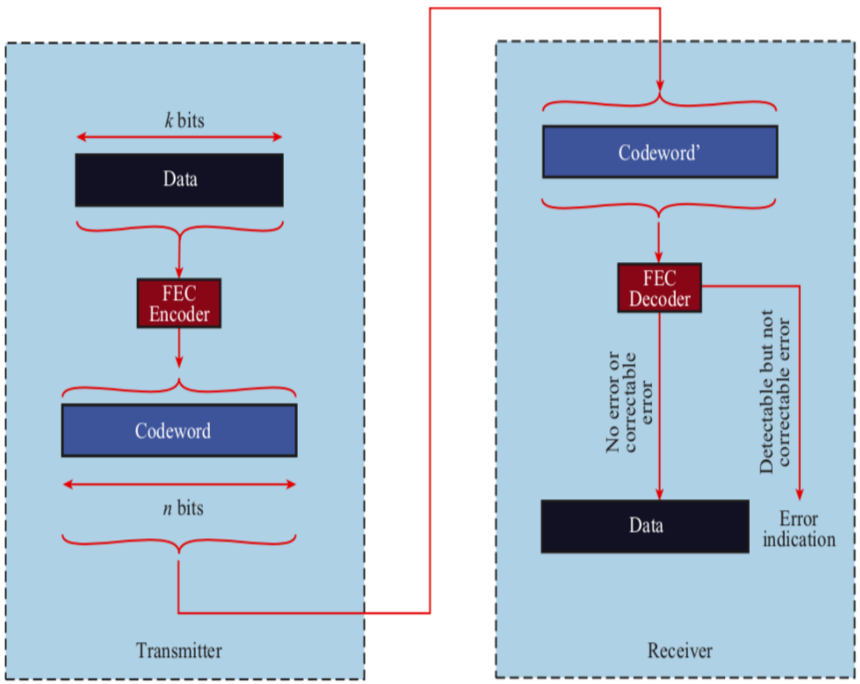
Outline

- Part I: Convolutional Codes
- Part 2: My study

Part I: Review of FEC Code

- FEC is used to correct transmission errors over an unreliable and noisy communication channel without asking for retransmission

- Idea:
  - Transmitter encodes data by adding some redundant bits
  - Receiver, using redundant bits, can correct errors



- There are two main types of FEC
  - Block code: systematic code
  - Convolutional code: non-systematic code

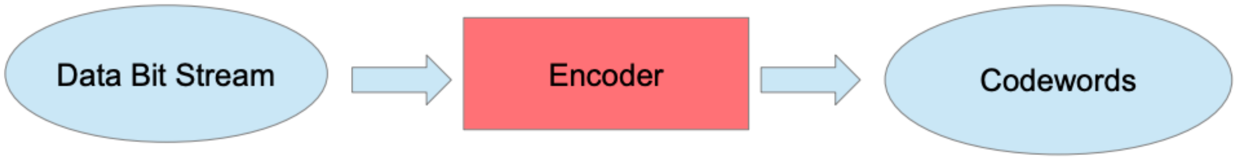
Convolutional Codes (CC)

- For applications which require a continuous stream of bits (e.g. Digital video Broadcasting-Terrestrial), the use of block codes may not convenient.

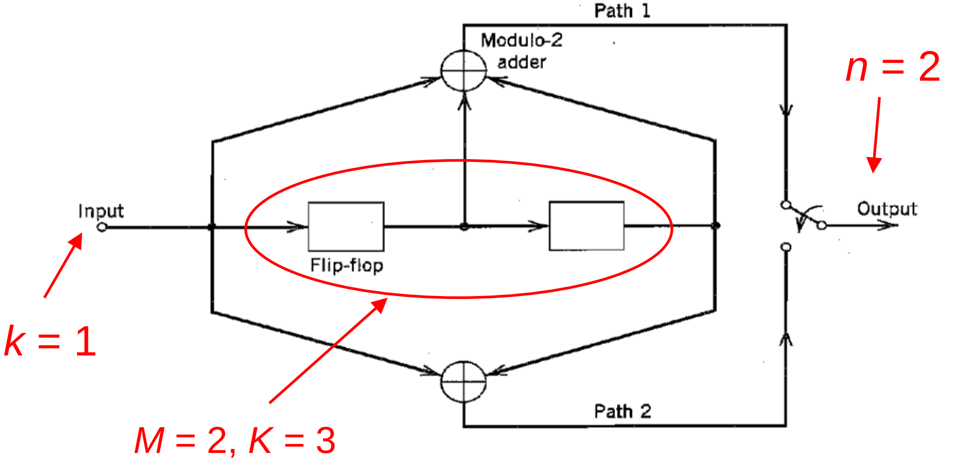
- The convolutional codes, that generate redundant bits continuously so that error checking and correcting are carried out continuously, are used for those applications.

- Features:
  - generates redundant bits by using *modulo-2 convolutions* (name of code)
  - has memory: output bits depend on not only the current input bits but also the previous bits
  - Non-systematic code: cannot distinguish the message bits and redundant bits

Encoder of CC

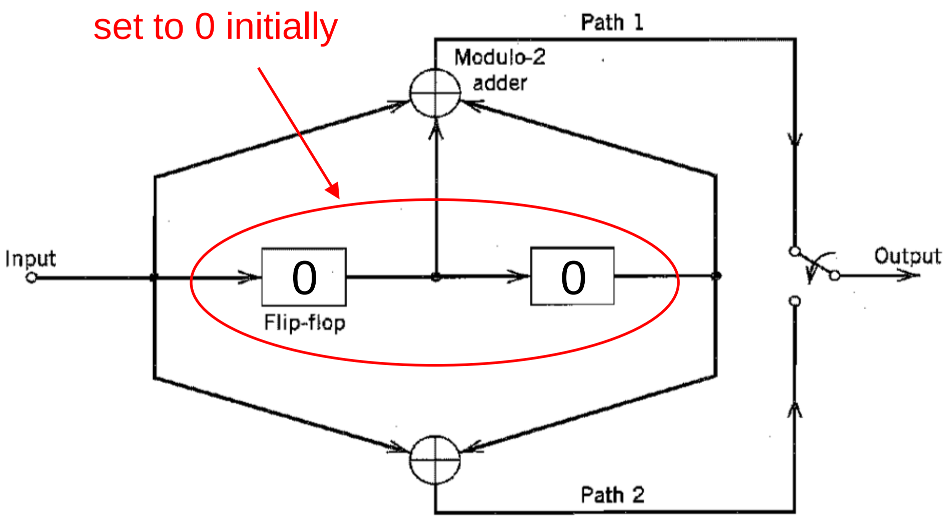


- A convolutional code (*n*, *k*, *K*)
  - *k*: no. of message bits shifted into the encoder at a time (k = 1 is usually used)
  - *n*: no. of encoder output bits corresponding to the k message input bits
  - *K*: constraint length; no. of shifts over which a single message bit can influence the encoder output ($K = M + 1$); *M*: shift registers
  - Coding rate: $R_c = k/n$
- E.g. CC (2, 1, 3)



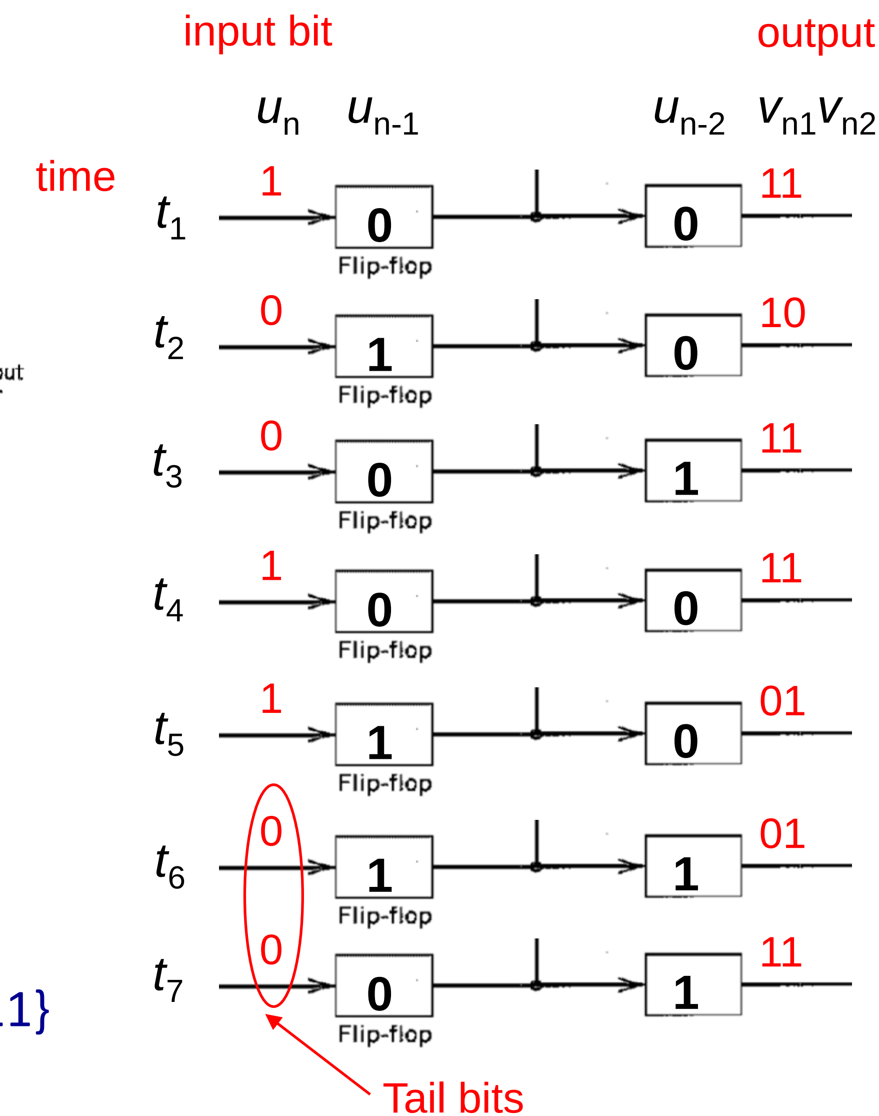
Example of CC

- $(n, k, K) = (2, 1, 3)$



$v_{n1} = u_n \oplus u_{n-1} \oplus u_{n-2}$
$v_{n2} = u_n \oplus u_{n-2}$

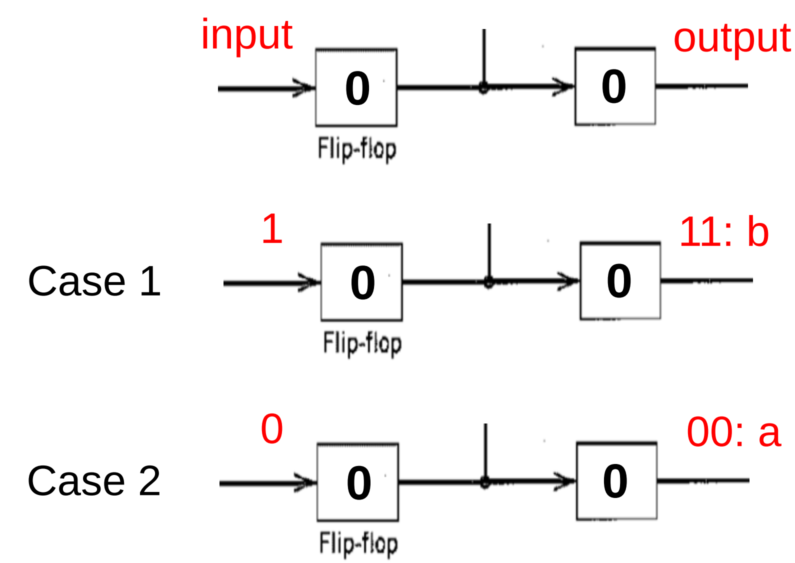
- Input: m = 10011
- Output: c = {11, 10, 11, 11, 01, 01, 11}

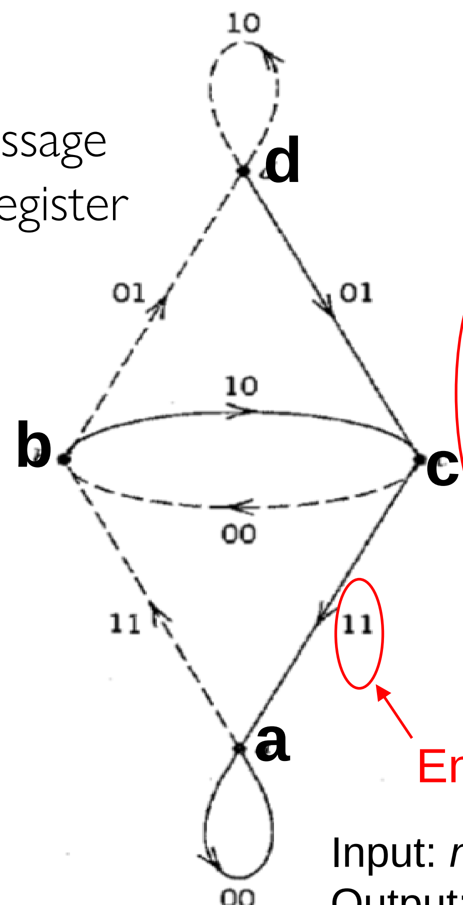


Representation of CC (1)

- The structure properties of a convolutional encoder can be illustrated in graphical form such as (1) state diagram (2) trellis

- **(1) State diagram**
  - States are defined as (K-1) message bits stored in encoder's shift register
  - E.g.: current state a: 00





——— Input bit = 0

-------- Input bit = 1

State	*Binary Description*
a	00
b	10
c	01
d	11

No. of states: 2^{K-1}

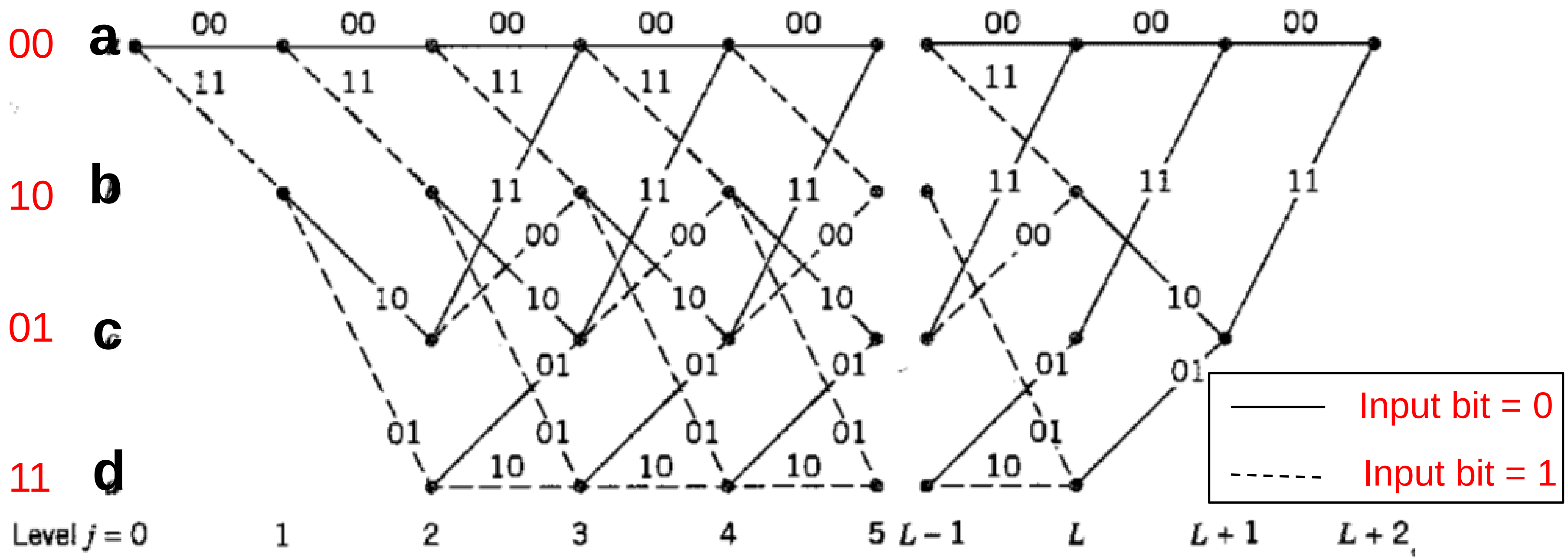
Encoder output bits

Input: m = 10011

Output: c = {11, 10, 11, 11, 01, 01, 11}

Representation of CC (2)

- **(2) Trellis**: extension of state diagram according to time



- The trellis contains ($L + K$) levels, where L is the length of incoming message
- Example: m = 10011 → c = {11, 10, 11, 11, 01, 01, 11}

Decoder of CC

- How to get the correct message at destination?

- There are two kinds of algorithm to encode the CC codes
  - Maximum likelihood algorithm
  - Viterbi algorithm

Maximum likelihood (ML) decoding (1)

- Given the received sequence r, the decoder is required to make an estimate $\widehat{m}$ of m (note: $\widehat{m}$ = m if and only if $\hat{c} = c^{(m)}$).
- The decoding rule is the selection of the estimate $\hat{c}$ so that the probability of decoding error (P_e) is minimized.
- P_e is minimized if the estimate $\hat{c}$ is chosen to maximize the likelihood function, $p(r \mid c)$

$$p(r|\hat{c}) = \max_{\text{over all c}} p(r|c)$$

where c is one of the possible transmitted sequences

Maximum likelihood (ML) decoding (2)

- For a binary symmetric channel, both *c* and *r* represent binary sequences of length *N*, we have : $p(r|c) = \prod_{i=1}^{N} p(r_i|c_i)$

where r_i and c_i are the *i*-th elements of r and c

- The log-likelihood: $p(r|c) = \sum_{i=1}^{N} p(r_i|c_i)$; where $p(r_i|c_i) = \begin{cases} p, & \text{if } r_i \neq c_i \\ 1-p, & \text{if } r_i = c_i \end{cases}$

- Suppose that *r* differs from *c* in exactly *d* positions; *d* is the *Hamming distance*, then we may re-write the log-likelihood,

$$\log p(r|c) = d\log p + (N-d)\log(1-p) = d\log\left(\frac{p}{1-p}\right) + N\log(1-p)$$

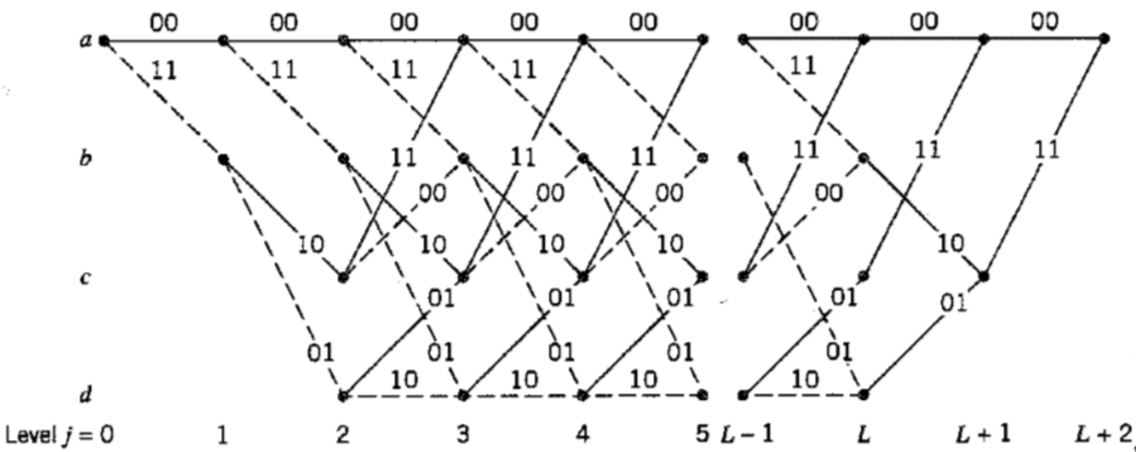
- In summary, the maximum-likelihood decoding rule for binary symmetric channel as follows,

Choose the estimate $\hat{c}$ that minimizes the Hamming distance between the received sequence *r* and the transmitted sequence *c*

ML decoding: Example

- E.g., m = 101 → c = {11 10 00 10 11}; p = 0.1;

received sequence r = {11 11 00 10 11}. Find $\hat{m}$?



path	Code sequence	Hamming distance
00000	00 00 00 00 00	7
00100	00 00 11 10 11	6
01000	00 11 10 11 00	6
01100	00 11 01 01 11	5
10000	11 10 11 00 00	6
10100	11 10 00 10 11	1
11000	11 01 01 11 00	5
11100	11 01 10 01 11	4

- ML decoding is too complex to search all available paths (in case of very long input message bits)
  - End-to-end calculation

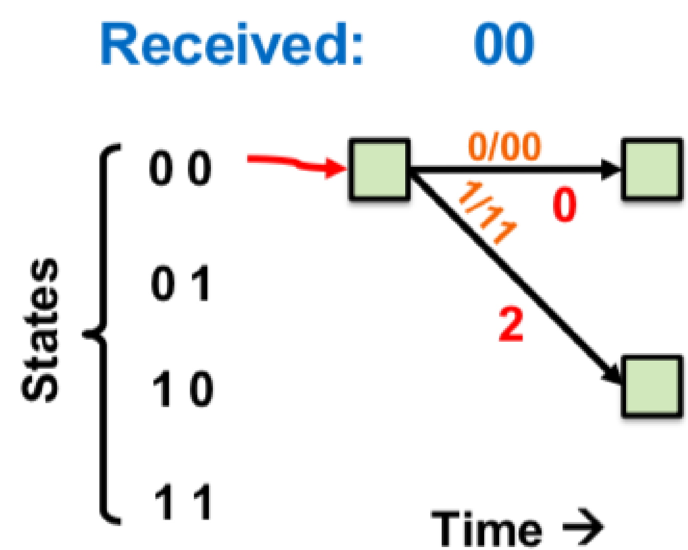
Viterbi algorithm performs ML by reducing its complexity

Viterbi Algorithm

- Viterbi reduces decoding complexity by removing the trellis paths that could not possibly be candidates for ML choice (early rejection)
- Origin of Viterbi Decoding
  - Andrew J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, Volume IT-13, pp. 260-269, April 1967.
  - Viterbi is a founder of Qualcomm.
- There are two kinds of Viterbi Decoding
  - Hard-decision Viterbi Algorithm
  - Soft-decision Viterbi Algorithm

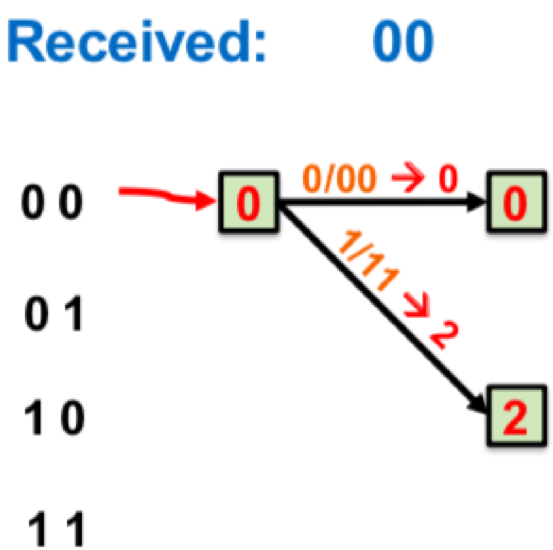
Hard-decision: branch metric

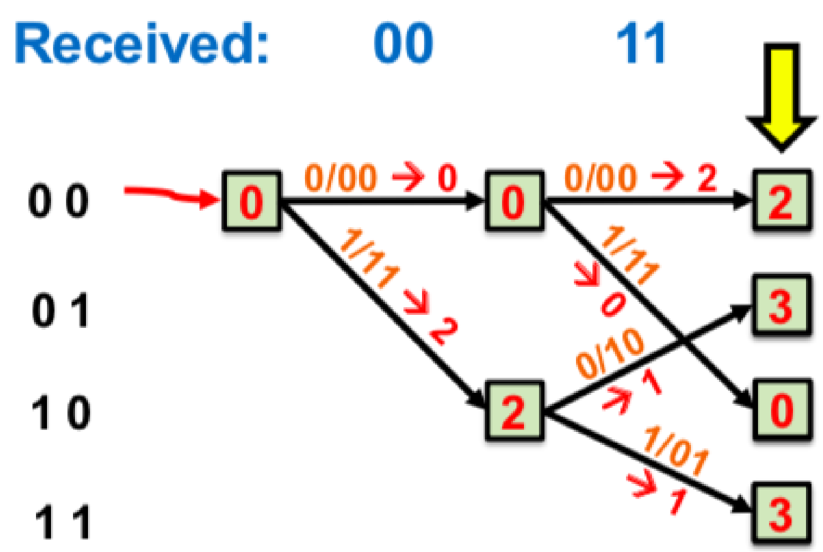
- Branch metric = Hamming distance between received and transmitted bits
- Encoder is initially in state 00, receive bits: 00



Hard-decision: path metric

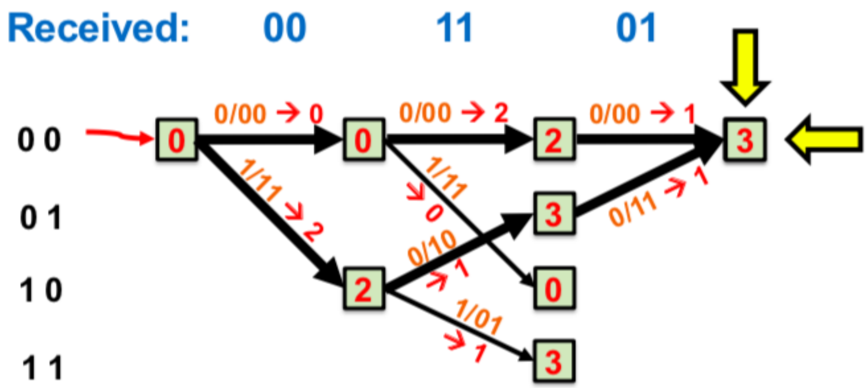
- Path metric = path metric of predecessor + branch metric
- Note: path metric for the left-most state of the trellis is 0



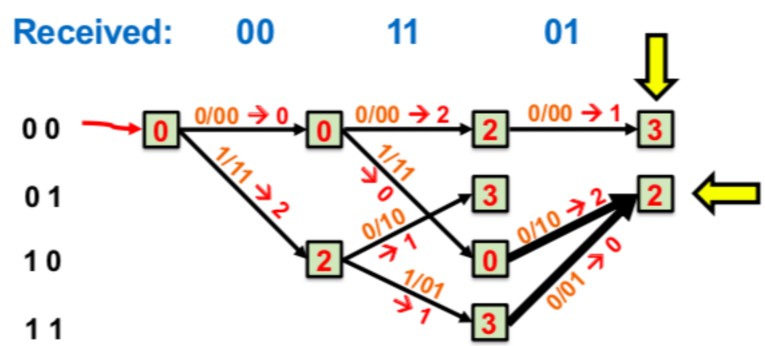


Hard-decision: early rejection

- Problem: each state has two predecessors (or two branches enter a node)

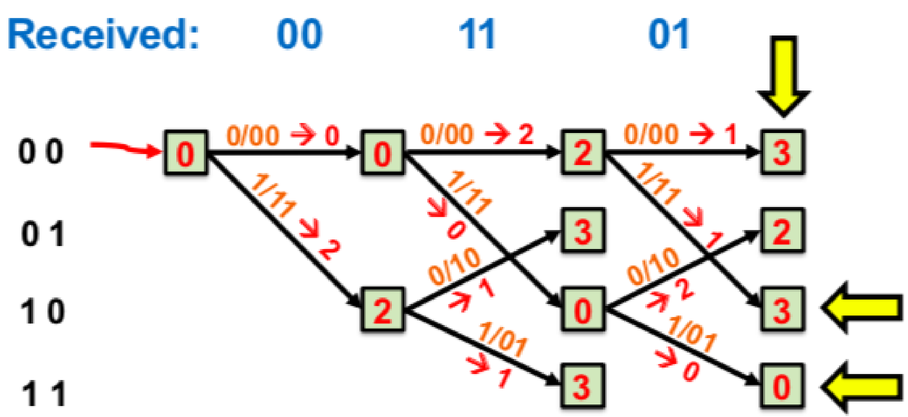


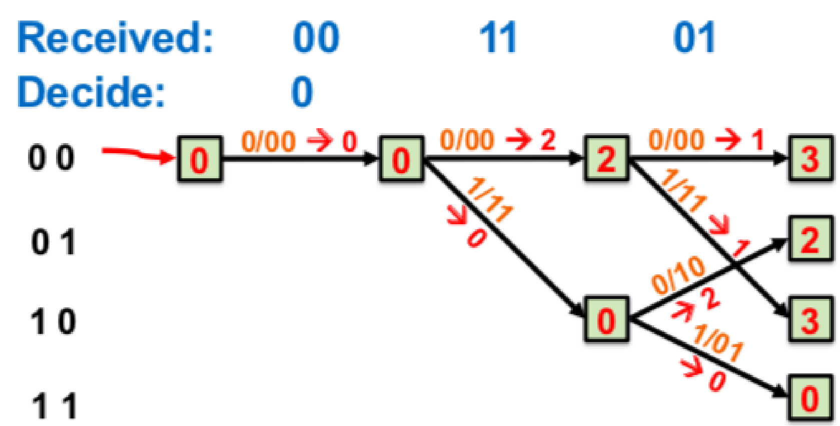
- The algorithm compares two path metrics corresponding to two predecessors. The path with lower metric is retained, and the other path is discarded



Hard-decision: survivor path

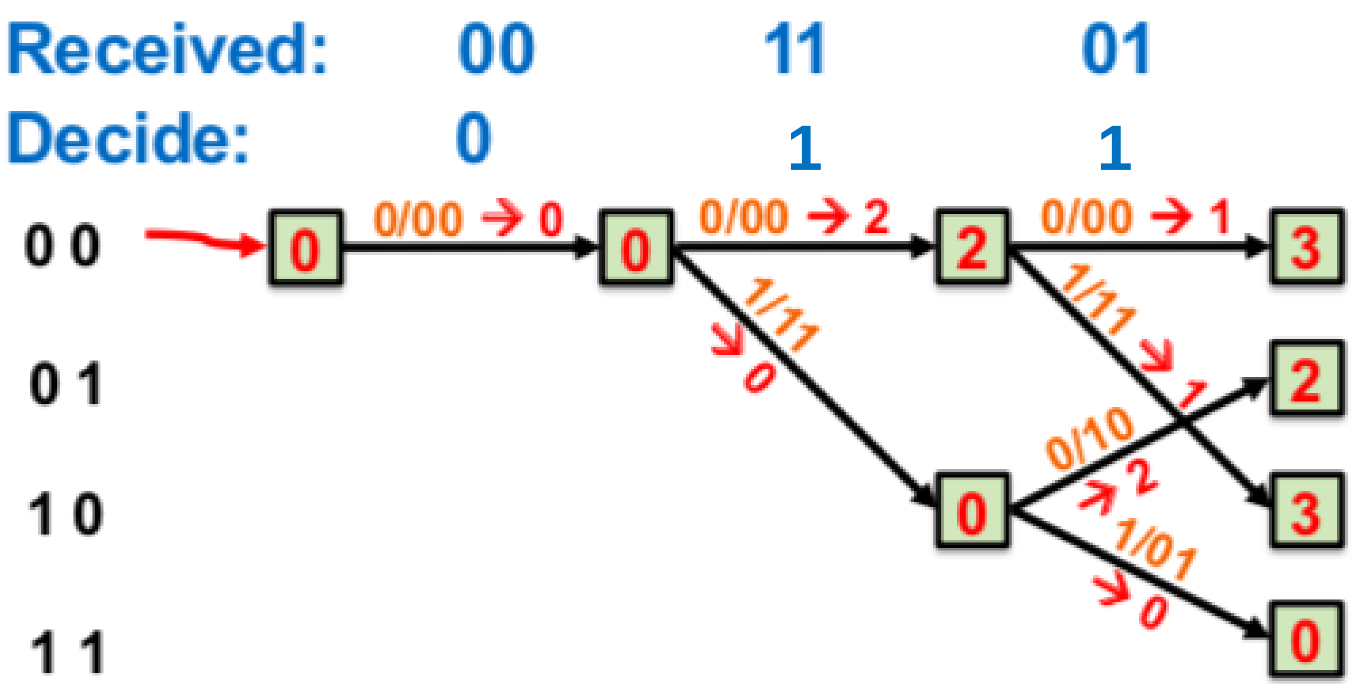
- The paths that are retained by the algorithm are called survivor
- Some branches are not the part of any survivor: remove them





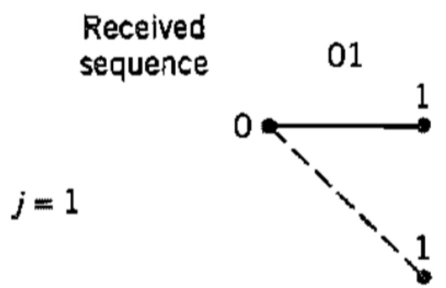
Hard-decision: estimate $\hat{m}$

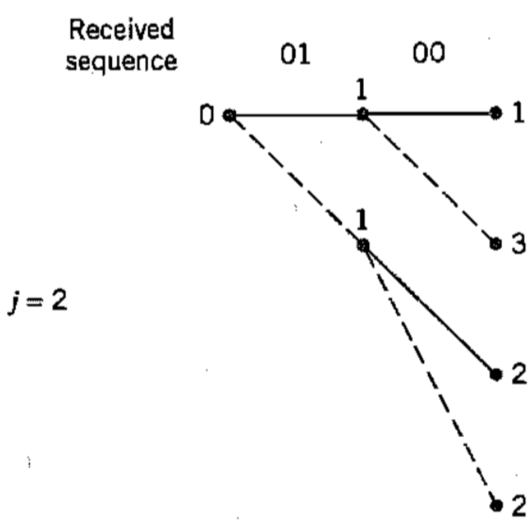
- Choose the survivor path with lowest metric
- Estimate $\hat{m}$ = 0 1 1

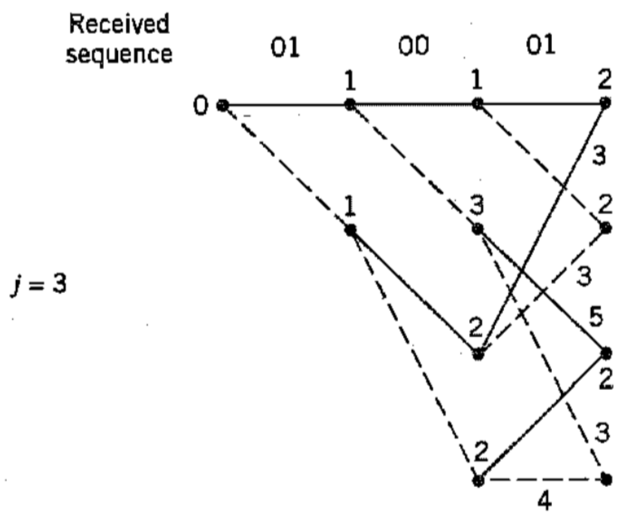


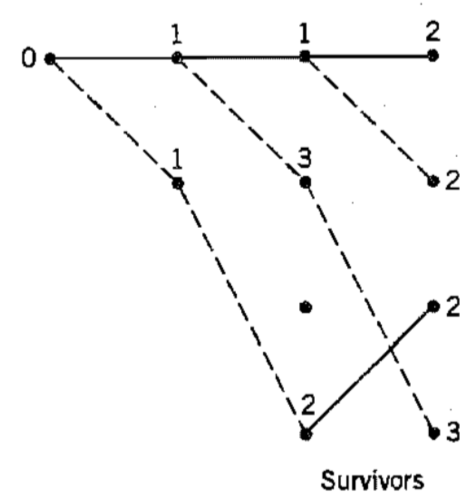
Hard-decision: Example (1)

- E.g., transmitted codeword: c = {00, 00, 00, 00, 00} and received sequence: r = {01, 00, 01, 00, 00}

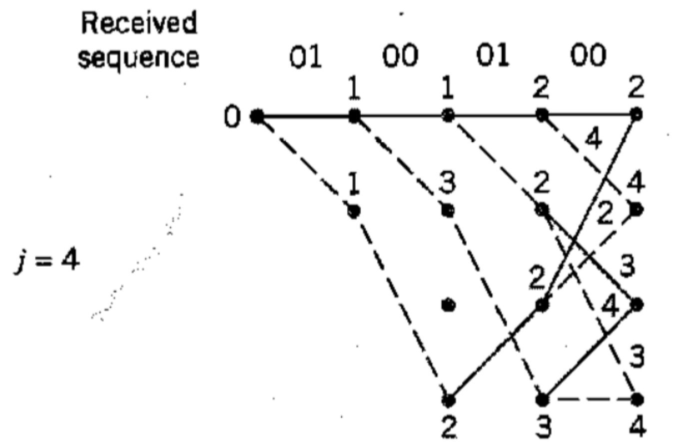


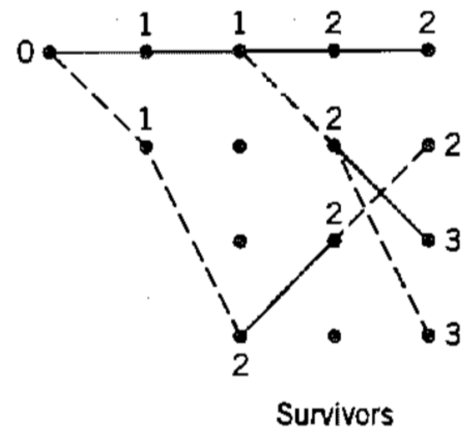






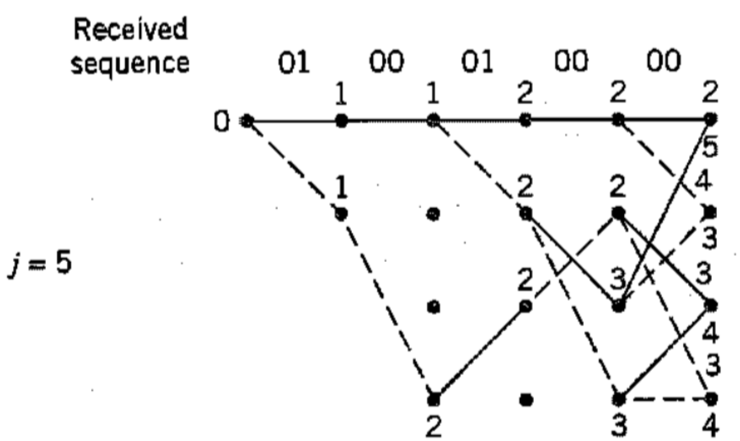
Hard-decision: Example (2)

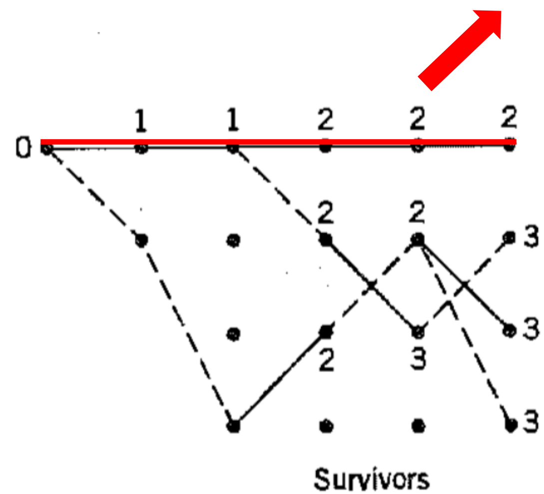




Choose the path with lowest metric

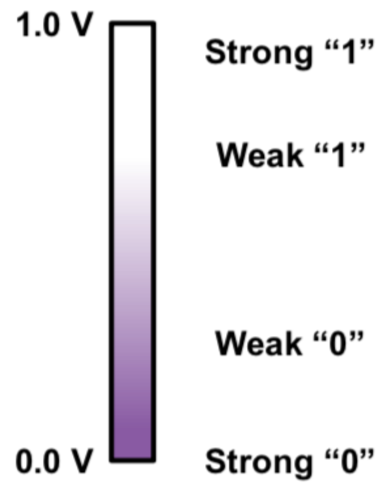
$\hat{c}$ = {00, 00, 00, 00, 00}





Viterbi Decoding: Soft-decision (1)

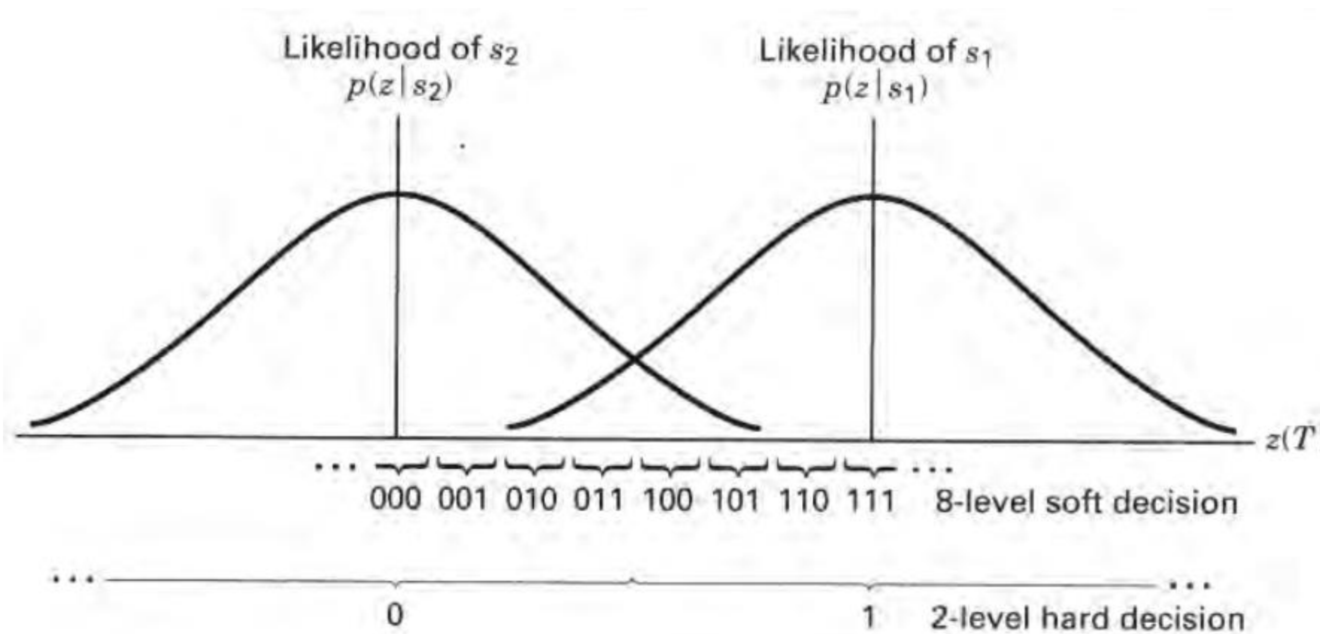
- Coded bits are actually continuously-valued "voltage" between 0V and 1 V



- Hard-decision decoding digitize each voltage to "0" and "1" by comparison against threshold voltage
  - Lose information about how "good" the bit is
    - Strong "1" (0.99V) treated equally to weak "1" (0.51V) with threshold of 0.5V

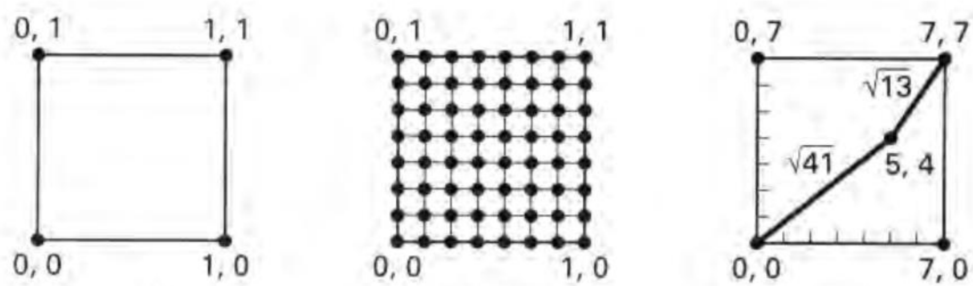
Viterbi Decoding: Soft-decision (2)

- Soft-decision requires a stream of "soft bits" where we get not only the 1 or 0 decision but also an indication of how certain we are
  - E.g. 000 (definitely 0); 001 (probably 0); 010 (maybe 0); 011 (guess 0); 100 (guess 1); 101 (maybe 1); 110 (probably 1); 111 (definitely 1)
  - We call the last two bits are the "confidence" bits

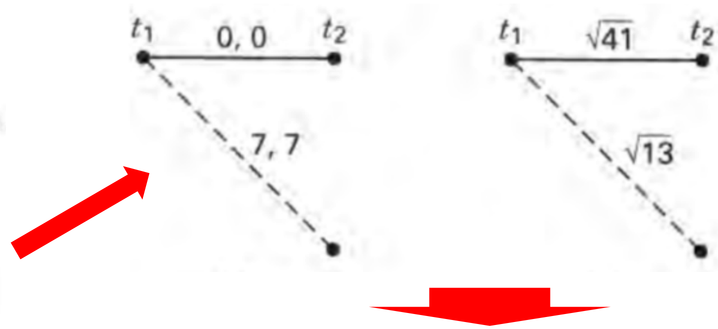


Viterbi Decoding: Soft-decision (3)

- For a rate 1/2, the demodulator delivers two code symbols at a time to the decoder
  - For hard-decision (2-level), each pair of received codes can be depicted on a plane (Fig. a)
  - For 8-level soft decision, each pair of symbols can be represented on an spaced 8 level by 8 level plane (Fig. b)
- Soft-decision branch metric: using Euclidean distance (Hamming distance metric cannot use because of its limited resolution)
- E.g. fig. c, a pair of noisy code-symbol values is the point (5,4). What's Euclidean distance?



(a) (b) (c)



- Same path metric computation
- Same Viterbi algorithm

Error Correcting Capability (1)

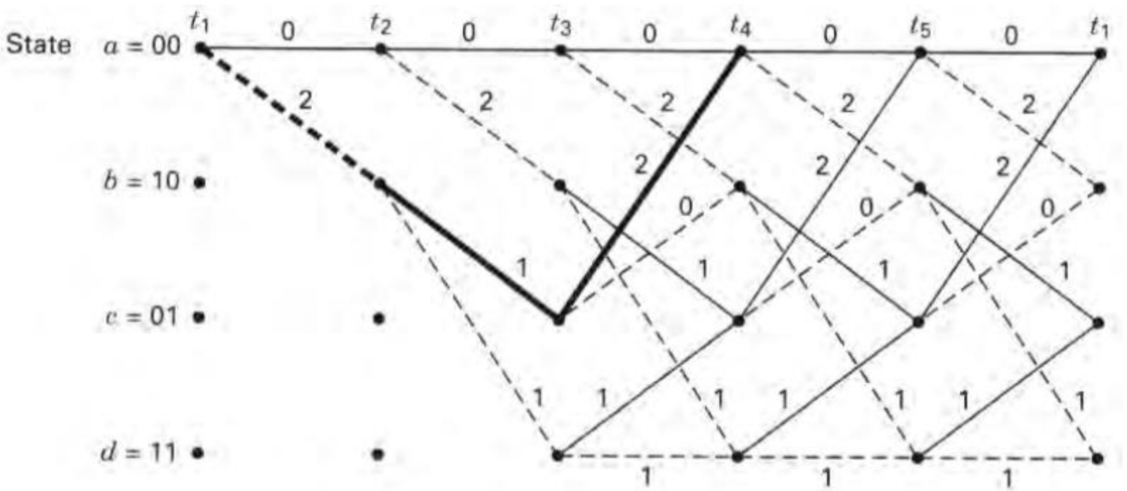
- How many bit errors can be corrected?

- Using the free distance d_{free} to calculate the error-correcting capability of the code
  - Free distance = minimum Hamming distance between each of possible codeword sequences and all-zeros sequence
  - A convolutional code with d_{free} can correct t errors if and only if d_{free} is greater than $2t$.

- E.g., Trellis with $K = 3$

Find the path with smallest non-zeros path metric

d_{free}= 5: we can correct 2 errors



Error Correcting Capability (2)

- The value of d_{free} depends on the constraint length *K*.

Constraint length (K)	Free distance (d_{free})
2	3
3	5
4	6
5	7
6	8
7	10

Source: A. J. Viterbi and J. K. Omura, Priciples of Digital Communication and Coding, McGraw-Hill Book Company, New York, 1979, p. 251

Performance of CC

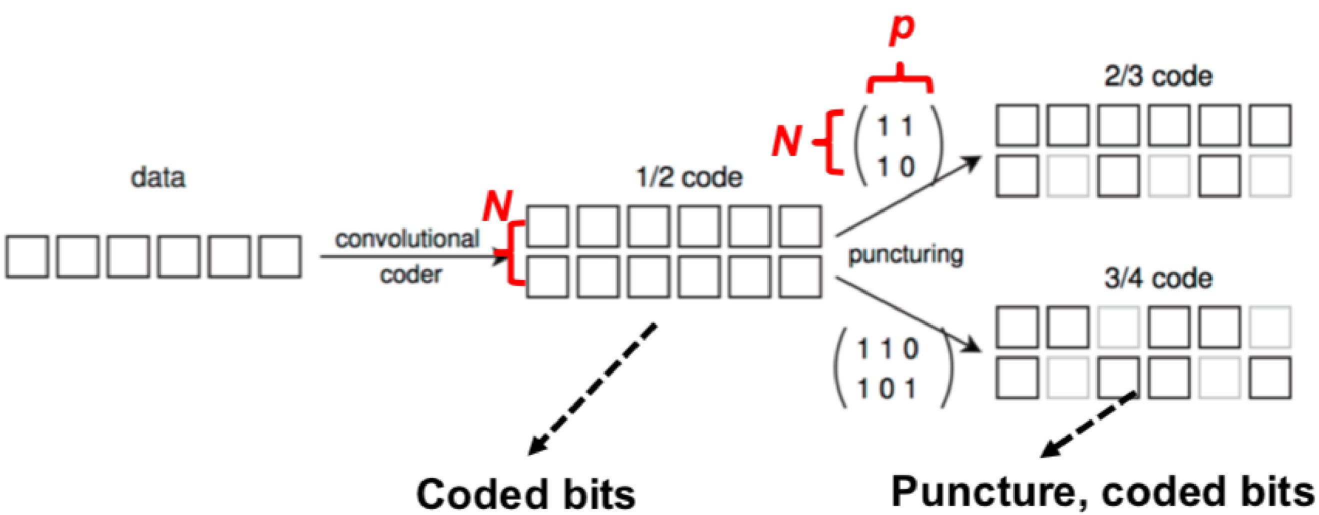
- Performance of CC depends on the coding rate and the constraint length
  - Longer constraint length *K*
    - More powerful code
    - More coding gain
    - More complex decoder
    - More decoding delay
  - Smaller coding rate R_c = k/n
    - More powerful code due to extra redundancy
    - Less bandwidth efficiency

Changing code rate: puncturing

- How to change coding rate?
- E.g. we have a coding rate R_c = 1/2; how to change it into a higher coding rate of 2/3. There are two ways
  - Reconstruct the encoder by using an input and output multiplexer: hardware
  - Use puncturing technique: software → more convenient
- Idea: delete some bits in the original low-rate coded bits
- Decoding: same Viterbi algorithm (decrease error correction capability)

Punctured Convolutional Code

- Using puncturing table (a N x p matrix) to indicate which bits to include
  - Contains p columns and N rows; p is puncturing period
  - If 1, the corresponding code bit is a part of punctured code
  - If 0, delete the corresponding code bit
- The total number of 1's in the matrix is $p + L$; with $L = 1,2,\ldots, (N-1)L$
- For p input information bits, there are $p + L$ output coded bits. Thus, the rate of the punctured convolutional code is $p/(p + L)$



Punctured Convolutional Code: Example (1)

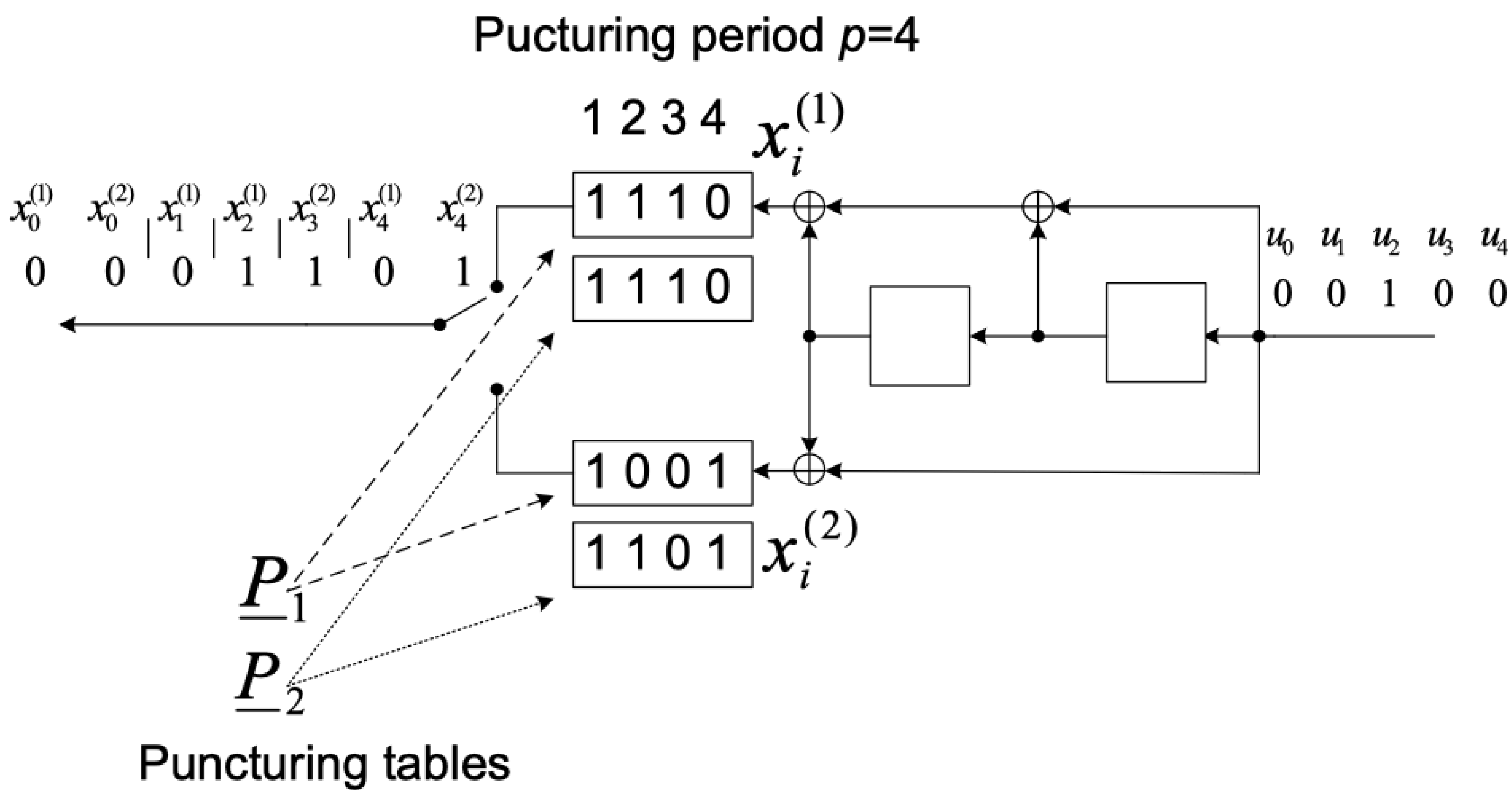
- E.g., message bits m = {0, 0, 1, 0, 0}, coded bits c = {00, 00, 11, 01, 11}, coding rate $R_0 = 1/2$
- c is punctured using two different puncturing tables (matrices) with the puncturing period is $p = 4$

$$\underline{P}_1 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \quad \underline{P}_2 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

- Using P_1, 3 out of 4 code bits of the mother code are used, the others are discarded, i.e., c = {00, 0x, 1x, x1, 11} = {00, 0, 1, 1, 11}
  - Coding rate, R = 4/5
- Using P_2, 2 out of 4 code bits of the mother code are used, the others are discarded, i.e., c = {00, 00, 1x, x1, 11} = {00, 00, 1, 1, 11}
  - Coding rate, R = 4/6 = 2/3

Punctured Convolutional Code: Example (2)

- Encoder of a rate 1/2 code is punctured to a rate 4/5 (top puncturing table) or a rate 2/3 code (bottom puncturing table)

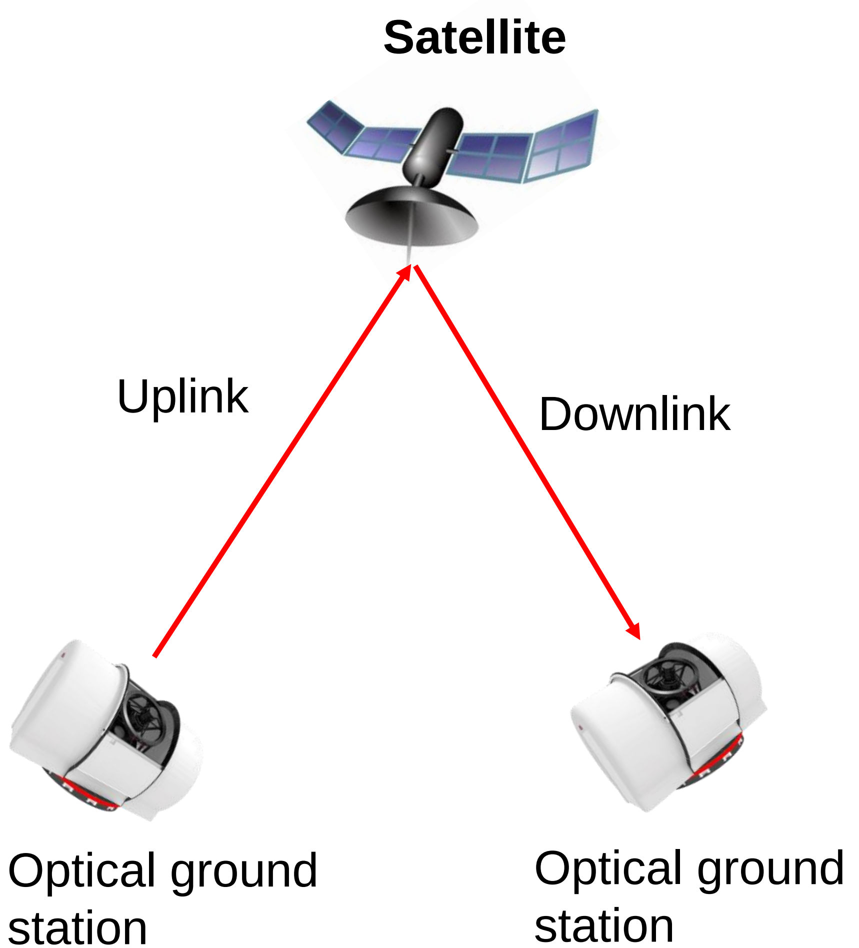


Rate-compatible punctured CC

- How to design convolutional code in adaptive systems (with variable-rate coding)?
  - Puncturing technique is used for change code rate
  - Using rate-compatible restriction: all code bits of higher rate punctured code of the family (from a mother code) are used by the lower rate codes

- This way guarantees smooth transition between different code rates in the systems using adaptive FEC codes
- Rate-compatible punctured convolutional (RCPC) codes
  - if higher rate codes are not sufficiently powerful to decode channel errors, only supplemental bits which were previously punctured have to be transmitted in order to upgrade the code

Part 2: FSO-based Satellite Systems



- Satellite systems widely use in:
  - Navigation
  - Broadcasting
  - Disaster recovery
- Classification: LEO (between 160 – 2000 km), MEO (between LEO and GEO), and GEO (~36,000 km)
- FSO-based satellite to provide high-speed connections (~Gbps)
- Challenge considered in my research: atmospheric turbulence (its effect is up to 40 km above the sea level)

➡ Proper error control methods are needed

Solutions (1): Error Control Methods

- In FSO-domain, there are two popular error control methods: ARQ protocols and FEC codes
- ARQ: retransmission
  - When the channel error rate is high: not efficient due to the increased frequency of retransmissions.
  - In satellite systems: delay is the important issue due to retransmissions.
    - Terrestrial (2 km) ~ 6.67 μs , LEO satellite(2000 km) ~ 6.67 ms
- FEC code: add redundancy to correct errors
  - When the channel less noisy: decrease the throughput due to adding redundancy
  - If the errors are uncorrectable by FEC: lose the reliability

Solutions (2): Hybrid FEC-ARQ (HARQ)

- HARQ: hybrid between FEC and ARQ
- FEC: try to correct the errors first in order to reduce the frequency of retransmissions.
- ARQ: is used for retransmission if the errors are uncorrectable by FEC.

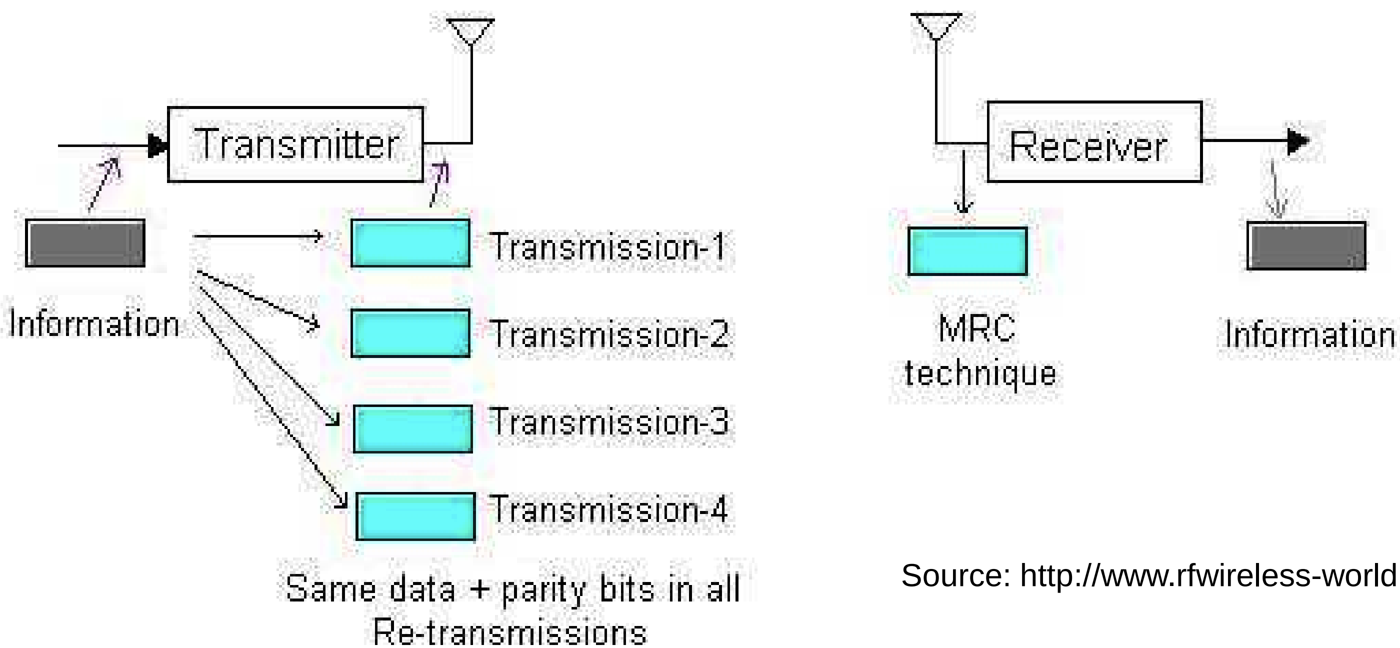
- In this way, it is possible to achieve a higher reliability than a FEC alone and lower delay than ARQ alone

Review of HARQ Protocol

- Type I-HARQ: always discards corrupted frames while they still contains some useful information → not efficient
- Type II-HARQ: is an advanced form of HARQ which uses the concept of frame combining
- Frame combining: the corrupted frames will be stored in the receiver's buffer to be combined with other retransmissions to enhance the correction performance
- Type II-HARQ can be classified into 2 types: chase combining (CC) and incremental redundancy (IR)

Chase Combining

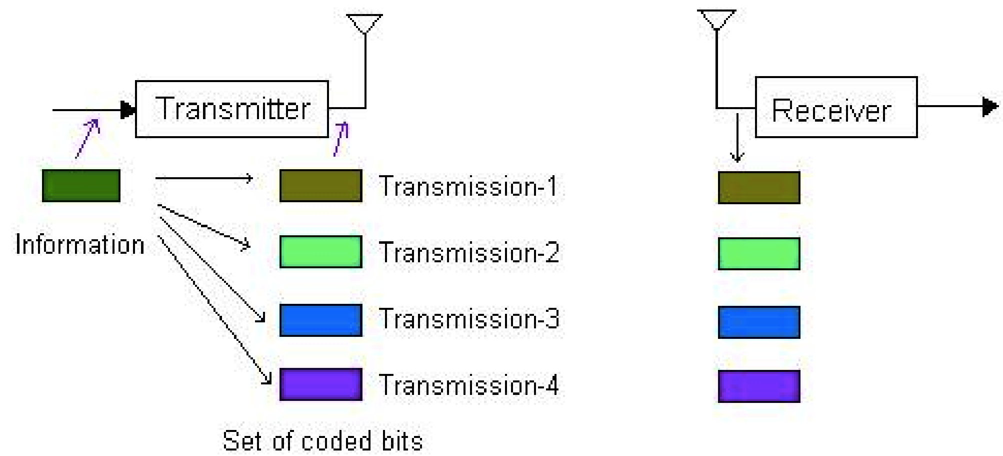
- CC-HARQ: same frames including FEC code are retransmitted each time and retransmitted frames will be combined to obtain the correctable information thanks to Maximum-ratio combining (MRC) technique

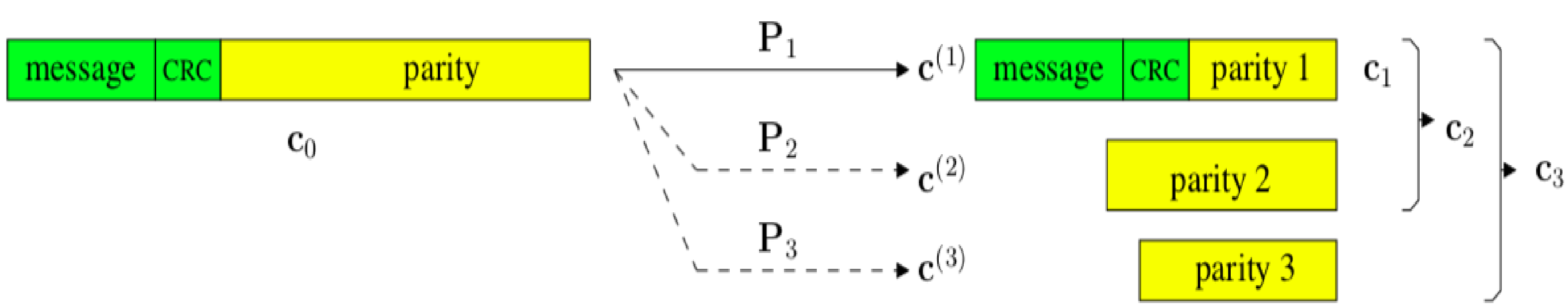


Source: http://www.rfwireless-world.com

Incremental Redundancy (IR)

- IR-HARQ: effective code rate is gradually lowered until received frame is decoded correctly → more efficient





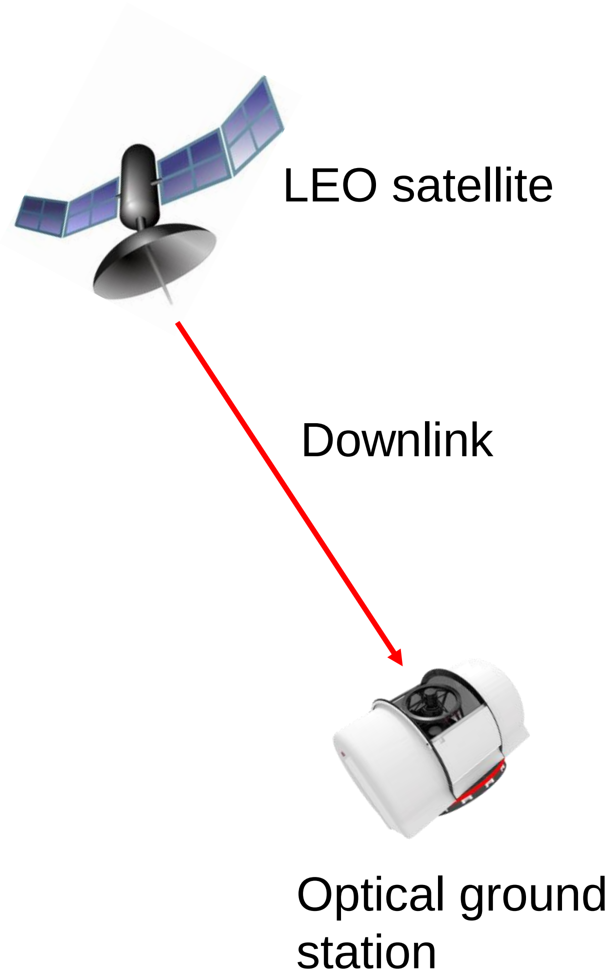
Literature Survey of HARQ

- In FSO domain (IEEE-Journal)

Reference	Main contribution	Type of FEC	Type of ARQ
[1] - 2011	Type II (CC) HARQ over Log-normal	Block code	Stop-and-wait
[2] - 2012	Type I and II HARQ over Gamma-gamma	Block code	Stop-and-wait
[3] - 2014	Type I and II HARQ in FSO with pointing error	Block code	Stop-and-wait
[4] - 2016	Type II HARQ in RF-FSO	Block code	Stop-and-wait
[5] - 2017	Type II RF-FSO multi-hop with HARQ	Block code	Stop-and-wait

- Problems:
  - Most of them considered the employ of HARQ in PHY layer point of view.
  - Hardware implementation in high-speed connection (~Gigabit) at PHY: big challenge, showed in [6] → should be employed at the link layer (faster).
  - The stop-and-wait ARQ in FSO: not efficient, demonstrated in my previous works → should be replaced by sliding window ARQ
  - AMC should be employed to improve system performance

My study

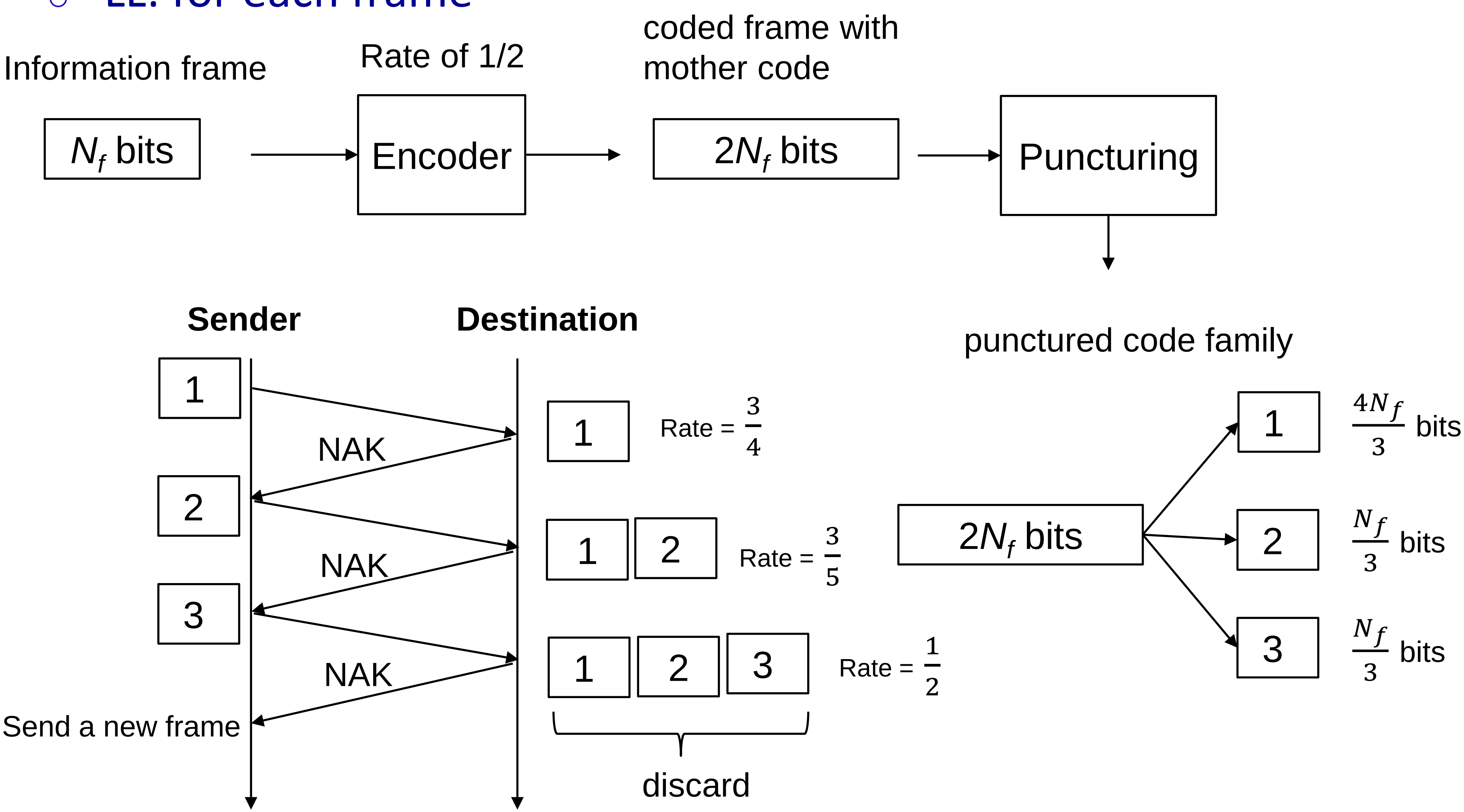


- **Link layer: truncated IR-HARQ**
  - FEC: rate-compatible punctured convolutional codes
  - ARQ: sliding window ARQ
- **PHY layer:**
  - Adaptive Modulation and Coding (AMC)
  - Burst transmission with adaptive number of frames
  - Channel model (source: NICT experiment): Gamma-gamma is the best fitting model for downlink in LEO satellite systems

2017. JOCN. Received-Power Fluctuation Analysis for LEO Satellite-to-Ground Laser Links

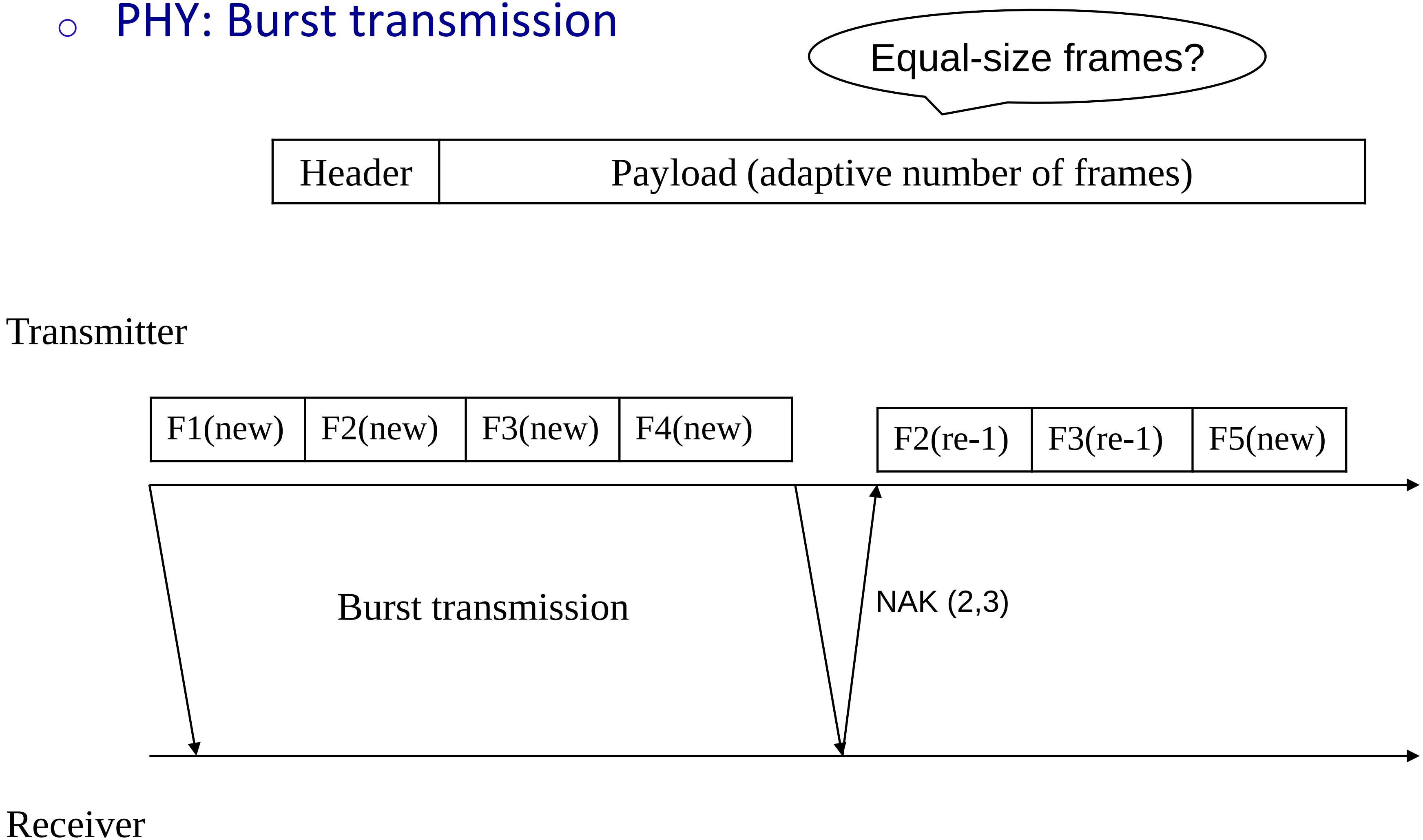
How it works? (1)

- LL: for each frame



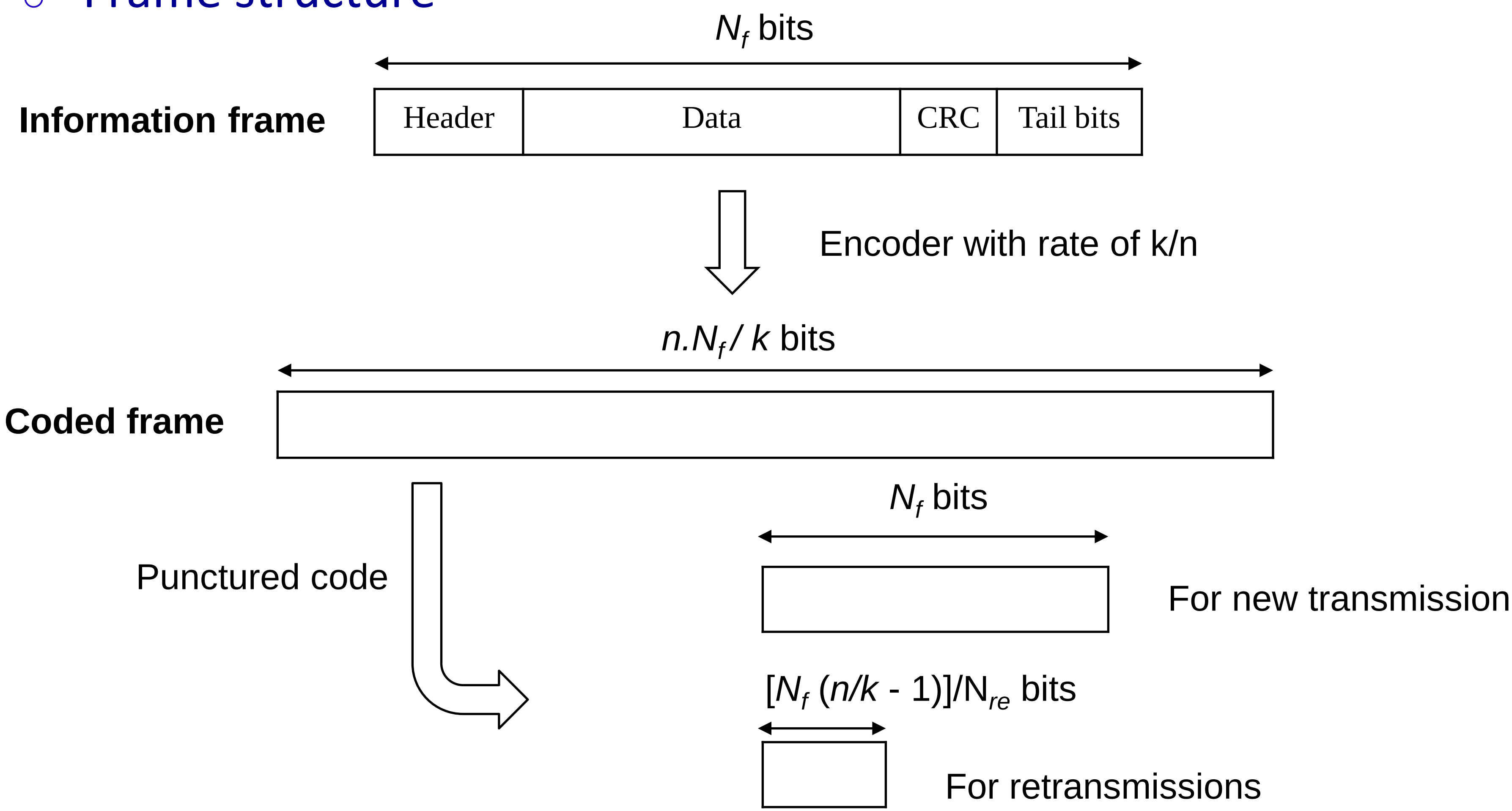
How it works? (2)

- PHY: Burst transmission



Frame Design

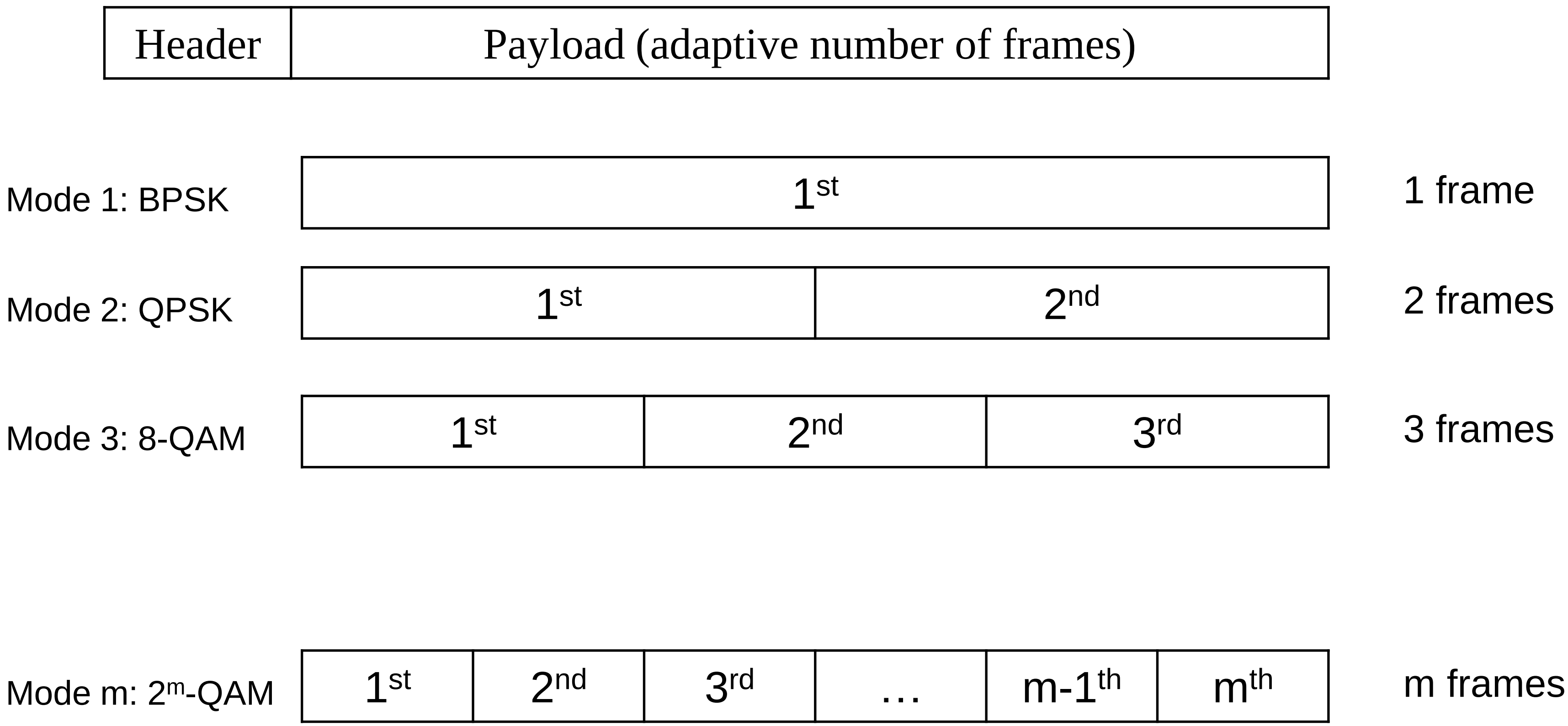
- Frame structure



Maximum no. of retransmissions: N_{re}

Burst Transmission Design

- How to design a burst transmission?



References

[1] 2010. Hybrid ARQ for FSO Communications Through Turbulent Atmosphere.

[2] 2012. Information Theoretic Analysis of Hybrid-ARQ Protocols in Coherent Free-Space Optical Systems.

[3] 2014. On the Performance Analysis of Hybrid ARQ With Incremental Redundancy and With Code Combining Over Free-Space Optical Channels With Pointing Errors.

[4] 2016. On the Performance of RF-FSO Links With and Without Hybrid ARQ.

[5] 2017. On the Performance of Millimeter Wave-Based RF-FSO Multi-Hop and Mesh Networks.

[6] 2016. 100 Gb/s Data Link Layer – from a Simulation to FPGA Implementation.

Others

1. B. Skalar, "Digital Communications: Fundamentals and Applications," second edition

2. S. Haykin, "Communication Systems," 4th edition

Thank you for your attention!
(Q&A)