ANKARA UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

COM3035 OPERATING SYSTEMS

FALL 2025-26

LAB1

PURPOSE:

In information theory, the Shannon capacity limit is the theoretical upper limit that determines the maximum data rate that can be transmitted through a noisy communication channel without error.
In an Additive White Gaussian Noise (AWGN) channel, the Shannon capacity is expressed as follows:

$$C = B \log_2 \left(1 + \frac{S}{N}\right) \quad (\text{bit/s})$$

Here:

• C: Channel capacity (bit/s)

• B: Bandwidth (Hz)

• S/N: Signal-to-noise ratio (SNR)

The bandwidth and signal-to-noise ratio values of the networks will be read from an input file with a .txt extension (such as input1.txt) until the "finish" command is received, and the Shannon limit corresponding to the channel capacity for each network will be calculated.
A higher Shannon limit means a higher maximum data transmission rate. Therefore, the highest Shannon capacity value is preferred in this scenario. Following this calculation, the Shannon capacity of each channel and the best channel selection will be written to another output file with a .txt extension (such as output1.txt).

REQUIREMENTS:

• You must use TCP socket communication.

• **Client**: Reads the channel bandwidth and signal-to-noise ratio values from the input file. Writes the Shannon limit values received from the server and selects the best channel based on these values.

• **Server**: Reads the data from the client, calculates the Shannon limit for the relevant channel, and returns the result to the client.

• **Communication**: The server opens a TCP server and listens on the specified port. The client establishes a TCP connection to the server using the host and port specified in the command line. The

client transmits the bandwidth and signal-to-noise ratio information to the server. The server performs the calculation and returns the results.

• Your code files must be in C programming language and must run via the terminal on Linux-based operating systems.

• When running your program, you must read input data from a file such as input1.txt and write output to a file such as output1.txt. **The use of fopen, fread, or fwrite within your program file is strictly prohibited!** Reading from and writing to a file must be done entirely with terminal commands. • Numerical values must be printed with two digits after the decimal point.

• **Your program output must match the format in the shared sample input-output files exactly!**

• **Your program code will be checked for copying. Code found to be unoriginal (taken from the internet or copied, etc.) will be evaluated as zero.**

SAMPLE INPUT (input1.txt):

```
10 5        # Channel 1: Bandwidth 10 Hz, Signal-to-noise ratio 5 SNR
7 3         # Channel 2: Bandwidth 7 Hz, Signal-to-noise ratio 3 SNR
12 6        # Channel 3: Bandwidth 12 Hz, Signal-to-noise ratio 6 SNR
finish # Indicates the end of the process
```

SAMPLE OUTPUT (output1.txt):

```
25.85                   # Channel 1 Shannon Capacity Limit
14.00                   # Channel 2 Shannon Capacity Limit
33.69                   # Channel 3 Shannon Capacity Limit
Selected Channel: 3     # Selected channel number
```

DELIVERY:

- Paste your codes into two files named student_number_server.c and student_number_client.c (e.g., 18888888_server.c and 18888888_client.c). You should write them in separate code files.
- **You must upload only the files student_number_server.c and student_number_client.c to the system. You do not need to upload it as a zip file. "student_number" here should be your student number. Points will be deducted for submissions that are not named correctly.**
- When running and testing both code files, you should open two separate terminals and run them separately.
- You should run your student_number_server.c file with the following commands:

- ***gcc student_number_server.c -o server -lm***
- ***./server -p 5000 &***
  - This command starts the server on port 5000 and runs it in the background.

- You should run your student_number_client.c file in another terminal with the following commands:
  - ***gcc student_number_client.c -o client -lm***
  - ***./client -h 127.0.0.1 -p 5000 < input1.txt > out1.txt***
    - With this command, you use the input1.txt file as input and write your output to the out1.txt file.
- The port number 5000 is given here as an example. If you receive a "already in use" warning, you can change this port and try using a different port number (e.g., 5555).
- You should also ensure that you uploaded your output without errors using the ***diff -w out1.txt output1.txt*** command.

TÜRKÇE:

AMAÇ:

Bilgi teorisinde Shannon limiti (Shannon capacity limit), gürültülü bir iletişim kanalından hata yapmadan aktarılabilecek maksimum veri hızını belirleyen teorik üst sınırdır.
Additive White Gaussian Noise (AWGN) kanalında Shannon kapasitesi şu şekilde ifade edilir:

$$C = B \log_2 \left(1 + \frac{S}{N}\right) \quad (\text{bit/s})$$

Burada:

- C: Kanal kapasitesi (bit/s)
- B: Bant genişliği (Hz)
- S/N: Sinyal-gürültü oranı (SNR)

Ağların bant genişliği ve sinyal-gürültü oranı değerleri .txt uzantılı bir girdi dosyasından (input1.txt gibi) “finish” komutu gelene kadar okunacak ve her ağ için kanal kapasitesine karşılık gelen Shannon limiti hesaplanacak.
Daha yüksek Shannon limiti, daha yüksek maksimum veri iletim hızı anlamına gelir. Bu nedenle en yüksek Shannon kapasite değeri bu senaryoda tercih edilir. Bu hesaplamanın ardından, her kanalın Shannon kapasitesi ve en iyi kanal tercihinin hangisi olduğu .txt uzantılı bir başka çıktı dosyasına (output1.txt gibi) yazılacak.

GEREKSİNİMLER:

- TCP soket haberleşmesi kullanmalısınız.
- **İstemci (Client):** Kanalların bant genişliği ve sinyal-gürültü oranı değerlerini girdi dosyasından okur. Sunucudan gelen Shannon limiti değerlerini yazar ve bu değerlere göre en iyi kanalı seçer.
- **Sunucu (Server):** İstemciden gelen verileri okuyup ilgili kanal için Shannon limitini hesaplar ve sonucu istemciye geri gönderir.
- **Haberleşme**: Sunucu TCP sunucusu açar ve belirtilen portta dinler. İstemci komut satırındaki host ve port ile sunucuya TCP bağlantısı kurar. İstemci, bant genişliği ve sinyal-gürültü oranı bilgilerini sunucuya iletir. Sunucu hesaplamayı yapıp sonucu geri iletir.
- Kod dosyalarınız C programlama dilinde olmalı ve Linux tabanlı işletim sistemlerinde terminal üzerinden çalışmalıdır.
- Programınızı çalıştırırken girdi verilerini input1.txt gibi bir dosyadan okumalısınız ve çıktıları da output1.txt gibi bir dosyaya yazmalısınız. **Program dosyanız içinde fopen, fread, fwrite kullanımı kesinlikle yasaktır!** Dosyadan okuma ve dosyaya yazma işlemleri tamamen terminal komutları ile yapılacaktır.
- Sayısal değerler noktadan sonra iki hane gelecek şekilde yazdırılmalıdır.
- **Program çıktılarınız paylaşılan örnek input-output dosyalarındaki formata bire bir uymalıdır!**
- **Program kodlarınız kopya kontrolünden geçirilecektir. Özgün olmadığı (internetten alınan ya da kopya vs.) tespit edilen kodlar sıfır olarak değerlendirilecektir.**

ÖRNEK GİRDİ (input1.txt):

```
10 5        # Kanal 1: Bant genişliği 10 Hz, Sinyal-gürültü oranı 5 SNR
7 3         # Kanal 2: Bant genişliği 7 Hz, Sinyal-gürültü oranı 3 SNR
12 6        # Kanal 3: Bant genişliği 12 Hz, Sinyal-gürültü oranı 6 SNR
finish      # İşlemin sona erdiğini belirtir
```

ÖRNEK ÇIKTI (output1.txt):

```
25.85                   # Kanal 1 Shannon Kapasite Limiti
14.00                   # Kanal 2 Shannon Kapasite Limiti
33.69                   # Kanal 3 Shannon Kapasite Limiti
Selected Channel: 3     # Seçilen kanal numarası
```

TESLİMAT:

- Kodlarınızı öğrenci_numarası_server.c ve öğrenci_numarası_client.c (örn: 18888888_server.c ve 18888888_client.c) olarak adlandırılan iki ayrı kod dosyasına yazmalısınız.
- **Sisteme sadece öğrenci_numarası_server.c ve öğrenci_numarası_client.c dosyalarını yüklemelisiniz. Zip dosyası olarak yüklemenize gerek yoktur. Buradaki "öğrenci_numarası" ifadesi sizin öğrenci numaranız olmalı. Doğru isimlendirilmeyen gönderimlerden puan kırılacaktır.**
- Her iki kod dosyasını çalıştırıp test ederken iki ayrı terminal açıp ayrı ayrı çalıştırmalısınız.
- öğrenci_numarası_server.c dosyanızı aşağıdaki komutlar ile çalıştırmalısınız:
  - ***gcc öğrenci_numarası_server.c -o server -lm***
  - ***./server -p 5000 &***
    - Bu komut sunucuyu 5000 numaralı portta başlatır ve arka planda çalıştırır.
- öğrenci_numarası_client.c dosyanızı da bir başka terminalde aşağıdaki komutlar ile çalıştırmalısınız:
  - ***gcc öğrenci_numarası_client.c -o client -lm***
  - ***./client -h 127.0.0.1 -p 5000 < input1.txt > out1.txt***
    - Bu komut ile girdi olarak input1.txt dosyasını kullanıp kendi çıktınızı out1.txt dosyasına yazdırmış olursunuz.
- Burada port numarası 5000 örnek olarak verilmiştir. Eğer çoktan kullanılıyor uyarısı alırsanız bu portu değiştirip farklı bir port numarası (örn: 5555) ile deneyebilirsiniz.
- ***diff -w out1.txt output1.txt*** komutu ile de çıktınızı hatasız bir şekilde yüklediğinizden emin olmalısınız.