ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

School of Computer and Communication Sciences

Handout 21 Principles of Digital Communications

Project Description Apr. 30, 2025

Channel Model.

We wish to communicate over a channel with two possible behaviors, but without knowledge of which one will occur. The channel acts on the even-length input sequence $x \in \mathbb{R}^n$ and produces outputs $Y \in \mathbb{R}^n$ as follows: either

$$\begin{aligned} &Y_{2k-1} = \sqrt{G}\, x_{2k-1} + Z_{2k-1}, \quad Y_{2k} = x_{2k} + Z_{2k}, \text{ or} \\ &Y_{2k-1} = x_{2k-1} + Z_{2k-1}, \quad Y_{2k} = \sqrt{G}\, x_{2k} + Z_{2k}, \end{aligned} \tag{1}$$

where $Z_1, Z_2, \dots, Z_n$ are i.i.d. $\mathcal{N}(0, \sigma^2)$ random variables. Here $G = 10$ is a known power gain parameter and $\sigma^2 = 10$ is the noise variance. Thus, either the "odd" channel inputs experience gain $\sqrt{G}$ and the "even" ones unit gain, or vice-versa. However, the transmitter and receiver do not know which of the two behaviors occurs.

The block length n of the input signal is part of your design, with the constraint that n is even and $n \leq$ 1,000,000, so that communication is efficient. For the same reason, we also enforce a constraint $\|x\|^2 \leq$ 2,000 on the transmitted energy. The channel rejects inputs that violate these constraints.

Assignment.

Develop a system capable of *reliably* transmitting text messages over the above channel. Specifically:

- Design a transmitter that reads a text message i and returns real-valued samples of an information-bearing signal $c_i \in \mathbb{R}^n$, for some even $n \leq$ 1,000,000 as above. The text message i is guaranteed to be a sequence of *40* characters from the set

  $$\mathcal{A} = \{\text{'a'}, \dots, \text{'z'}, \text{'A'}, \dots, \text{'Z'}, \text{'0'}, \text{'1'}, \dots, \text{'9'}, \text{' '}, \text{'.'}\},$$

  which has 64 elements.
- Send c_i to a server (which we provide, see below) that applies the channel effect as in (1) and returns $Y \in \mathbb{R}^n$.
- Design a receiver that, having received Y, reconstructs the text message with as few errors as possible.

Hint: Solve the theory part first.

Submission and Evaluation Rules.

- The project is to be done in teams of *four* (but we can also allow a small number of teams of *three*).

  Please choose your teammates at the latest by **Friday, May 2** and send an email to adway.girish@epfl.ch with the subject "PDC project team" in order to register your team.

- For the theoretical part, please prepare one solution per team, and submit it through Moodle. The deadline for this submission is on **Wednesday, May 28**.
- For the practical part, you may use any programming language, as long as all the code pertaining to the transmitter and receiver is produced by your team.
- During the last session of the semester (**May 30**), each group presents their project in 5–10 minutes and demonstrates their working implementation using a text message that we provide.
  - (i) You will run the transmitter and the receiver on your own laptop.
  - (ii) You must submit your team's code to Moodle before **Friday, May 30, 1 pm**. One submission per team is enough.
  - (iii) Your presentation should contain a brief explanation of your signaling scheme (this is not expected to be formal; there is no need to prepare slides, we simply want you to be able to explain your choices in designing your transmitter and receiver), followed by the transmission and decoding of the chosen text (that will be given to you on the spot).
  - (iv) You will be given *two* chances for transmission, i.e., if you fail to transmit the message during your first transmission, you can repeat the transmission once more.
- The grade is based on the solution that you submit for the theoretical part, and the reliability of your scheme during the demonstration. The theoretical part counts for 7 points of the grade and the demonstration part counts for 13 points (together counting for a total of 20 points for the project). The demonstration part will be graded as follows:
  - (a) If you manage to transmit the text message without error during one of the two transmissions, you will get full marks (13 pts).
  - (b) Otherwise your score will be $\max\{10-\chi, 0\}$ where χ is the number of incorrect characters in the reproduced text at the receiver. We will grade based on the better of the two transmissions.
  - (c) Additionally, the team which uses smallest energy $\|x\|^2$ (with $\chi = 0$) will get 5 bonus points.
  - (d) You will be given 0 if you do not attend the presentation and your teammates cannot specify your contribution during the presentation.

Python Client.

We simulate the channel (1) by a server, to which you can send an input $x \in \mathbb{R}^n$ and receive the noisy output $Y \in \mathbb{R}^n$. To access this server, we provide you a Python script `client.py` that you can download from Moodle. Please read the associated docstrings for more information. You can only connect to the server if you are inside the EPFL network. If you are working outside of EPFL, you can use EPFL's VPN (please visit the link[1] for more information). Please use the following connection parameters:

[1] https://www.epfl.ch/campus/services/en/it-services/network-services/remote-intranet-access/

- `--srv_hostname=iscsrv72.epfl.ch`
- `--srv_port=80`

For example, running the command

```
python3 client.py --input_file input.txt --output_file output.txt
--srv_hostname=iscsrv72.epfl.ch --srv_port=80
```

reads x from `input.txt` and writes the corresponding Y to `output.txt`.

[You may need to call `python` instead of `python3` depending on your installation. The client requires the `numpy` library to be installed.]

Tips and Practical Considerations.

- To avoid overloading the server, we only allow each client to connect once every 30 seconds.
- The input file for python client should be organized such that the first n lines of the input file contain the signal components $x_1, \dots, x_n$.
- The channel model is implemented on the server by the following function:

  ```
  import numpy as np
  import random

  def channel(x):
      G = 10
      sigma2 = 10
      s = random.choice([1, 2])

      n = x.size
      Y = np.random.normal(0, np.sqrt(sigma2), n)
      if s == 1:
          x_even = np.array(x[::2]) * np.sqrt(G)
          x_odd = x[1::2]
      else:
          x_even = np.array(x[::2])
          x_odd = x[1::2] * np.sqrt(G)

      Y[::2] += x_even
      Y[1::2] += x_odd

      return Y
  ```

  For testing your code locally, using this code snippet to simulate the channel (or an equivalent version in your preferred language) may accelerate your code development. However, be sure to also check your code with the server using the python client as mentioned above as the demonstration on May 30th must use the server.