206862666_208040261

רשתות – מטלה 3

התחלנו עם ה server:

ד

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('localhost', 12345))
server_socket.listen(1)
print("Server is listening...")
conn, addr = server_socket.accept()
print(f"Connected by {addr}")
```

פקודה לפתיחת socket חדש כאשר:

```
variable = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

`SOCK_STREAM` means that it is a TCP socket.

`SOCK_DGRAM` means that it is a UDP socket.

`AF_INET` is an **a**ddress **f**amily that is used to designate the type of addresses that your socket can communicate with (in this case, Internet Protocol v4 addresses). When you create a socket, you have to specify its address family, and then you can only use addresses of that type with the socket. The Linux kernel, for example, supports 29 other address families such as UNIX (`AF_UNIX`) sockets and IPX (`AF_IPX`), and also communications with IRDA and Bluetooth (`AF_IRDA` and `AF_BLUETOOTH`, but it is doubtful you'll use these at such a low level).

For the most part, sticking with `AF_INET` for socket programming over a network is the safest option. There is also `AF_INET6` for Internet Protocol v6 addresses.

Finally, the argument to `listen` tells the socket library that we want it to queue up as many as 5 connect requests (the normal max) before refusing outside connections. If the rest of the code is written properly, that should be plenty.

במחקלת socket:

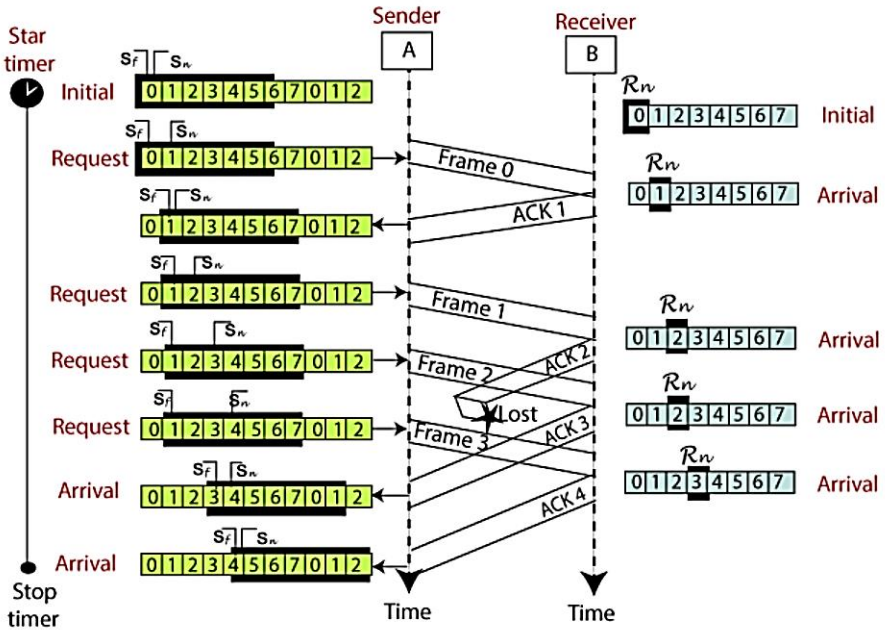
The primary socket API functions and methods in this module are:

- `socket()`
- `.bind()`
- `.listen()`
- `.accept()`
- `.connect()`
- `.connect_ex()`
- `.send()`
- `.recv()`
- `.close()`

Sliding window protocol – go-back-N ARQ:

The size of the sender window is N in this protocol. For example, Go-Back-8, the size of the sender window, will be 8. The receiver window size is always 1.

If the receiver receives a corrupted frame, it cancels it. The receiver does not accept a corrupted frame. When the timer expires, the sender sends the correct frame again. The design of the Go-Back-N ARQ protocol is shown below.



קביעת הפורמט :

```
def input_format():
    return {
        "message": "this is test message",
        "max_message_size": 20,
        "window_size": 4,
        "timeout": 5
    }
```

```
client_socket.send(max_message_size.encode(FORMAT))
```

אחרי שיצרתי חיבור שלחנו אותו לשרת דרך הסוקט, יש צורך בלקודד אותו על מנת להעביר אותו.

בצד שרת:

```
data = conn.recv(HEADER).decode(FORMAT)
```

ווידוא שהגיע

עכשיו צריך להבין איך לחלק את ההודעה לצ'אנקים לפי כמו הבתים שנקבעו.

בקובץ:

Since you are working with a file, the easiest is to simply read the file 430 bytes at a time.

```
CHUNK_SIZE = 430

f = open(image_file, 'rb')
chunk = f.read(CHUNK_SIZE)
while chunk: #loop until the chunk is empty (the file is exhausted)
        send_chunk_to_space(chunk)
        chunk = f.read(CHUNK_SIZE) #read the next chunk
f.close()
```

במשתנים:

Alternate Implementation :

Python

```python
l = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# How many elements each
# list should have
n = 4

# using list comprehension
x = [l[i:i + n] for i in range(0, len(l), n)]
print(x)
```

Output:

```
[[1, 2, 3, 4], [5, 6, 7, 8], [9]]
```

Example

Convert from JSON to Python:

```python
import json

# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["age"])
```

על מנת שיהיה קל לקרוא מידע מהקובץ השתמשנו בjson שימיר את המידע לdictionary

רשתות – מטלה 3

- מה שאומר – שלפי בקשת המטלה, נצטרך שבקובץ כל מידע יצטרך להיות בשורה בקובץ.
- מגבלה נוספת שיש, על מנת לוודא בשרת שהוא מקבל את המידע בסדר הנכון אז קודדנו את המידע בצורה כזו:
  - "massage"|{seq_number}
- מה שאומר שאם ההודעה ממש ארוכה, נצטרך שגודל הבתים המקסימלי לכל הודעה צריך להיות מספיק גדול כדי שיהיה מקום גם להודעה וגם למספר הודעה.
  - לדוגמה: "poop"|245
  - צריך לוודא שההודעה כולה עומדת בתנאי של שליחת גודל הקבצים המקסימלי. שלא יווצר מצב שבו יש מקום רק למספר ההודעה בלי להוסיף עוד מידע על זה.

ארכיטקטורה –

- נפעיל שרת, בשרת נחכה לחיבור של לקוח. ברגע שקיים חיבור הלקוח בוחר את סוג תצורת שליחת ההודעה:
  - 1 – input by keyboard
  - 2 – input by files
- במקרה והלקוח בחר 1 – השרת יקליד את גודל ההודעה המקסימלי וישלח ללקוח.
  - הלקוח יקבל ויכבד את בקשת השרת לגודל המקסימלי.
  - נבחר את הקלט שנרצה להעביר לשרת, גודל החלון, ומה זמן שהוא צריך לחכות על מנת לנג'ס שוב לשרת.
- במקרה והלקוח בחר 2 – השרת הלקוח יקראו את המידע הרצוי מהקובץ שנגיד לו לפי path שלו. צריך לכתוב את ה path שבו יש את הקובץ שממנו נקרא.
  - לדוגמה: file_path = "/home/parallels/Desktop/tomer.txt"
  - בשלב הזה הלקוח מפענח את המידע שיש בקובץ וממיר למשתנים ומעביר את המידע הזה לפונקציה של החלון.
  - השרת עושה את זה באופן דומה אך וק לגודל הבתים המקבמלים שנשלח.

Client.py –

- 1 – input from keyboard
- 2 – reading from file : fopen to read the file, and json for easier translation to variables.

Server.py –

- Sames as client for 1 and 2

Wireshark readings:

server:

```
C:\Users\ariel\PycharmProjects\ex_3\.venv\Scripts\python.exe C:\Users\ariel\PycharmProjects\ex_3\server.py
Server is listening...
Got connection from ('127.0.0.1', 59792)
1
input max size byte for client
6
ack0
hell
ack1
o wo
ack2
rld,
ack3
 how
ack4
 you
ack5
 doi
ack6
ng..
ack7
. im
ack8
 ver
ack9
y ti
ack10
red
ack11
 :)
[message received from client]
hello world, how you doing... im very tired :)
Server is listening...
```

client:

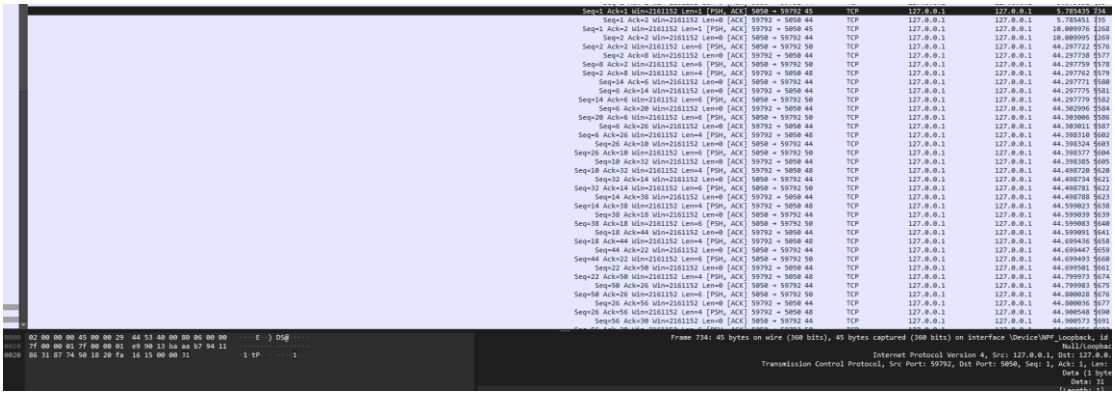
```
Connected to server
Select input type:
1 - Text input
2 - File input
1
[wait for server response...]
input data for server
hello world, how you doing... im very tired :)
select sliding window size (bytes)
3
input timeout for server messages
1
dividing the data into chunks 6 size
[{'seq': 0, 'data': b'0|hell', 'sent_time': 0.0, 'ack': False}, {'seq': 1, 'data': b'1|o wo', 'sent_time': 0.0, 'ack': False}, {'seq': 2, 'data': b'2|rld,', 'sent_time': 0.0, 'ack': False}, {'seq': 3, 'data': b'3| how', 'se

sending messages
sent to server:  b'0|hell'
sent to server:  b'1|o wo'
sent to server:  b'2|rld,'
Acknowledgment message M0
sent to server:  b'3| how'
Acknowledgment message M1
sent to server:  b'4| you'
Acknowledgment message M2
sent to server:  b'5| doi'
Acknowledgment message M3
sent to server:  b'6|ng..'
Acknowledgment message M4
sent to server:  b'7|. im'
Acknowledgment message M5
sent to server:  b'8| ver'
Acknowledgment message M6
sent to server:  b'9|y ti'
```

```
ent to server:  b'8| ver'
cknowledgment message M6
ent to server:  b'9|y ti'
cknowledgment message M7
ent to server:  b'10|red'
cknowledgment message M8
ent to server:  b'11| :)'
cknowledgment message M9
cknowledgment message M10
cknowledgment message M11
```

206862666_208040261

רשתות – מטלה 3



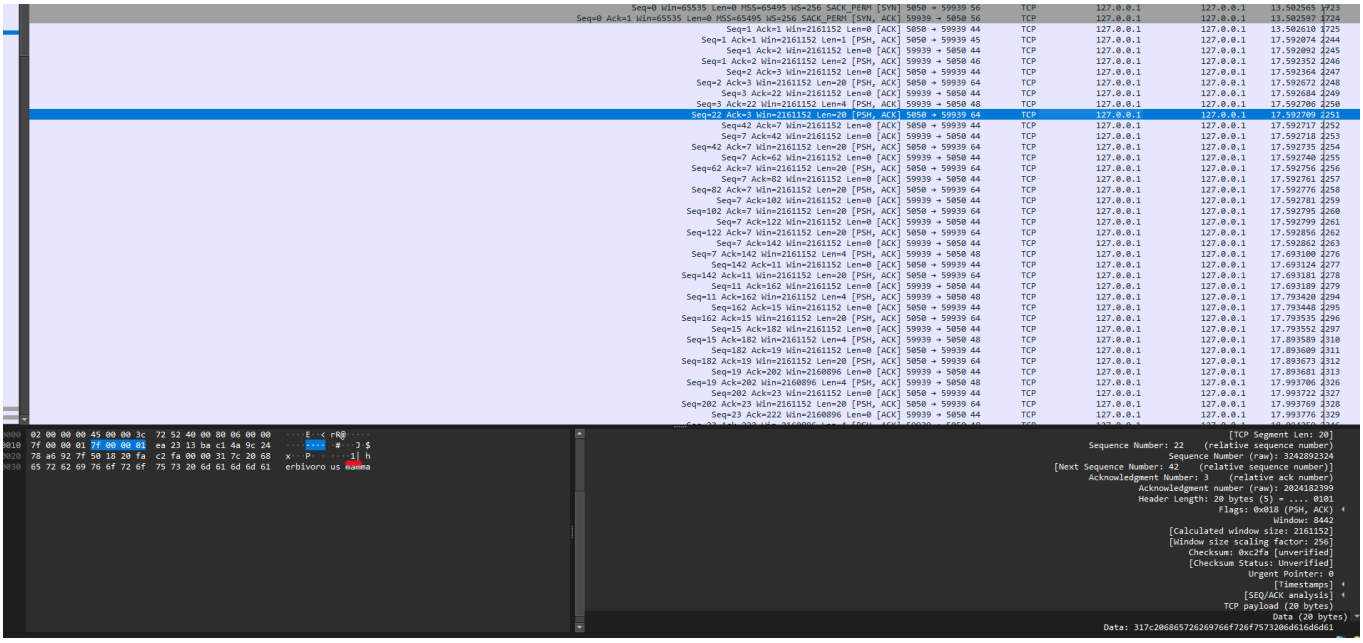
הלקוח שולח הודעה 0 לשרת. ואכן גודל כל הודעה הוא 6 בתים לבקת השרת.

שליחת הודעה הראשונה מהלקוח לשרת

קבלת הודעה ack 0 מהשרת עבור הודעה 0 מהלקוח.

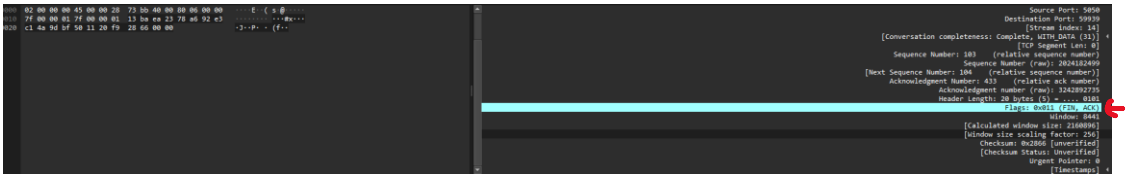
עבור קריאה מקובץ:

השרת מקבל 2



נשלחת הודעה לשרת בגודל 20 בתים לפי הנדרש. וקבלת ההודעה מהשרת עם ack.

- כמובן ההקלטות יתווספו לקובץ זיפ שתוכל להתרשם מהם בעצמך.



לאחר סיום התקשורת השרת מזהה שהיה סגירת חיבור.

```
C:\Users\ariel\PycharmProjects\ex_3\.venv\Scripts\python.exe C:\Users\ariel\PycharmProjects\ex_3\client.py
Connected to server
Select input type:
1 - Text input
2 - File input
2
[wait for server response...]
dividing the data into chunks 15 size
[{'seq': 0, 'data': b'0|Rabbits are s', 'sent_time': 0.0, 'ack': False}, {'seq': 1, 'data': b'1|mall, herbivo', 'sent_time': 0.0, 'ack': False}, {'seq': 2, 'data': b'2|rous mammals ', 'sent_time': 0.0, 'ack': Fals

sending messages
sent to server:  b'0|Rabbits are s'
sent to server:  b'1|mall, herbivo'
sent to server:  b'2|rous mammals '
sent to server:  b'3|belonging to '
sent to server:  b'4|the family Le'
Acknowledgment message M0
sent to server:  b'5|poridae. Know'
Acknowledgment message M1
sent to server:  b'6|n for their l'
Acknowledgment message M2
sent to server:  b'7|ong ears, sof'
Acknowledgment message M3
sent to server:  b'8|t fur, and di'
Acknowledgment message M4
sent to server:  b'9|stinctive hop'
Acknowledgment message M5
sent to server:  b'10|ping movemen'
Acknowledgment message M6
sent to server:  b'11|t, they are '
Acknowledgment message M7
sent to server:  b'12|found in man'
Acknowledgment message M8
sent to server:  b'13|y parts of t'
Acknowledgment message M9
sent to server:  b'14|he world and'
Acknowledgment message M10
sent to server:  b'15| are widely '
Acknowledgment message M11
Acknowledgment message M12
sent to server:  b'17|or their pla'
Acknowledgment message M13
sent to server:  b'16|recognized f'
sent to server:  b'18|yful and gen'
Acknowledgment message M14
sent to server:  b'19|tle nature. '
Acknowledgment message M15
sent to server:  b'20|Rabbits come'
Acknowledgment message M17
Acknowledgment message M16
sent to server:  b'21| in a variet'
Acknowledgment message M18
Acknowledgment message M19
Acknowledgment message M20
Acknowledgment message M21
Timeout occurred, resending unacknowledged chunks.
sent to server:  b'17|or their pla'
sent to server:  b'18|yful and gen'
sent to server:  b'20|Rabbits come'
sent to server:  b'21| in a variet'
Acknowledgment message M17
```

```
sent to server:  b'19|tle nature. '
sent to server:  b'22|y of sizes, '
Acknowledgment message M18
Acknowledgment message M20
sent to server:  b'23|colors, and '
Acknowledgment message M21
Acknowledgment message M19
sent to server:  b'24|breeds, rang'
Acknowledgment message M22
Acknowledgment message M23
Acknowledgment message M24
Timeout occurred, resending unacknowledged chunks.
sent to server:  b'20|Rabbits come'
sent to server:  b'21| in a variet'
sent to server:  b'22|y of sizes, '
sent to server:  b'24|breeds, rang'
Acknowledgment message M20
sent to server:  b'23|colors, and '
sent to server:  b'25|ing from the'
Acknowledgment message M21
sent to server:  b'26| small Nethe'
```

```
Acknowledgment message M22
sent to server:  b'27|rland Dwarf '
Acknowledgment message M24
Acknowledgment message M23
sent to server:  b'28|to the large'
Acknowledgment message M25
Acknowledgment message M26
Acknowledgment message M27
Acknowledgment message M28
Timeout occurred, resending unacknowledged chunks.
sent to server:  b'24|breeds, rang'
sent to server:  b'25|ing from the'
sent to server:  b'26| small Nethe'
sent to server:  b'27|rland Dwarf '
Acknowledgment message M24
sent to server:  b'28|to the large'
sent to server:  b'29|r Flemish Gi'
Acknowledgment message M25
sent to server:  b'30|ant.'
Acknowledgment message M26
Acknowledgment message M27
Acknowledgment message M28
Acknowledgment message M29
Acknowledgment message M30
```

דוגמה למקרה בו בכוונה שיבשנו את סדר פעולת השליחה של המידע, על מנת לראות שבאמת שהלקוח אכן מזהה שלא קיבל את מספר הack הנכון, ושולח שוב לשרת את מה שעוד לא קיבל בחלון מחדש.

אחרי קבלת הודעה 15, הוא קפץ וקיבל 17 ואז את 16 מה שיצראת זה שצריך לשלוח שוה 17 18 19, ואכן ניתן לראות ששלח שוב.

מקורות:

- https://stackoverflow.com/questions/35725732/what-is-the-function-of-sock-stream
- https://docs.python.org/3/howto/sockets.html
- https://realpython.com/python-sockets/
- https://www.javatpoint.com/sliding-window-protocol
- https://www.reddit.com/r/learnpython/comments/51hxul/how_do_you_split_a_/binary_file_into_specific
- https://www.geeksforgeeks.org/break-list-chunks-size-n-python/
- https://www.w3schools.com/python/python_json.asp