

מטלה 3

רשתות תקשורת

מגישים:

אילן שמחון ת.ז. 212036396

תומר גוזלן ת.ז. 314770058

הקדמה - מהות המטלה

מטרתנו במטלה זו הינה לחקור את איכות האלגוריתמים reno וcubic אשר מטרם הינה לנהל את זרימת הנתונים בחיבורי TCP ולוודא התנהלות תקינה של העברת נתונים, תוך התמודדות עם בעיות התעכבות ואיבוד מידע ברצוננו לענות על השאלה איזה מהם איכותי מהיר יותר, נסביר כיצד נבדוק זאת:

הקוד בנוי בשני קבצים: sender.py, receiver.py

תחילה ניצור חיבור client-server בין השולח של הקובץ לבין המקבל שלו לאחר שנוצר חיבור בין הsender לreceiver השולח לו קובץ טקסט (בגודל מעל MB2) בשני חלקים שונים כאשר בחלק הראשון מופעל אלגוריתם reno ואילו בחלק השני מופעל אלגוריתם cubic, הreceiver שומר את הזמנים שלקח לו לקבל את כל אחד מהחלקים על מנת לראות בעזרת איזה אלגוריתם זה נעשה יותר מהר

אנו נריץ את התכנית מספר פעמים כאשר בכל פעם נשנה את אחוז איבוד הפאקטות (בין 0 ל 20 אחוז) ונבחן את התנהגות התוכנה בתנאים אלו ובעיקר את זמן הריצה שלוקח לכל אלגוריתם

חלק א' – מהלך הקוד

נעבור כעת על מהלך הקוד בצורה משולבת בין 2 הקבצים ונראה מה קורה בכל קובץ במקביל

(1) הגדרת משתנים ופונקציות שימשו אותנו בתכנית:

בשלב זה נגדיר משתנים בשביל בדיקת התקינות, נפתח את file בכל אחד מהקבצים במצב המתאים ונגדיר משתנים ליצירת החיבור (ת.ז מספר 2 נגמרת ב0058 ולכן זה מוגדר להיות רק 58)

```
import socket
import time
from statistics import mean

ID1 = 6396
ID2 = 58

file_path = "receiver_file.txt"
file = open(file_path, 'wb')

part1_time = []
part2_time = []

receiver_ip = '127.0.0.1'
receiver_port = 9999
```

```
import socket
import os

ID1 = 6396
ID2 = 58

2 usages
def check_got(xor):
    bool = (xor == (ID1 ^ ID2))
    return bool

file_path = "myFile1.txt"
file = open(file_path, "r")

file_len = os.path.getsize(file_path)
half_len = file_len // 2

receiver_ip = '127.0.0.1'
receiver_port = 9999
```

(2) יצירת חיבור בין הsender לreceiver וקריאת הקובץ

בחלק זה ניצור חיבור TCP בין הsender לבין הreceiver, הreceiver יוצר חיבור:

```
receiver_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
buffer_size = receiver_socket.getsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF)

print("Creating a connection...")
receiver_socket.bind((receiver_ip, receiver_port))

receiver_socket.listen(1)
print("listening on IP:" + receiver_ip + " Port:" + receiver_port)
```

receiver מתחבר אל החיבור שיצר sender

```
sender_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
buffer_size = sender_socket.getsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF)

print("Connecting to receiver...")
sender_socket.connect((receiver_ip, receiver_port))
```

הreceiver מאשר את בקשת החיבור של sender והחיבור מושלם:

```
sender_socket, sender_address = receiver_socket.accept()
print("connection Succeeded!")
```

הsender מחלק את הקובץ ל2 חלקים ושולח לreceiver את גודל הקובץ:

```
file_read = file.read()

part1 = file_read[:half_len]
part2 = file_read[half_len:]

file.close()
print("Sending the len of the file")
sender_socket.send(str(file_len).encode())
```

הreceiver מקבל גודל הקובץ מהsender:

```
file_len = int(sender_socket.recv(buffer_size))
```

3) כעת ניכנס ללולאה הראשית בה נשלח בכל איטרציה את הקובץ פעם אחת
הsender מגדיר אצלו את האלגוריתם CC להיות reno (עבור החלק הראשון בלבד) ואז שולח
את החלק הראשון לreceiver:

```
send = True
while send is True:

    print("Defines the CC algorithm be Reno.")
    sender_socket.setsockopt(socket.IPPROTO_TCP, socket.TCP_CONGESTION, b'reno')

    print("Sending the first part of the file..")
    sender_socket.send(part1.encode())
```

החלק הראשון

במקביל, receiver מגדיר גם אצלו את האלגוריתם CC להיות reno ואז מקבל בלולאה את החלק הראשון של הקובץ מהsender, כשהוא מסיים לקרוא את החלק הראשון הוא מחשב את הזמן שזה לקח ושומר אותו בlist, לאחר מכן הוא שולח לreceiver אישור שהקובץ התקבל בהצלחה ע"י שליחת xor של 2 התי"ז כפי שנדרש (ת.ז מספר 2 נגמרת ב0058 ולכן זה מוגדר להיות רק 58)

```
while file_len != -1:
    print("Defines the CC algorithm be reno.")
    receiver_socket.setsockopt(socket.IPPROTO_TCP, socket.TCP_CONGESTION, b'reno')

    start_time = time.time()

    data_size1 = 0
    print("receives the first part of the file")
    print(file_len // 2)
    while data_size1 < file_len // 2:
        chunk = sender_socket.recv(buffer_size)
        data_size1 += len(chunk.decode())
        file.write(chunk)

    first_part_time = time.time() - start_time
    part1_time.append(first_part_time)

    print("Sending OK to the sender")
    sender_socket.send((str(ID1 ^ ID2)).encode())
```

קבלת האק
הראשון

שליחת
אימות

כעת, הsender מקבל מהreceiver את התוצאה של xor ושולח אותה לפונקצייה שבודקת האם זה מה שהיה צריך להתקבל, אם זה שגוי תיזרק שגיאה

```
xor = float(sender_socket.recv(buffer_size).decode())
if check_got(xor) is False:
    Exception
else:
    print("Succeeded!")
```

בדיקת אימות

ובכן, סיימנו לשלוח בהצלחה את החלק הראשון ונעבור לשלוח את החלק השני
נשנה את האלגוריתם CC להיות cubic
לאחר מכן נשלח את החלק השני ואז נקבל מה receiver שוב את האות המוסכם שמאשר כי המידע הנכון
הגיע אליו:

קוד המקור:

```
print("Defines the CC algorithm be Cubic")
sender_socket.setsockopt(socket.IPPROTO_TCP, socket.TCP_CONGESTION, b'cubic')

print("Sending the second part of thr file..")
sender_socket.send(part2.encode())

xor = float(sender_socket.recv(buffer_size).decode())
if check_got(xor) is False:
    raise Exception
```

הערה: קוד המקור מציין "Sending the second part of thr file.." (יש טעות באותיות).

במקביל יתרחש תהליך זהה אצל ה receiver, הוא יקבל בלולאה את החלק השני של הקובץ מה sender, לאחר שיסיים ישמור את הזמנים וישלח הודעת אישור ל sender

קוד המקור:

```
while data_size2 < file_len - data_size1:
    chunk = sender_socket.recv(buffer_size)
    data_size2 += len(chunk)
    file.write(chunk)

second_part_time = time.time() - start_time
part2_time.append(second_part_time)

print("Sending OK to the sender")
sender_socket.send((str(ID1 ^ ID2)).encode())
```

הערה: קוד המקור מציין "Sending OK to the sender".

כעת, על המשתמש לקבל החלטה האם לשלוח את הקובץ שוב, או לסגור את החיבור ולסיים את התכנית.
ה sender שואל את המשתמש שמחליט האם להמשיך ושולח את התשובה ל receiver.
במידה והוחלט להמשיך, ה sender חוזר לתחילת הלולאה. במידה והוחלט לסיים הוא יוצא מהלולאה וסוגר את החיבור.

קוד המקור:

```
again = input("Should you send the file again? (y/n):")
if again == "n":
    send = False
    sender_socket.send("Stop sending the file".encode())
    sender_socket.recv(buffer_size).decode()
else:
    sender_socket.send("Sending the file again!".encode())

sender_socket.close()
```

הערה: קוד המקור מציין "Should you send the file again? (y/n):".

במקביל אצל receiver, הוא מקבל את תוצאת החלטת המשתמש מהsender האם הוא מקבל את הקובץ שוב או לא.

אם הוחלט להמשיך ולשלוח שוב את הקובץ, הוא חוזר לתחילת הלולאה ומתחיל לקלוט את הקובץ מההתחלה

אחרת, הוא יוצר מהלולאה ומדפיס את כל הזמנים שלקח לו לקבל את הקובץ מהsender, כולל ממוצע לכל חלק בנפרד וכולל ממוצע כללי

לאחר מכן, הreceiver סוגר את החיבור לsender ואת הקובץ שאליו כתב. קבלת ההחלטה

זורם לצורת
האל אק

```
decision = sender_socket.recv(buffer_size).decode()
print(decision)
if decision == "Stop sending the file":
    file_len = -1
    sender_socket.send("Thank you, goodbye!".encode())
```

הקבוצה
סטטיסטיקה

```
for i in range(1, len(part1_time)+1):
    print("Time of receiving number ", i, ": ")
    print("part 1: ", part1_time[i-1], " seconds")
    print("part 2: ", part2_time[i-1], " seconds")
```

```
print("Average of part 1:", mean(part1_time), "seconds")
print("Average of part 2:", mean(part2_time), "seconds")
print("Total Average: ", mean(part1_time+part2_time), "seconds")
print("Close the connection..")
```

```
receiver_socket.close()
file.close()
```

סגירת
חיבור וקובץ

חלק ב – הרצה כללית

בחלק זה נריץ את הקוד בצורה רגילה ללא איבוד פאקטות על מנת להראות את ההתנהלות התקינה של ההרצה ואת ההדפסות במהלך הריצה, במהלך ההרצה נבקש מהsender לשלוח 2 פעמים את הקובץ כיוון שהיא מיועדת רק על מנת להמחיש את התהליך

נריץ את ה receiver

```
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject$ python3 receiver.py
Creating a connection...
listening on IP: 127.0.0.1 Port: 9999
connection Succeeded!
```

קבלת חלק ראשון

```
Defines the CC algorithm be reno.
receives the first part of the file
Sending OK to the sender
```

קבלת חלק שני

```
Defines the CC algorithm be Cubic..
receives the second part of the file
Sending OK to the sender
```

בתמונה זו אנו רואים חיבור ושליחה ראשונה של הקובץ, כעת ה receiver מחכה לעדכון מה sender האם הוא שולח פעם נוספת את הקובץ או לא

נריץ את ה sender

```
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject$ python3 sender.py
Connecting to receiver...
connection Succeeded!
```

```
Sending the len of the file
```

שלוחת חלק ראשון

```
Defines the CC algorithm be Reno.
Sending the first part of the file..
Succeeded!
```

שלוחת חלק שני

```
Defines the CC algorithm be Cubic
Sending the second part of thr file..
Succeeded!
```

החלטת המשתמש

```
Should you send the file again? (y/n):
```

בתמונה זו אנו רואים את מהלך החיבור של ה sender ל receiver ואת שליחה הקובץ בפעם הראשונה, לבסוף שואלת התכנית את המשתמש האם לשלוח את הקובץ שוב, במידה והמשתמש בוחר לשלוח שוב התכנית תשלח את הקובץ פעם נוספת עד שהמשתמש יחליט לעצור

הקובץ
שליח חוזר

נציג בקשה לשלוח את הקובץ פעם נוספת:

```
Sending the file again!

Defines the CC algorithm be reno.
receives the first part of the file
Sending OK to the sender

Defines the CC algorithm be Cubic..
receives the second part of the file
Sending OK to the sender

Stop sending the file

Time of receiving number 1 :
part 1: 0.0029425621032714844 seconds
part 2: 0.0014846324920654297 seconds

Time of receiving number 2 :
part 1: 0.004225969314575195 seconds
part 2: 0.003233194351196289 seconds

Average of part 1: 0.00358426570892334 seconds
Average of part 2: 0.0023589134216308594 seconds
Total Average: 0.0029715895652770996 seconds

Close the connection..
(env) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject$
```

```
Should you send the file again? (y/n):y
```

```
Defines the CC algorithm be Reno.
Sending the first part of the file..
Succeeded!
```

```
Defines the CC algorithm be Cubic
Sending the second part of the file..
Succeeded!
```

```
Should you send the file again? (y/n):n
```

```
Close the connection..
```

בתמונה הימנית, רואים את התכנית שואלת את המשתמש של הsender האם לשלוח את הקובץ פעם נוספת, המשתמש עונה בחיוב ולכן התכנית מעדכנת את הreceiver ואכן רואים בתמונה השמאלית את הreceiver מדפיס הודעה שהקובץ עומד להישלח שוב וכן ניתן לראות גם בsender וגם בreceiver את תהליך שליחת הקובץ שוב, לאחר מכן התכנית שואלת את המשתמש האם לשלוח שוב (שלחנו את הקובץ כבר פעמיים כמו שרצינו בדוגמא זו ולכן נענה בשלילה) ואכן רואים את הsender ואת הreceiver מדפיסים הודעה על סגירת החיבור וסיום התכנית.

בנוסף, לאחר ההודעה מהsender על סגירת החיבור הreceiver מדפיס את כל הזמנים וסיכום הנתונים.

חלק ג – איבוד פאקטות

בחלק זה נבחן את ההבדלים שנגרמים בהרצת הקוד כתוצאה מאיבוד של פאקטות בתהליך התקשורת, לפי הנתונים שנקבל תחת הרצות אלו נוכל לקבוע איזה אלגוריתם טוב יותר

אנו מריצים את הקוד 4 פעמים, בכל פעם עם אחוז איבוד פאקטות שונה בהתאם להוראות המטלה (20%,15%,10%,0%)

כזכור, בהרצה יש למשתמש אפשרות לבחור האם לשלוח את הקובץ שוב או לסגור את ההתחברות, בכל הרצה נבחר לשלוח את הקובץ 5 פעמים על מנת לקבל נתונים מספיק מבוססים בנוגע לזמן השליחה על פי כל אלגוריתם ולהבדלים שביניהם

נריץ את הקוד, נציג את הזמנים שקיבלנו ונוסיף צילומי מסך רלוונטיים מהקלטות ה wireshark

הרצה ראשונה – ללא איבוד פאקטות

בהדפסת הזמנים נקבל אינדיקציה ראשונית כמה זמן לוקח לתכנית לרוץ ואיזה אלגוריתם הוא טוב יותר ללא ההתערבות שלנו באובדן פאקטות

ובכן כפי שניתן לראות בצילום ההרצה, מהנתונים שאנו מקבלים לאחר שליחת הקובץ 5 פעמים, מהירות השליחה באמצעות כל אחד מהאלגוריתמים קרובה מאוד, עם יתרון קל לאלגוריתם cubic

```
Stop sending the file

Time of receiving number 1 :
part 1: 0.002384185791015625 seconds
part 2: 0.0011560916900634766 seconds

Time of receiving number 2 :
part 1: 0.004258155822753906 seconds
part 2: 0.003230571746826172 seconds

Time of receiving number 3 :
part 1: 0.0036559104919433594 seconds
part 2: 0.0033414363861083984 seconds

Time of receiving number 4 :
part 1: 0.0038270950317382812 seconds
part 2: 0.003756284713745117 seconds

Time of receiving number 5 :
part 1: 0.005266427993774414 seconds
part 2: 0.003618001937866211 seconds

Average of part 1: 0.003878355026245117 seconds
Average of part 2: 0.003020477294921875 seconds
Total Average: 0.003449416160583496 seconds

Close the connection..
(env) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject$
```

נתבון בחקלטות הwireshark

נראה בתחילה את הפאקטות המעידות על יצירת חיבור בין sender לreceiver, ניתן לזהות אותם לפי סוגם, פאקטות SYN המעידות על בקשה לחיבור ופאקטות ACK המאשרות את החיבור

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	51628 → 9999 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=835183653 TSecr=0 WS=128
2	0.000000123	127.0.0.1	127.0.0.1	TCP	74	9999 → 51628 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=835183653 TSecr=835183653 WS=128
3	0.000014842	127.0.0.1	127.0.0.1	TCP	66	51628 → 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=835183653 TSecr=835183653

כעת, יישלחו פאקטות רבות המעבירות את המידה, ניתן לזהות את העברת המידע מתחילה בפאקטה הרביעית ע"י זה שנראה כי הפאקטה מעבירה את אורך הקובץ לreceiver, ולאחר מכן המידע מתחיל לעבור

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	51628 → 9999 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=835183653 TSecr=0 WS=128
2	0.000000123	127.0.0.1	127.0.0.1	TCP	74	9999 → 51628 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=835183653 TSecr=835183653 WS=128
3	0.000014842	127.0.0.1	127.0.0.1	TCP	66	51628 → 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=835183653 TSecr=835183653
4	0.004114744	127.0.0.1	127.0.0.1	TCP	73	51628 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=7 TSval=835183657 TSecr=835183653 [TCP segment of a reassembled PDU]
5	0.004122378	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=8 Win=65536 Len=0 TSval=835183657 TSecr=835183657
6	0.004094066	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=0 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183657 [TCP segment of a reassembled PDU]
7	0.00488344	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=32776 Win=48512 Len=0 TSval=835183658 TSecr=835183658
8	0.004896794	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=32776 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183657 [TCP segment of a reassembled PDU]
9	0.00498806	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=65544 Win=15744 Len=0 TSval=835183658 TSecr=835183658
10	0.004923854	127.0.0.1	127.0.0.1	TCP	66	[TCP Window Update] 9999 → 51628 [ACK] Seq=1 Ack=65544 Win=65536 Len=0 TSval=835183658 TSecr=835183658
11	0.004966440	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=65544 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183658 [TCP segment of a reassembled PDU]
12	0.004973513	127.0.0.1	127.0.0.1	TCP	32834	[TCP Window Full] 51628 → 9999 [PSH, ACK] Seq=88312 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
13	0.005059550	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=131888 Win=65536 Len=0 TSval=835183658 TSecr=835183658
14	0.005147891	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=131888 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
15	0.005159519	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=163848 Win=196488 Len=0 TSval=835183658 TSecr=835183658
16	0.005157253	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=163848 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
17	0.005159088	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=196616 Win=327424 Len=0 TSval=835183658 TSecr=835183658
18	0.005166535	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=196616 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
19	0.005169124	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=199924 Win=65536 Len=0 TSval=835183658 TSecr=835183658
20	0.005189117	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
21	0.005191050	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=199924 Win=65536 Len=0 TSval=835183658 TSecr=835183658
22	0.005196429	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
23	0.005197975	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=199924 Win=65536 Len=0 TSval=835183658 TSecr=835183658
24	0.005203387	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
25	0.005204878	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=199924 Win=65536 Len=0 TSval=835183658 TSecr=835183658
26	0.005218771	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
27	0.005214483	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=199924 Win=65536 Len=0 TSval=835183658 TSecr=835183658
28	0.005217631	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
29	0.005219367	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=199924 Win=65536 Len=0 TSval=835183658 TSecr=835183658
30	0.005225215	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
31	0.005231819	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
32	0.005238698	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
33	0.005247301	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
34	0.005253531	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=199924 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]

כעת נבחין בפאקטה (מספר 70) המעבירה את הsize של תעודות הזהות שלנו (6396 ^ 0058), פאקטות האוטנטיקאציה לחלק הראשון של הקובץ

tcp.port = 9999

No.	Time	Source	Destination	Protocol	Length	Info
38	0.005282650	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=655368 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
39	0.005290678	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=688136 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
40	0.005290902	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=720984 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
41	0.005306178	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=753672 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
42	0.005313941	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=786440 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
43	0.005320806	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=819200 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
44	0.005334240	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=851976 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
45	0.005341070	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=884744 Ack=1 Win=65536 Len=32768 TSval=835183658 TSecr=835183658 [TCP segment of a reassembled PDU]
46	0.005456972	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1 Ack=917512 Win=792192 Len=0 TSval=835183659 TSecr=835183658
47	0.005469667	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=917512 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
48	0.005476936	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=950280 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
49	0.005487035	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=983048 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
50	0.005495113	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1015816 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
51	0.005505158	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1048584 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
52	0.005507873	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1081352 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
53	0.005513981	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1114120 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
54	0.005519999	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1146080 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
55	0.005527228	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1179056 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
56	0.005533166	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1211832 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
57	0.00554106	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1244600 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
58	0.005549097	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1277368 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
59	0.005557128	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1310136 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
60	0.005565159	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1342904 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
61	0.005573190	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1375672 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
62	0.005581221	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1408440 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
63	0.005589252	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1441208 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
64	0.005597283	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1473976 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
65	0.005605314	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1506744 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
66	0.005613345	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1539512 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
67	0.005621376	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1572280 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
68	0.005629407	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [PSH, ACK] Seq=1605048 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
69	0.005637438	127.0.0.1	127.0.0.1	TCP	32834	51628 → 9999 [ACK] Seq=1637816 Ack=1 Win=65536 Len=32768 TSval=835183659 TSecr=835183659 [TCP segment of a reassembled PDU]
70	0.005645469	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=1670584 Ack=1 Win=65536 Len=0 TSval=835183659 TSecr=835183659
71	0.005653500	127.0.0.1	127.0.0.1	TCP	70	[TCP ACKS] Seq=1670584 Ack=1 Win=65536 Len=0 TSval=835183659 TSecr=835183659
72	0.005661531	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
73	0.005669562	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
74	0.005677593	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
75	0.005685624	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
76	0.005693655	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
77	0.005701686	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
78	0.005709717	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
79	0.005717748	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
80	0.005725779	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
81	0.005733810	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
82	0.005741841	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
83	0.005749872	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
84	0.005757903	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
85	0.005765934	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
86	0.005773965	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
87	0.005781996	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]
88	0.00578	127.0.0.1	127.0.0.1	TCP	60	[TCP Previous segment not captured]

חישוב

מתכנת

6396 XOR 58 = 6,342

Wireshark - Packet 70: Loopback: lo

Frame 70: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 9999, Dst Port: 51628, Seq: 1, Ack: 2167745, Len: 4
Source Port: 9999
Destination Port: 51628
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 4]
Sequence Number: 1 (relative sequence number)

0000 00

לאחר מכן החלק השני של הקובץ ממשיך להישלח, עד שבפאקטה מספר 107 נבחין שוב בxor שלנו, המאשר את קבלת החלק השני

The image shows a Wireshark packet capture window. The main pane displays a list of network packets. Packet 107 is selected, showing it is a TCP segment from 127.0.0.1 to 127.0.0.1, port 9999 to 51628, with sequence number 5. The packet details pane shows the Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol layers. The packet bytes pane shows the raw data in hexadecimal and ASCII. The ASCII column shows the text "E-8-@-3-4PTX1z-1-1-6342".

No.	Time	Source	Destination	Protocol	Length	Info
75	0.007182565	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2298711 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
76	0.007198732	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2364194 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
77	0.007206093	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=5 Ack=2429677 Win=2413568 Len=0 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
78	0.007216094	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2429677 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
79	0.007223296	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2495108 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
80	0.007233916	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=5 Ack=2560643 Win=2675456 Len=0 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
81	0.007249076	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2560643 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
82	0.007264176	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2626126 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
83	0.007266080	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=5 Ack=2691609 Win=2912128 Len=0 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
84	0.007285756	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2691609 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
85	0.007302249	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2757992 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
86	0.007318371	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2822575 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
87	0.007334779	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2888958 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
88	0.007351089	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=2953541 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
89	0.007367256	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3019924 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
90	0.007383891	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [PSH, ACK] Seq=3084597 Ack=5 Win=65536 Len=65483 TSval=835183660 TSecr=835183660 [TCP segment of a reassembled PDU]
91	0.007408051	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3149999 Ack=5 Win=65536 Len=65483 TSval=835183661 TSecr=835183660 [TCP segment of a reassembled PDU]
92	0.007453998	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3215473 Ack=5 Win=65536 Len=65483 TSval=835183661 TSecr=835183660 [TCP segment of a reassembled PDU]
93	0.007474088	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=5 Ack=3354479 Win=2429677 Len=0 TSval=835183661 TSecr=835183661 [TCP segment of a reassembled PDU]
94	0.007487765	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
95	0.007503264	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
96	0.007529234	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
97	0.007545249	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
98	0.007568847	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
99	0.007609282	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
100	0.007660318	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] 9999 → 51628 [ACK] Seq=5
101	0.007638289	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
102	0.007697995	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
103	0.007727893	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=3
104	0.007728495	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] 9999 → 51628 [ACK] Seq=5
105	0.007731706	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [PSH, ACK] Seq=5
106	0.007789232	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=5
107	0.008439788	127.0.0.1	127.0.0.1	TCP	70	9999 → 51628 [PSH, ACK] Seq=5
108	0.051406632	127.0.0.1	127.0.0.1	TCP	66	51628 → 9999 [ACK] Seq=4

Frame 107: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 9999, Dst Port: 51628, Seq: 5, Ack: 4335483, Len: 4

כך נשלח את הקובץ שוב ושוב במשך 5 פעמים ובסופם נוזה את הפאקטה האחרונה של העברת מידע בה receivern שולח לsender הודעת תודה ולהתראות לאחר שהsender הודיע לו על סיום החיבור

לאחר מכן יופיעו פאקטות FIN המעידות על סגירת חיבור קיים

The image shows a Wireshark packet capture window. The main pane displays a list of network packets. Packet 371 is selected, showing it is a TCP segment from 127.0.0.1 to 127.0.0.1, port 9999 to 51628, with sequence number 41. The packet details pane shows the Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol layers. The packet bytes pane shows the raw data in hexadecimal and ASCII. The ASCII column shows the text "G-@-3-4PTX1z-1-1-6342".

No.	Time	Source	Destination	Protocol	Length	Info
338	15.810083204	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=37 Ack=19837152 Win=3012224 Len=0 TSval=835199463 TSecr=835199463
339	15.810099732	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=19968118 Ack=37
340	15.810142822	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20633601 Ack=37
341	15.810175151	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20699004 Ack=37
342	15.810190701	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20164567 Ack=37
343	15.810205747	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20230050 Ack=37
344	15.810222196	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20295533 Ack=37
345	15.810239795	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20361916 Ack=37
346	15.810256753	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20426499 Ack=37
347	15.810273826	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20491982 Ack=37
348	15.810288867	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=37 Ack=2036101
349	15.810292895	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20557465 Ack=37
350	15.810307982	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20622948 Ack=37
351	15.810323993	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20688431 Ack=37
352	15.810339957	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20753914 Ack=37
353	15.810356361	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20819397 Ack=37
354	15.810373294	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20884880 Ack=37
355	15.810392494	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=20950363 Ack=37
356	15.810406575	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=37 Ack=2095036
357	15.810413045	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [PSH, ACK] Seq=21015046
358	15.810430678	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=21081329 Ack=37
359	15.810448310	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=21146812 Ack=37
360	15.810465943	127.0.0.1	127.0.0.1	TCP	65549	51628 → 9999 [ACK] Seq=21212295 Ack=37
361	15.810483576	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] 9999 → 51628 [ACK] Seq=5
362	15.810501209	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [PSH, ACK] Seq=5
363	15.810518842	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [ACK] Seq=5
364	15.811211159	127.0.0.1	127.0.0.1	TCP	70	9999 → 51628 [PSH, ACK] Seq=37 Ack=21277475
365	15.851594611	127.0.0.1	127.0.0.1	TCP	66	51628 → 9999 [ACK] Seq=21677475 Ack=41 Win=65536 Len=0 TSval=835199505 TSecr=835199464
370	20.000720472	127.0.0.1	127.0.0.1	TCP	87	51628 → 9999 [PSH, ACK] Seq=21677475 Ack=41 Win=65536 Len=21 TSval=835203054 TSecr=835199464 [TCP segment of a reassembled PDU]
371	20.000830703	127.0.0.1	127.0.0.1	TCP	85	9999 → 51628 [PSH, ACK] Seq=21677496 Ack=60 Win=65536 Len=0 TSval=835203654 TSecr=835203654 [TCP segment of a reassembled PDU]
372	20.000908035	127.0.0.1	127.0.0.1	TCP	66	51628 → 9999 [ACK] Seq=21677496 Ack=60 Win=65536 Len=0 TSval=835203654 TSecr=835203654
373	20.001011837	127.0.0.1	127.0.0.1	TCP	66	51628 → 9999 [FIN, ACK] Seq=21677496 Ack=60 Win=65536 Len=0 TSval=835203654 TSecr=835203654
374	20.012709576	127.0.0.1	127.0.0.1	TCP	66	9999 → 51628 [FIN, ACK] Seq=60 Ack=21677497 Win=3112448 Len=0 TSval=835203666 TSecr=835203654
375	20.012734860	127.0.0.1	127.0.0.1	TCP	66	51628 → 9999 [ACK] Seq=21677497 Ack=61 Win=65536 Len=0 TSval=835203666 TSecr=835203666

Frame 371: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 9999, Dst Port: 51628, Seq: 41, Ack: 2

הרצה שנייה – 10% איבוד פאקטות

ניתן להבחין בבירור בהאטה משמעותית ביותר של שליחת הקבצים בהשוואה לחלק הראשון, בנוסף מבחינים כי הפער בין האלגוריתמים נהיה משמעותי יותר כאשר cubic מאשר reno

```
Stop sending the file

Time of receiving number 1 :
part 1: 0.22402334213256836 seconds
part 2: 0.003892660140991211 seconds

Time of receiving number 2 :
part 1: 0.004171133041381836 seconds
part 2: 0.2207174301147461 seconds

Time of receiving number 3 :
part 1: 0.2729053497314453 seconds
part 2: 0.0023365020751953125 seconds

Time of receiving number 4 :
part 1: 0.6241581439971924 seconds
part 2: 0.002256631851196289 seconds

Time of receiving number 5 :
part 1: 0.026320934295654297 seconds
part 2: 0.003447294235229492 seconds

Average of part 1: 0.23031578063964844 seconds
Average of part 2: 0.04653010368347168 seconds
Total Average: 0.13842294216156006 seconds

Close the connection..
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject$
```

ברור כי כל התהליך שתיארנו בהרצה ללא איבוד הפאקטות מתרחש גם כאן וניתן לזהות אותו באופן זהה, אלא שכאן המערכת צריכה להתמודד עם איבוד הפאקטות, נתבונן בהקלטות wireshark ונלמד מהם כיצד המערכת מגיבה לאיבוד הפאקטות

199	30.845418106	127.0.0.1	127.0.0.1	TCP	65549 → 38966 [ACK] Seq=6560726 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
200	30.845435636	127.0.0.1	127.0.0.1	TCP	66 9999 → 38966 [ACK] Seq=13 Ack=6634209 Win=3879840 Len=0 TSval=838772045 TSecr=838772045
201	30.845452282	127.0.0.1	127.0.0.1	TCP	65549 38966 → 9999 [ACK] Seq=6634209 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
202	30.845460084	127.0.0.1	127.0.0.1	TCP	66 9999 → 38966 [ACK] Seq=13 Ack=6699692 Win=3879840 Len=0 TSval=838772045 TSecr=838772045
203	30.845626806	127.0.0.1	127.0.0.1	TCP	66 → 9999 [ACK] Seq=6765175 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045
204	30.845631489	127.0.0.1	127.0.0.1	TCP	78 [TCP Window Update] 9999 → 38966 [ACK] Seq=13 Ack=6699692 Win=3112448 Len=0 TSval=838772045 TSecr=838772045 SLE=6765175 SR...
205	30.845642783	127.0.0.1	127.0.0.1	TCP	65549 38966 → 9999 [ACK] Seq=6830658 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
206	30.845644615	127.0.0.1	127.0.0.1	TCP	78 [TCP Dup ACK 202#1] 9999 → 38966 [ACK] Seq=13 Ack=6699692 Win=3112448 Len=0 TSval=838772045 TSecr=838772045 SLE=6765175 SR...
207	30.845661241	127.0.0.1	127.0.0.1	TCP	65549 38966 → 9999 [ACK] Seq=6896141 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
208	30.845663874	127.0.0.1	127.0.0.1	TCP	78 [TCP Dup ACK 202#2] 9999 → 38966 [ACK] Seq=13 Ack=6699692 Win=3112448 Len=0 TSval=838772045 TSecr=838772045 SLE=6765175 SR...
209	30.845683443	127.0.0.1	127.0.0.1	TCP	65549 38966 → 9999 [ACK] Seq=6961624 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
210	30.845701715	127.0.0.1	127.0.0.1	TCP	65549 [TCP Fast Retransmission] 38966 → 9999 [ACK] Seq=699692 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP s...
211	30.845714524	127.0.0.1	127.0.0.1	TCP	66 9999 → 38966 [ACK] Seq=13 Ack=702107 Win=2988864 Len=0 TSval=838772045 TSecr=838772045
212	30.845724684	127.0.0.1	127.0.0.1	TCP	65549 38966 → 9999 [ACK] Seq=702107 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
213	30.845727290	127.0.0.1	127.0.0.1	TCP	66 9999 → 38966 [ACK] Seq=13 Ack=7092590 Win=2948896 Len=0 TSval=838772045 TSecr=838772045
214	30.845737618	127.0.0.1	127.0.0.1	TCP	65549 38966 → 9999 [ACK] Seq=7092590 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
215	30.845748077	127.0.0.1	127.0.0.1	TCP	66 9999 → 38966 [ACK] Seq=13 Ack=7158073 Win=2914688 Len=0 TSval=838772045 TSecr=838772045
216	30.845758922	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 38966 → 9999 [ACK] Seq=7289039 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045
217	30.845753034	127.0.0.1	127.0.0.1	TCP	78 [TCP Dup ACK 202#3] 9999 → 38966 [ACK] Seq=13 Ack=7158073 Win=2914688 Len=0 TSval=838772045 TSecr=838772045 SLE=7289039 SR...
218	30.845761838	127.0.0.1	127.0.0.1	TCP	65549 38966 → 9999 [ACK] Seq=7354522 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
219	30.845764120	127.0.0.1	127.0.0.1	TCP	78 [TCP Dup ACK 202#4] 9999 → 38966 [ACK] Seq=13 Ack=7158073 Win=2914688 Len=0 TSval=838772045 TSecr=838772045 SLE=7289039 SR...
220	30.845775224	127.0.0.1	127.0.0.1	TCP	65549 [TCP Fast Retransmission] 38966 → 9999 [ACK] Seq=7158073 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP s...
221	30.845777764	127.0.0.1	127.0.0.1	TCP	78 9999 → 38966 [ACK] Seq=13 Ack=7223556 Win=2849280 Len=0 TSval=838772045 TSecr=838772045 SLE=7289039 SRE=7420905
222	30.845788110	127.0.0.1	127.0.0.1	TCP	65549 [TCP Retransmission] 38966 → 9999 [ACK] Seq=7223556 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045
223	30.845790058	127.0.0.1	127.0.0.1	TCP	66 9999 → 38966 [ACK] Seq=13 Ack=7420905 Win=2817152 Len=0 TSval=838772045 TSecr=838772045
224	30.845800888	127.0.0.1	127.0.0.1	TCP	65549 38966 → 9999 [ACK] Seq=7420905 Ack=13 Win=65536 Len=65483 TSval=838772045 TSecr=838772045 [TCP segment of a reassembled PD...
225	30.845804429	127.0.0.1	127.0.0.1	TCP	66 9999 → 38966 [ACK] Seq=13 Ack=7485488 Win=2783744 Len=0 TSval=838772045 TSecr=838772045

נבחין פה בפאקטות הבעייתיות ונבין את המשמעות שלהם:

"TCP previous segment not captured" הפאקטה הקודמת עדיין לא התקבלה ולכן receiver

מסמן sender לשלוח לו אותה שוב

"TCP Dup ACK" receiver שולח לsender שיידע כי ישנה פאקטה שלא הגיעה אליו, או שהיא אבדה

או שעדיין לא הגיעה (בתוספת x#y כאשר אמסמל את מספר הפאקטה וx סופר כמה פעמים כבר הוא התריע על זה)

"TCP out-of-order" הפאקטה לא הגיעה במיקום הנכון שלה פי סדר השליחה

"TCP retransmission" - השולח לא מקבל אישור מהיעד כעבור זמן מסוים (טיימאאוט מוגדר) ולכן מבין שהפאקטה לא הגיעה כמו שצריך ושולח אותה מחדש

"TCP fast retransmission" השולח מבין על פי סדר האישורים שהוא מקבל מהיעד כי יש פאקטות נאבדו בדרך ולכן שולח אותם שוב עוד בטרם נגמר הטיימאאוט שהוקצב לקבלת האישור

נבחין כי אכן שורה 203 מציינת כי פאקטה 202 לא הגיעה ולאחר מכן נשלחת שוב ושוב בקשה מהreceiver לsender (פאקטות 206,208) עד שהוא שולח אותה שוב בשורה 210, לאחר מכן קורה תהליך דומה עם פאקטה 215 שלא מגיעה ליעדה.

בצורה זו מתמודדת המערכת עם אובדן הפאקטות ומנסה להמשיך את התנהלות התכנית בצורה רגילה ולשלוח שוב את הפאקטות החסרות על מנת להשלים את אובדן המידע

ניתן כעת להבין מדוע לוקח יותר זמן לשליחת הקבצים, המערכת צריכה להקצות זמן ומשאבים למרדף אחרי הפאקטות שלא הגיעו ושליחתם מחדש ולכן זמן השליחה של הקבצים גדל

הרצה שלישית – 15% איבוד פאקטות

שוב, הזמנים הכלליים של שליחה הולכים ועולים (הממוצע עלה בכ0.3 שניות) כאשר cubic שומר על יתרון לעומת reno אך הפער מצטמצם כאשר ב10% איבוד הוא היה מהיר יותר מפי 4 ואילו ב15% הוא מהיר בערך פי 3.
נמשיך לעקוב מה יקרה ב20%

```
Stop sending the file

Time of receiving number 1 :
part 1: 0.28852009773254395 seconds
part 2: 0.06921982765197754 seconds

Time of receiving number 2 :
part 1: 0.21334266662597656 seconds
part 2: 0.447465181350708 seconds

Time of receiving number 3 :
part 1: 0.6975066661834717 seconds
part 2: 0.045142173767089844 seconds

Time of receiving number 4 :
part 1: 0.5262246131896973 seconds
part 2: 0.2357017993927002 seconds

Time of receiving number 5 :
part 1: 1.4934086799621582 seconds
part 2: 0.26207995414733887 seconds

Average of part 1: 0.6438005447387696 seconds
Average of part 2: 0.2119217872619629 seconds
Total Average: 0.4278611660003662 seconds

Close the connection..
(venv) ilan@ilan-Latitude-5420:~/PycharmProjects/pythonProject$
```

באופן זהה לאיבוד של 10% המערכת מתמודדת בצורה של שליחת התרעות על איבוד פאקטות ושליחה שלהם מחדש, אולם כמות הפעמים שתהליך זה קורה עלתה ואנו מגיעים למספרי שיא של כמות פאקטות בתהליך השליחה (פאקטות רבות נשלחות על מנת להתמודד עם אובדן הפאקטות)
כפי שניתן לראות, סגירת החיבור וסיום ההתקשרות מגיעות רק בפאקטה מספר 632 (לעומת סגירה בפאקטה מספר 375 בהרצה ללא איבודים)

618	213.613906592	127.0.0.1	127.0.0.1	TCP	65549	37496	→	9999	[ACK]	Seq=21474227	Ack=37	Win=65536	Len=65483	TSval=844132633	TSecr=844132633	[TCP segment of a reassembled P...	
619	213.613910228	127.0.0.1	127.0.0.1	TCP	65549	37496	→	9999	[ACK]	Seq=21539710	Ack=37	Win=65536	Len=65483	TSval=844132633	TSecr=844132633	[TCP segment of a reassembled P...	
620	213.613913893	127.0.0.1	127.0.0.1	TCP	65549	37496	→	9999	[ACK]	Seq=21695193	Ack=37	Win=65536	Len=65483	TSval=844132633	TSecr=844132633	[TCP segment of a reassembled P...	
621	213.614011385	127.0.0.1	127.0.0.1	TCP	66	9999	→	37496	[ACK]	Seq=37	Ack=21670676	Win=2684800	Len=0	TSval=844132633	TSecr=844132633		
622	213.829435288	127.0.0.1	127.0.0.1	TCP	6865	37496	→	9999	[PSH, ACK]	Seq=21670676	Ack=37	Win=65536	Len=6799	TSval=844132249	TSecr=844132633	[TCP segment of a reassembl...	
623	213.829559868	127.0.0.1	127.0.0.1	TCP	70	9999	→	37496	[PSH, ACK]	Seq=37	Ack=21677475	Win=3145728	Len=4	TSval=844132249	TSecr=844132249	[TCP segment of a reassembl...	
624	213.873485931	127.0.0.1	127.0.0.1	TCP	66	37496	→	9999	[ACK]	Seq=21677475	Ack=41	Win=65536	Len=0	TSval=844132293	TSecr=844132249		
625	215.391540248	127.0.0.1	127.0.0.1	TCP	87	37496	→	9999	[PSH, ACK]	Seq=21677475	Ack=41	Win=65536	Len=21	TSval=844133811	TSecr=844132249	[TCP segment of a reassembled...	
626	215.391626849	127.0.0.1	127.0.0.1	TCP	85	9999	→	37496	[PSH, ACK]	Seq=41	Ack=21677496	Win=3145728	Len=19	TSval=844133811	TSecr=844133811	[TCP segment of a reassembl...	
627	215.391636408	127.0.0.1	127.0.0.1	TCP	66	37496	→	9999	[ACK]	Seq=21677496	Ack=60	Win=65536	Len=0	TSval=844133811	TSecr=844133811		
628	215.391889124	127.0.0.1	127.0.0.1	TCP	66	37496	→	9999	[FIN, ACK]	Seq=21677496	Ack=60	Win=65536	Len=0	TSval=844133811	TSecr=844133811		
629	215.601489315	127.0.0.1	127.0.0.1	TCP	66	[TCP Retransmission]	37496	→	9999	[FIN, ACK]	Seq=21677496	Ack=60	Win=65536	Len=0	TSval=844134021	TSecr=844133811	
630	215.601419929	127.0.0.1	127.0.0.1	TCP	78	[TCP Previous segment not captured]	9999	→	37496	[ACK]	Seq=61	Ack=21677497	Win=3145728	Len=0	TSval=844134021	TSecr=8441340...	
631	215.625358086	127.0.0.1	127.0.0.1	TCP	66	[TCP Retransmission]	9999	→	37496	[FIN, ACK]	Seq=60	Ack=21677497	Win=3145728	Len=0	TSval=844134045	TSecr=844134021	
632	215.625373358	127.0.0.1	127.0.0.1	TCP	66	37496	→	9999	[ACK]	Seq=21677497	Ack=61	Win=65536	Len=0	TSval=844134045	TSecr=844134045		

הרצה רביעית – 20% איבוד פאקטות

הזמנים שוב גדלו ממש בכמעט 4.5 שניות בממוצע מאשר 15%, אולם קורה כאן מהפך גדול בו אלגוריתם reno נהייה לראשונה מהיר יותר ועוד בצורה משמעותית (בממוצע 2.5 שניות מהיר יותר)

```
Stop sending the file

Time of receiving number 1 :
part 1: 8.277481317520142 seconds
part 2: 5.599479913711548 seconds

Time of receiving number 2 :
part 1: 0.6230425834655762 seconds
part 2: 0.9399144649505615 seconds

Time of receiving number 3 :
part 1: 0.2369091510772705 seconds
part 2: 1.8524489402770996 seconds

Time of receiving number 4 :
part 1: 6.867160320281982 seconds
part 2: 2.511601209640503 seconds

Time of receiving number 5 :
part 1: 0.015921592712402344 seconds
part 2: 18.908395290374756 seconds

Average of part 1: 3.2041029930114746 seconds
Average of part 2: 5.962367963790894 seconds
Total Average: 4.583235478401184 seconds

Close the connection..
```

מבחינת הקלטת wireshark, נבחין כי הקו ממשיך מהאיבודים של 10%, 15% כאשר דרך הטיפול באיבוד פאקטות נשארת זהה, אולם כמות המקרים שאיבוד פאקטות קורה (וישירות מכך כמות הפעמים שהיה צריך לטפל באיבוד) עלתה משמעותית וניתן לראות זאת בצורה ישירה בכמות הפאקטות שהתהליך דרש (סגירת החיבור נעשתה רק בפאקטה 717!)

710	108.1639932900	127.0.0.1	127.0.0.1	TCP	66 39444 → 9999 [ACK] Seq=21677496 Ack=61 Win=65536 Len=0 TSval=845394739 TSecr=845394739
711	108.394503756	127.0.0.1	127.0.0.1	TCP	87 39444 → 9999 [PSH, ACK] Seq=21677475 Ack=41 Win=65536 Len=21 TSval=845394739 TSecr=845392529 [TCP segment of a reassembled...
712	108.394616302	127.0.0.1	127.0.0.1	TCP	85 9999 → 39444 [PSH, ACK] Seq=41 Ack=21677496 Win=3145728 Len=19 TSval=845394739 TSecr=845394739 [TCP segment of a reassembl...
713	108.394629943	127.0.0.1	127.0.0.1	TCP	66 39444 → 9999 [ACK] Seq=21677496 Ack=60 Win=65536 Len=0 TSval=845394739 TSecr=845394739
714	108.394674145	127.0.0.1	127.0.0.1	TCP	66 39444 → 9999 [FIN, ACK] Seq=21677496 Ack=60 Win=65536 Len=0 TSval=845394739 TSecr=845394739
715	108.396989159	127.0.0.1	127.0.0.1	TCP	66 9999 → 39444 [FIN, ACK] Seq=60 Ack=21677497 Win=3145728 Len=0 TSval=845394742 TSecr=845394739
716	108.776090155	127.0.0.1	127.0.0.1	TCP	66 [TCP Retransmission] 9999 → 39444 [FIN, ACK] Seq=60 Ack=21677497 Win=3145728 Len=0 TSval=845395121 TSecr=845394739
717	108.776107062	127.0.0.1	127.0.0.1	TCP	66 39444 → 9999 [ACK] Seq=21677497 Ack=61 Win=65536 Len=0 TSval=845395121 TSecr=845394742

מבדיקה יסודית בהקלטה, נבחין כי ב-20% איבוד לא נעשו בכלל פעולות TCP fast retransmission אלא רק TCP retransmission רגיל. ההגיון המסתבר הוא כי מרוב שהרבה פאקטות נאבדות אזי השולח לא מצליח להבין בצורה ברורה האם פאקטה מסוימת נאבדה רק לפי האישורים של הפאקטות שלאחריה ולכן הוא שולח אותה שוב רק במקרה של טיימאאוט

סיכום הנתונים שאספנו

	ללא איבוד	10% איבוד	15% איבוד	20% איבוד
חלק ראשון (reno)	0.00387 שניות	0.23031 שניות	0.64380 שניות	3.20410 שניות
חלק שני (cubic)	0.00302 שניות	0.04653 שניות	0.21192 שניות	5.96236 שניות
ממוצע	0.00344 שניות	0.13842 שניות	0.42786 שניות	4.58323 שניות

סיכום:

נראה כי התכנית הצליחה להתמודד בצורה טובה עם אובדן הפאקטות ולהשלים בצורה תקינה את שליחת הקבצים וסגירת החיבור גם באחוז איבוד גבוה יחסית כמו שהגדרנו.

מבחינת איכות האלגוריתמים, קיבלנו כי אלגוריתם cubic מהיר יותר כאשר התכנית פועלת בצורה תקינה וללא איבוד פאקטות משמעותי, אולם ברגע שהתכנית נהיית לא יציבה ומתאפיינת בכמות גדולה של איבוד פאקטות אזי אלגוריתם reno מתעלה עליו ונהיה טוב יותר ולכן עדיף במקרה כזה.