

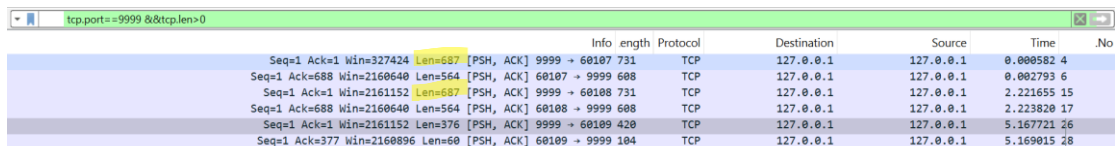
רשתות תקשורת מטלה 2

יוחנן קארו 207698218

אליסף דנה 315489534

צילומי מסך לפי הוראות:

(1) רק השרת בלי פרוקסי:



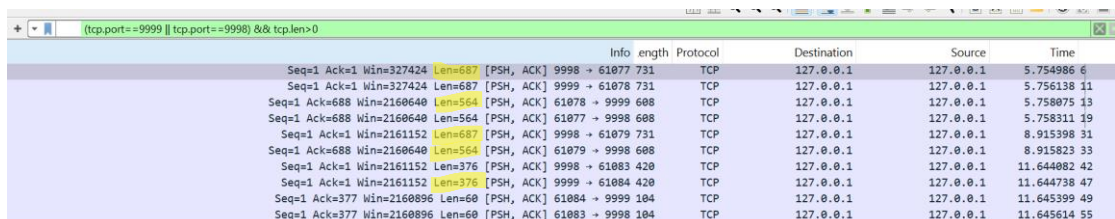
No.	Time	Source	Destination	Protocol	Length	Info
4	0.000582	127.0.0.1	127.0.0.1	TCP	731	Seq=1 Ack=1 Win=327424 Len=687 [PSH, ACK] 9999 → 60107
6	0.002793	127.0.0.1	127.0.0.1	TCP	608	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK] 60107 → 9999
15	2.221655	127.0.0.1	127.0.0.1	TCP	731	Seq=1 Ack=1 Win=2161152 Len=687 [PSH, ACK] 9999 → 60108
17	2.223820	127.0.0.1	127.0.0.1	TCP	608	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK] 60108 → 9999
26	5.167721	127.0.0.1	127.0.0.1	TCP	420	Seq=1 Ack=1 Win=2161152 Len=376 [PSH, ACK] 9999 → 60109
28	5.169015	127.0.0.1	127.0.0.1	TCP	104	Seq=1 Ack=377 Win=2160896 Len=60 [PSH, ACK] 60109 → 9999

אפשר לראות שהוא זה לפלט בטרמינל

```
Listening on 127.0.0.1:9999
Conection established with 127.0.0.1:60107
{127.0.0.1:60107} Got request of length 687 bytes
{127.0.0.1:60107} Sending response of length 564 bytes
{127.0.0.1:60107} Connection closed
Conection established with 127.0.0.1:60108
{127.0.0.1:60108} Got request of length 687 bytes
{127.0.0.1:60108} Sending response of length 564 bytes
{127.0.0.1:60108} Connection closed
Conection established with 127.0.0.1:60109
{127.0.0.1:60109} Got request of length 376 bytes
{127.0.0.1:60109} Sending response of length 60 bytes
{127.0.0.1:60109} Connection closed
```

ואפשר לזהות הבדל בין ההרצות הראשונות לשניה.

(2) הרצה של השרת וגם של הפרוקסי. עשר פקטות שיש בהן גם פעמיים בקשת catch לכל אחד מהפקודות החדשות שנשלחו



No.	Time	Source	Destination	Protocol	Length	Info
6	5.754986	127.0.0.1	127.0.0.1	TCP	731	Seq=1 Ack=1 Win=327424 Len=687 [PSH, ACK] 9998 → 61077
11	5.756138	127.0.0.1	127.0.0.1	TCP	731	Seq=1 Ack=1 Win=327424 Len=687 [PSH, ACK] 9999 → 61078
13	5.758075	127.0.0.1	127.0.0.1	TCP	608	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK] 61078 → 9999
19	5.758311	127.0.0.1	127.0.0.1	TCP	608	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK] 61077 → 9998
31	8.915398	127.0.0.1	127.0.0.1	TCP	731	Seq=1 Ack=1 Win=2161152 Len=687 [PSH, ACK] 9998 → 61079
33	8.915823	127.0.0.1	127.0.0.1	TCP	608	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK] 61079 → 9998
42	11.644082	127.0.0.1	127.0.0.1	TCP	420	Seq=1 Ack=1 Win=2161152 Len=376 [PSH, ACK] 9998 → 61083
47	11.644738	127.0.0.1	127.0.0.1	TCP	420	Seq=1 Ack=1 Win=2161152 Len=376 [PSH, ACK] 9999 → 61084
49	11.645399	127.0.0.1	127.0.0.1	TCP	104	Seq=1 Ack=377 Win=2160896 Len=60 [PSH, ACK] 61084 → 9999
55	11.645614	127.0.0.1	127.0.0.1	TCP	104	Seq=1 Ack=377 Win=2160896 Len=60 [PSH, ACK] 61083 → 9998

ברביועים הלבנים נבחין בהבדלים שבין המידע שכבר שמור ב catch לזה שלא-

cath miss / catch hit

```
Listening on 127.0.0.1:9998
{127.0.0.1:61077} Connected established
{127.0.0.1:61077} Got request of length 687 bytes
{127.0.0.1:61077} Cache miss, response cached ,server time remaining: inf, client time remaining: inf
{127.0.0.1:61077} Sending response of length 564 bytes
{127.0.0.1:61077} Connection closed
{127.0.0.1:61079} Connected established
{127.0.0.1:61079} Got request of length 687 bytes
{127.0.0.1:61079} Cache hit ,server time remaining: inf, client time remaining: inf
{127.0.0.1:61079} Sending response of length 564 bytes
{127.0.0.1:61079} Connection closed
{127.0.0.1:61083} Connected established
{127.0.0.1:61083} Got request of length 376 bytes
{127.0.0.1:61083} Cache miss, response cached ,server time remaining: inf, client time remaining: inf
{127.0.0.1:61083} Sending response of length 60 bytes
{127.0.0.1:61083} Connection closed
```

(3) הקלטה עם סגירה של servern :

tcp.port==9999 && tcp.len>0									
				Info	length	Protocol	Destination	Source	Time
Seq=1	Ack=1	Win=327424	Len=687	[PSH, ACK]	9998 → 61309	TCP	127.0.0.1	127.0.0.1	110.945729 49
Seq=1	Ack=1	Win=327424	Len=687	[PSH, ACK]	9999 → 61310	TCP	127.0.0.1	127.0.0.1	110.947522 54
Seq=1	Ack=688	Win=2160640	Len=564	[PSH, ACK]	61310 → 9999	TCP	127.0.0.1	127.0.0.1	110.949919 56
Seq=1	Ack=688	Win=2160640	Len=564	[PSH, ACK]	61309 → 9998	TCP	127.0.0.1	127.0.0.1	110.950238 61
Seq=1	Ack=1	Win=2161152	Len=687	[PSH, ACK]	9998 → 61312	TCP	127.0.0.1	127.0.0.1	126.460130 73
Seq=1	Ack=688	Win=2160640	Len=564	[PSH, ACK]	61312 → 9998	TCP	127.0.0.1	127.0.0.1	126.460552 75
Seq=1	Ack=1	Win=2161152	Len=548	[PSH, ACK]	9998 → 61318	TCP	127.0.0.1	127.0.0.1	149.467004 88
Seq=1	Ack=1	Win=2161152	Len=548	[PSH, ACK]	9999 → 61319	TCP	127.0.0.1	127.0.0.1	149.467697 93
Seq=1	Ack=549	Win=2160640	Len=196	[PSH, ACK]	61318 → 9998	TCP	127.0.0.1	127.0.0.1	149.483117 96

ב proxy אפשר לזהות שגיאה בשלב של הבקשה השניה שאינה שמורה ב catch עדיין

```
Listening on 127.0.0.1:9998
{127.0.0.1:61309} Connected established
{127.0.0.1:61309} Got request of length 687 bytes
{127.0.0.1:61309} Cache miss, response cached ,server time remaining: inf, client time remaining: inf
{127.0.0.1:61309} Sending response of length 564 bytes
{127.0.0.1:61309} Connection closed
{127.0.0.1:61312} Connected established
{127.0.0.1:61312} Got request of length 687 bytes
{127.0.0.1:61312} Cache hit ,server time remaining: inf, client time remaining: inf
{127.0.0.1:61312} Sending response of length 564 bytes
{127.0.0.1:61312} Connection closed
{127.0.0.1:61318} Connected established
{127.0.0.1:61318} Got request of length 548 bytes
Unexpected server error: [WinError 10054] An existing connection was forcibly closed by the remote host
{127.0.0.1:61318} Connection closed
```

שינויים שנעשו בקוד:

proxy:

```
proxy_socket.bind(proxy_address)
proxy_socket.listen(1) # מקסימום 1 חיבורים ממתינים בתור
```

bind : מחברת את ה proxy- לכתובת ולפורט שבהם הוא צריך להאזין.

Listen : מאפשר לשרת להמתין לחיבור אחד בתור.

```
client_socket, client_address = proxy_socket.accept()
```

קבלת חיבורים נכנסים מהלקוחות והחזרת ה socket של הלקוח ואת כתובתו.
רק כך הפרוקסי יכול להמשיך בעבודה.

```
data = client_socket.recv(api.BUFFER_SIZE)
```

השורה הזו קוראת נתונים שהתקבלו מהלקוח. Rcv משמש לקבלת הנתונים מה socket עם הגבלה על גודל המידע. (api.BUFFER_SIZE)
בלעדיו, ה proxy לא יקבל נתונים מלקוחות, ולא יוכל לעבד בקשות.

```
client_socket.sendall(response)
```

החזרת תשובה ללקוח. Sendall מבטיח שהכל ישלח.

```
proxy(proxy_address: (proxy_host, proxy_port), server_address: (server_host, server_port))
```

שליחת המידע ל הוסט ולפורט של הפרוקסי.

```
print(f"{client_prefix} Connection closed")
```

שליחת הודעה על סגירת החיבור

הסרור יהיה עם אותם שינויים אבל במקום לשלוח הלאה הוא יחזיר את:

```
server(host, port)
```

בקליינט: השינויים הם בניית התפריט בעיקר אבל בהקשר של תקשורת הוא שולח את הבקשה לפרוקסי:

```
# רשימת ביטויים לבחירה
while True:
    print("\nChoose a calculation function:")
    print("1: (sin(max(2, 3 * 4, 5, 6 * ((7 * 8) / 9), 10 / 11)) / 12) * 13")
    print("2: max(2, 3) + 3")
    print("3: 3 + ((4 * 2) / ((1 - 5) ** (2 ** 3)))")
    print("4: ((1 + 2) ** (3 * 4)) / (5 * 6)")
    print("5: --(1 + (2 + 3)) ** -4 + 5")
    print("6: max(2, 3 * 4, log(E), 6 * 7, 9 / 8)")
    print("0: Exit")

    choice = input("Enter your choice (0-6): ").strip()

    if choice == "0":
        print("Exiting...")
```

```
if choice == "0":
    print("Exiting...")
    break
elif choice == "1":
    expr = mul_b(div_b(sin_f(max_f(2, mul_b(3, 4), 5, mul_b(6, div_b(mul_b(7, 8), 9)), div_b(10, 11))),
elif choice == "2":
    expr = add_b(max_f(2, 3), 3)
elif choice == "3":
    expr = add_b(3, div_b(mul_b(4, 2), pow_b(sub_b(1, 5), pow_b(2, 3))))
elif choice == "4":
    expr = div_b(pow_b(add_b(1, 2), mul_b(3, 4)), mul_b(5, 6))
elif choice == "5":
    expr = neg_u(neg_u(pow_b(add_b(1, add_b(2, 3)), neg_u(add_b(4, 5))))
elif choice == "6":
    expr = max_f(2, mul_b(3, 4), log_f(e_c), mul_b(6, 7), div_b(9, 8))
else:
    print("Invalid choice. Please select again.")
```

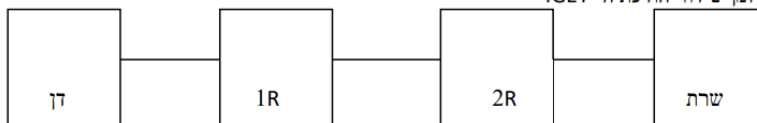
```
# ממשק למחשב
show_steps = input("Show steps? (y/n): ").strip().lower() == "y"
cache_result = input("Cache result? (y/n): ").strip().lower() == "y"
try:
    cache_control = int(input("Cache control (seconds, default 65535): ") or 2**16 - 1 )
except ValueError:
    print("Invalid input for cache control. Using default (65535).")
    cache_control = 65535

# שליחת הבקשה לשרת דרך הפרוקסי
client(server_address: (api.DEFAULT_PROXY_HOST, api.DEFAULT_PROXY_PORT), expr, show_steps, cache_result, cache_co
```

שאלות מהדף

שאלה 1

(1) נניח שדן בדפדפן האינטרנט ניגש לאתר www.google.com. גודל הדף שהדפדפן מוריד הוא F Kbyte ויש בתוכו X תמונות כאשר כל אחת מהתמונות היא בגודל $4F$ Kbyte. הדף והתמונות מאוחסנים באותו שרת. זמן העיבוד של כל אחד מהנתבים הוא d , קצב השידור R , אורך כל קו הוא x , מהירות התפשטות בקוים היא s ואפשר להזניח את זמן שידור הודעת ה-GET.



א. חשבו כמה זמן ידרוש לטעינת הדף כולל התמונות במחשבו של דן בכל אחד מהמקרים הבאים:

נרשום את הנתונים כ: $RTT : RTT = (3\frac{x}{s} + d)$

א. :

$$T(Non\ Persistent) = 2RTT + \frac{F}{R} + X\left(2RTT + \frac{4F}{R}\right) :$$

ב.

$$T(Persistent\ without\ pipelining) = 2RTT + \frac{F}{R} + X\left(RTT + \frac{4F}{R}\right) :$$

$$T(Persistent\ with\ pipelining) = 2RTT + \frac{F}{R} + X\left(\frac{4F}{R}\right) : \text{ג.}$$

הבדלים בין השיטות

HTTP non-persistent: שולח כל אובייקט בחיבור נפרד, בזבז זמן על פתיחה וסגירה של חיבורים, הכי איטי.

HTTP persistent: שולח אובייקטים באותו חיבור בצורה עוקבת, ללא בזבז זמן על חיבורים, ביצועים טובים יותר.

HTTP persistent with pipelining: שולח את כל האובייקטים במקביל באותו חיבור, הכי מהיר.

ג. הבדל בין HTTP GET-HTTP POST

הודעת GET עוברת ישירות ב URL ואין בה כמעט מידע מה שהופך את ההעברה שלה למהירה מאוד. לעומדת זאת ב POST עובר כל המידע מעמוד האינטרנט מה שהופך את זמן ההעברה שלה לאיטי משמעותית.

שאלה 2

- גודל הקובץ 1GByte
- מהירויות החיבור לרשת של המחשבים השונים:

המחשב	קצב העלאת נתונים	קצב הורדת נתונים
דן	10MByte/sec	10MByte/sec
10 מחבריו של דן	5MByte/sec	10MByte/sec
10 חברים נוספים	5MByte/sec	8MByte/sec
10 חברים נוספים	2MByte/sec	4MByte/sec

$$d_{cs} = \max \{ NF/u_s, F/\min(d_i) \}$$

1 של המשוואה ישאר תמיד אותו דבר:

$$10 \cdot 1000m / 100m = 100$$

כי השרת מעלה לקבוצה של 10 חברים בכל פעם באותו קצב העלאת

נחשב את שלב 2:

הקבוצה הראשונה של החברים מורידים מהסרבר בבת אחת ותופסים את כל קצב העלאת הנתונים של השרת במשך הזמן שלוקח להם כלומר:

$$1G/10m = 1024/10 = 102.4sec$$

נשתמש בנוסחא כדי לדעת איזה זמן לקחת. לוקחים את המקסימום בין 102.4 ל 100

ברגע שהקבוצה הראשונה סיימה הקבוצה השנייה מורידה את הנתונים במשך:

$$1g/8m = 1024/8 = 128sec$$

לוקחים את המקסימום בין 128 ל 100 לפי המשוואה

נשים לב שכולם ביחד לוקחים 80מגה ונשאר לשרת עוד 20 מגה פנויים בכל שנייה כך שהקבוצה השלישית מתחילה לנצל את זה כלומר :

הקבוצה השלישית מנצלת בכל שנייה 20 מגה בית כלומר 2 מגה בית לכל אחד במשך 128 שניות לכן בסוף התהליך הקבוצה השלישית הורידה $128\text{sec} \cdot 2\text{m} = 256\text{m}$ כלומר נשאר להם להוריד $768\text{m} = 1024 - 256$.

לכן לאחר מכן הקבוצה השלישית מורידה במשך $768\text{m}/4\text{m} = 192\text{sec}$

לוקחים את המקסימום בין 192 ל 100 לפי המשוואה

לכן סהכ הזמן הכולל הוא חיבור של כל הזמנים כלומר:

$$102.4 + 128 + 192 = 422.4$$

P2P

ברשת P2P דני מעלה את הקובץ פעם אחת וחבר אחד מוריד ובו זמנית משתף גם לחברים וכל חבר גם מוריד וגם מעלה יוצא מצב שכאילו כולם מורידים את הקובץ בו זמנית כי כל אחד מוריד ומעלה לכן נעשה את המקסימום מבין קצב העלאת של דני קצב ההורדה של החבר שהכי מעבד וקצב העלאת של כל החברים לפי כמות החברים

File Distribution Time: P2P

- ▶ **Server**
 - ▶ F/u_s (sec)
- ▶ **Client_i**
 - ▶ F/d_i (sec)
- ▶ **Aggregate Data**
 - ▶ NF (bits)

$$d_{p2p} = \max \{ F/u_s, F/\min(d_i), NF/(u_s + \sum u_i) \}$$

$$\text{Max}(1024/10, 1024/4, 30 \cdot 1024 / (10 + 20 \cdot 5 + 10 \cdot 2))$$

$$\text{Max}(102.4, 256, 236.307)$$

לכן התשובה היא 256

לכן עדיף להתשמש בP2P

סעיף ב

בסרבר הגורם המעכב הוא קצב ההורדה של החברים כי הם מורידים בקצב הרבה יותר נמוך מהקצב של השרת לכן כדי לתקן את זה היינו משנים את קצב ההורדה שיהיה זהה לקצב העלאה של השרת ואז היינו מקבלים

$$300 = 100 + 100 + 100 \text{ שניות סהכ}$$

ב $p2p$ הגורם השמפיע היה קצב ההורדה הכי נמוך כלומר של 10 החברים האחרונים לכן יש 2 אופציות:

במקרה הראשון שהוא פחות טוב (ביחס לשני) נעלה את קצב ההורדה של הקבוצה האחרונה לכדי מצב שקצב ההורדה יהיה שווה לקצב ההעלאה של כולם ביחד (כלומר החלק השלישי במשוואת המקסימום) כלומר נעלה את קצב ההורדה שלהם ל4.333 ואז נקבל $1024/4.333 = 236.307$ שזה קצב ההורדה הכללי כמו שצינו למעלה

המקרה השני והיותר טוב מחולק לשני מקרים

מקרה 1: נרצה שקצב ההורדה של הלקוח הכי איטי יהיה זהה לקצב ההעלאה של השרת כלומר 102.4 נעשה את זה עי כך שקצב ההורדה של הלקוח הכי איטי יהיה M10 לשנייה. ואז הגורם המעכב הוא קצב ההעלאה של כולם ביחד כלומר נקבל 236.307

במקרה השני נטפל גם בקצב ההעלאה של כולם ונשנה אותו לM10 לשנייה לכולם ואז נקבל $99.09 = (30 * 1024) / (10 + 10 * 30)$ בערך. קיבלנו ערך דומה פחות או יותר לקצב ההעלאה של השרת והזמן שנקבל יהיה קצב ההעלאה של השרת.

סעיף ג

ברשתות P2P יש כמה בעיות עיקריות שצריך להתמודד איתן:

חיפוש קובץ: קשה לדעת מי מחזיק את הקובץ שאתה מחפש. ב BitTorrent יש משהו שנקרא "טרקרים" שעוזר למחשב שלי להתחבר למחשב האחר שמחלק את אותו קובץ.

פיזור קובץ :

אם מחשב לא זמין אי אפשר להוריד את הקובץ ב BitTorrent הקובץ מחולק לחלקים קטנים שאפשר להוריד ממספר מקורות גם אם המקור שלי הפסיק לעבוד הקבצים שהורדתי נשמרים אצלי ואני יכול להמשיך את ההורדה ממקור אחר

מהירות הורדה

אם אין מספיק שותפים להורידה המהירות יכולה להיות איטית ב BITTORRENT אפשר להוריד חלקים שונים של הקובץ ממספר מקורות בו זמנית שזה עוזר להאיץ את ההורדה

בעיות בשיתוף :

לפעמים אנשים מורידים קבצים בלי לשתף ב BITTORRENT אין אפשרות כזו אם לא משתפים את מה שמורידים אין אופציה להמשיך את ההורדה

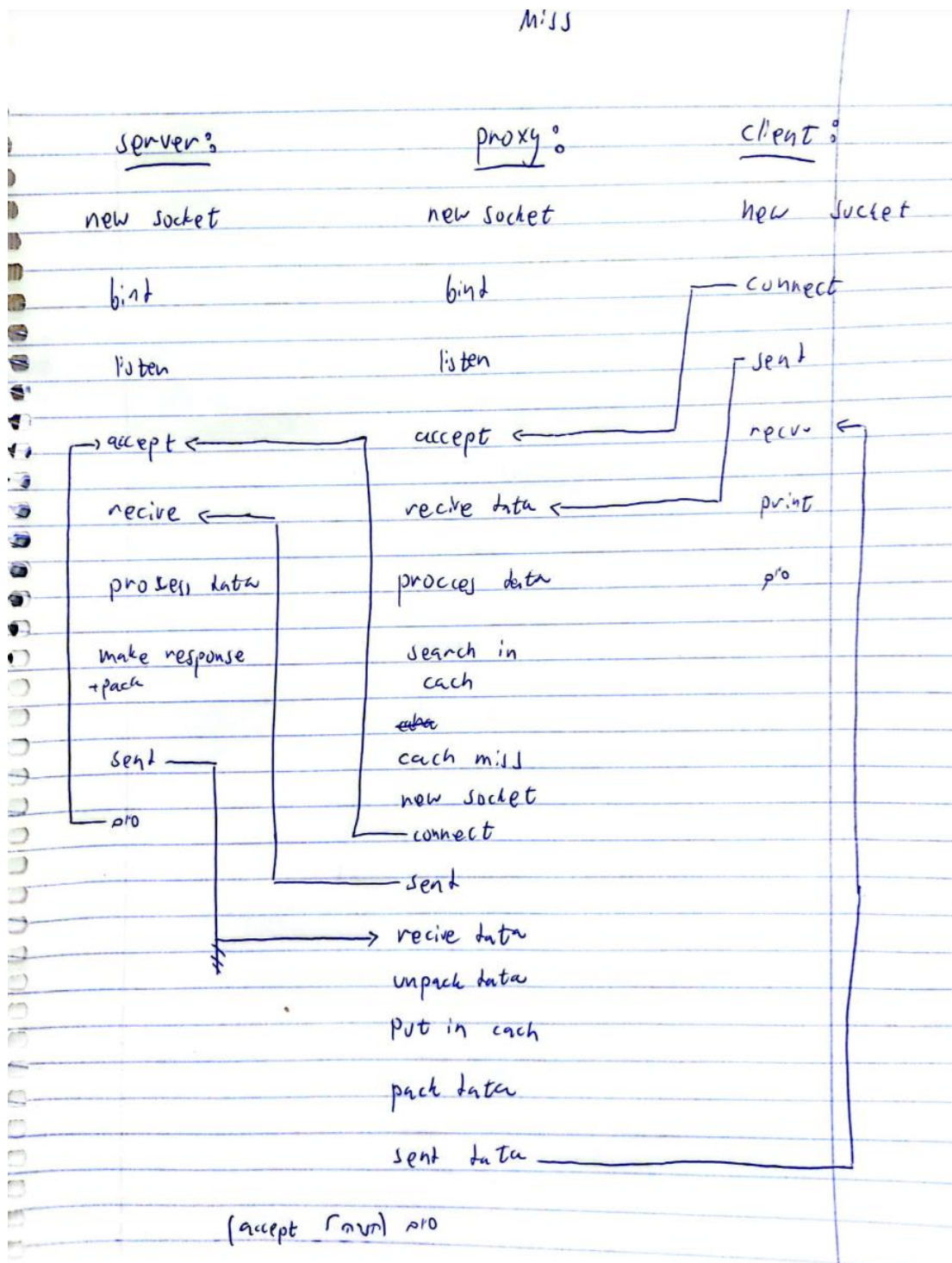
שמירה על פרטיות :

כל מחשב ברשת חשוף עם כתובת IP שלו ב BitTorrent אפשר להסתיר את הכתובת

ניהול עומס :

לפעמים יש עומס על הרשת כי יש המון תעבורה בין כל המחשבים. ב BitTorrent-יש עדיפות למי שמשתף יותר, וזה מונע עומס ברשת.

תרשים:



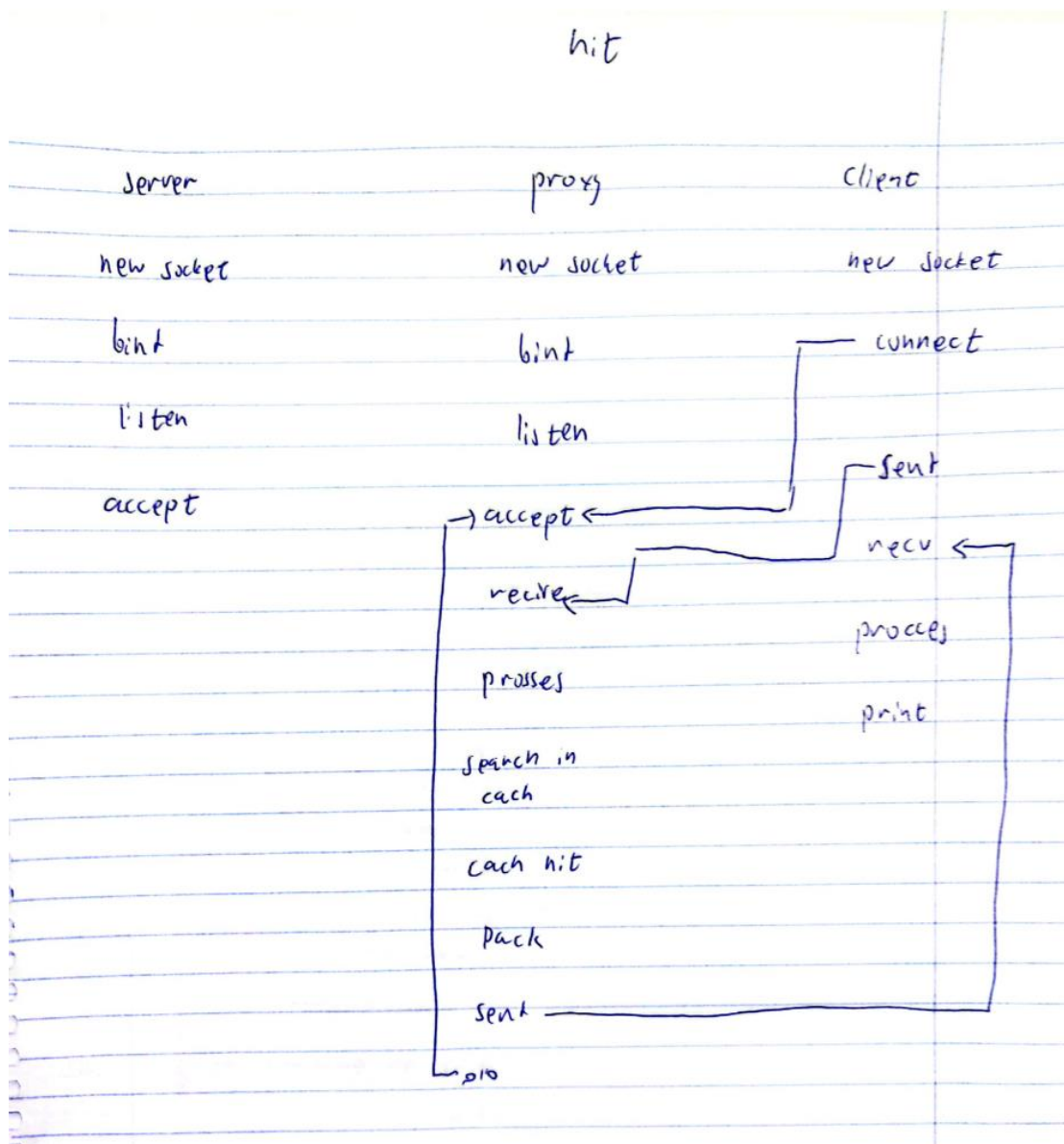
בהתחלה הקליינט פרוקסי וסרבר מופעלים כלומר יוצרים סוקט הסרבר עושה קישור לפורט ולIP של הסרבר והוא מאזין לבקשה אחת ומחכה שמישהו יתחבר אליו אותו כנל לפרוקסי

הקליינט מתחבר לפרוקסי (CONNECT) הפרוקסי מתחבר אליו (ACCEPT) ומתאפשרת העברת נתונים בין הפרוקסי לקליינט הקליינט שולח את הבקשה שלו (SEND) הפרוקסי מקבל את הבקשה (RECV) מכאן הפרוקסי עובד על הבקשה והקליינט מחכה לקבל תשובה:

הפרוקסי פותח את הבקשה כלומר ממיר אותה מביטים (PROCCES) ואז הוא מחפש בCACH . הוא לא ימצא את מה שהוא מבקש ב chach לכן הוא יפנה לסרבר (ויחכה לתשובה ממנו):

הוא פותח סוקט חדש והוא מבקש להתחבר לSERVER (CONNECT) הסרבר כמו שאמרנו בפתיחה מחכה לבקשה והוא מאשר אותה ברגע שהיא נשלחת אליו. לאחר מכן הפרוקסי שולח אליו את הנתונים שהקליינט ביקש ממנו (SEND) הסרבר מקבל את המידע (RECV) הוא ממיר את המידע מביטים (PROCESS) הוא בונה את התשובה שלו (MAKE RESPONSE) ואז הוא אורז את המידע לביטים (PACK) ושולח בחזרה לפרוקסי (SEND) בשלב זה החביר בין הסרבר לפרוקסי נסגר והסרבר חוזר להאזין לבקשות התחברות נוספות.

הפרוקסי מחכה לתשובה הוא מקבל אותה (RECV) בסיום התהליך של הסרבר הוא ממיר את המידע מביטים (UNPACK) ושם בCACH בהתאם לHEADER אורז שוב את המידע לביטים (PACK) ואז שולח אותו בחזרה לקליינט (SEND) לאחר מכן נסגר החיבור בין הקליינט לפרוקסי והפרוקסי חוזר להאזין שוב לעוד בקשות חיבור. כפי שאמרנו הקליינט מחכה לתשובה זו מגיעה מהפרוקסי לאחר כל התהליך והוא מדפיס את התשובה שקיבל ונסגר.



CHACH HIT

בהתחלה הקליינט פרוקסי וסרבר מופעלים כלומר יוצרים סוקט הסרבר עושה קישור לפורט ולקו של הסרבר והוא מאזין לבקשה אחת ומחכה שמישהו יתחבר אליו אותו כנל לפרוקסי

הקליינט בוחר את הבקשה שלו בMENU ומבקש להתחבר לשרת (CONNECT) השרת שמאזין מקבל את הבקשה (ACCEPT) לאחר מכן הקליינט מעביר את הבקשה שלו לשרת (SEND) השרת מקבל את המידע (RECVE) הוא ממיר אותו מביטים (PROCCES) ומחפש בCHACH לראות האם יש לי את המידע

במקרה זה יש לו את המידע לכן הוא אורז את המידע (PACK) ושולח אותו לקליינט (send) בשלב זה נסגר החיבור עם הקליינט והסרבר חוזר להאזין לעוד בקשות חיבור הקליינט מקבל את המידע (RECVE) ששלח לו הפרוקסי ממיר אותו מביטים (PROCCES) ולאחר מכן מדפיס אותו

שים לב שלאורך כל הזמן שסרבר מאזין אך לא היה בו שימוש