

רשתות תקשורת מטלה 2

יוחנן קארו 207698218

אליסף דנה 315489534

צלומי מסך לפי הוראות:

1) רק השרת בלי פרוקסִי:

tcp.port==9999 &&tcp.len>0						
	Info	length	Protocol	Destination	Source	Time
No						
1	Seq=1 Ack=1 Win=327424 Len=687 [PSH, ACK]	9999 → 60107 731	TCP	127.0.0.1	127.0.0.1	0.000582 4
2	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK]	60107 → 9999 608	TCP	127.0.0.1	127.0.0.1	0.002793 6
3	Seq=1 Ack=1 Win=2161152 Len=687 [PSH, ACK]	9999 → 60108 731	TCP	127.0.0.1	127.0.0.1	2.221655 15
4	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK]	60108 → 9999 608	TCP	127.0.0.1	127.0.0.1	2.223820 17
5	Seq=1 Ack=1 Win=2161152 Len=376 [PSH, ACK]	9999 → 60109 420	TCP	127.0.0.1	127.0.0.1	5.167721 26
6	Seq=1 Ack=377 Win=2160896 Len=60 [PSH, ACK]	60109 → 9999 184	TCP	127.0.0.1	127.0.0.1	5.169015 28

אפשר לראות שהוא זה לפולט בטרמינל

```
Listening on 127.0.0.1:9999
Connection 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
Connection 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
Connection 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
```

אפשר לזהות הבדל בין ההרצות הראשונות לשניה.

2) הרצה של השרת וגם של הפרוקסִי. עשר פקודות שיש בהן גם פעמיים בקשת catch לכל אחד מהפקודות החדשות שנשלחו

tcp.port==9999 tcp.port==9998 &&tcp.len>0						
	Info	length	Protocol	Destination	Source	Time
No						
1	Seq=1 Ack=1 Win=327424 Len=687 [PSH, ACK]	9998 → 61077 731	TCP	127.0.0.1	127.0.0.1	5.754986 6
2	Seq=1 Ack=1 Win=327424 Len=687 [PSH, ACK]	9998 → 61078 731	TCP	127.0.0.1	127.0.0.1	5.756138 11
3	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK]	61078 → 9998 608	TCP	127.0.0.1	127.0.0.1	5.758075 13
4	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK]	61077 → 9998 608	TCP	127.0.0.1	127.0.0.1	5.758311 19
5	Seq=1 Ack=1 Win=2161152 Len=687 [PSH, ACK]	9998 → 61079 731	TCP	127.0.0.1	127.0.0.1	8.915898 31
6	Seq=1 Ack=688 Win=2160640 Len=564 [PSH, ACK]	61079 → 9998 608	TCP	127.0.0.1	127.0.0.1	8.915823 33
7	Seq=1 Ack=1 Win=2161152 Len=376 [PSH, ACK]	9998 → 61083 420	TCP	127.0.0.1	127.0.0.1	11.644082 42
8	Seq=1 Ack=1 Win=2161152 Len=376 [PSH, ACK]	9999 → 61084 420	TCP	127.0.0.1	127.0.0.1	11.644078 47
9	Seq=1 Ack=377 Win=2160896 Len=60 [PSH, ACK]	61084 → 9999 184	TCP	127.0.0.1	127.0.0.1	11.645399 49
10	Seq=1 Ack=377 Win=2160896 Len=60 [PSH, ACK]	61083 → 9998 184	TCP	127.0.0.1	127.0.0.1	11.645614 55

בריבועים הלבנים נבחין בהבדלים שבין המידע שכבר שוכר שמור בזיה של א-

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

```

: הקלטה עם סגירה של ה-server (3)

	Info	Length	Protocol	Destination	Source	Time	No
	Seq=1 Ack=1 Win=327424 Len=687	[PSH, ACK] 9998 -> 61309 731	TCP	127.0.0.1	127.0.0.1	110.945729 49	
	Seq=1 Ack=1 Win=327424 Len=687	[PSH, ACK] 9999 + 61310 731	TCP	127.0.0.1	127.0.0.1	110.947522 54	
	Seq=1 Ack=688 Win=2160640 Len=564	[PSH, ACK] 61310 -> 9999 608	TCP	127.0.0.1	127.0.0.1	110.949919 56	
	Seq=1 Ack=688 Win=2160640 Len=564	[PSH, ACK] 61309 -> 9998 608	TCP	127.0.0.1	127.0.0.1	110.958238 61	
	Seq=1 Ack=1 Win=2161152 Len=687	[PSH, ACK] 9999 -> 61312 731	TCP	127.0.0.1	127.0.0.1	126.468138 73	
	Seq=1 Ack=1 Win=2161152 Len=687	[PSH, ACK] 61312 -> 9998 608	TCP	127.0.0.1	127.0.0.1	126.468552 75	
	Seq=1 Ack=1 Win=2161152 Len=548	[PSH, ACK] 9998 -> 61318 592	TCP	127.0.0.1	127.0.0.1	149.467004 88	
	Seq=1 Ack=1 Win=2161152 Len=548	[PSH, ACK] 61319 -> 9999 592	TCP	127.0.0.1	127.0.0.1	149.467697 93	
	Seq=1 Ack=549 Win=2160640 Len=196	[PSH, ACK] 61318 -> 9998 240	TCP	127.0.0.1	127.0.0.1	149.483117 96	

בproxy אפשר לזהות שגיאה בשלב של הבקשה השנייה שאינה שמורה ב"עד"

```

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 חיבורין ממוליכים בתוגן
```

bind : לחברת את ה proxy- לכתובת ולפורט שבהם הוא צריך להאזין.
Listen : מאפשר לשרת להמתין לחיבור אחד בתוגן.

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

קיבלה חיבורים נכנים מהלקוחות והחזרת ה socket של הלוקה ואת כתובתו.
רק כרגע יכול להמשיך בעבודה.

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

השורה הזאת קוראת נתונים שהתקבלו מהלוקה. Rec משמש לקבלת הנתונים מה socket עם הגבלה
על גודל המידע (api.BUFFER_SIZE).
בניגודיו, ה proxy לא יכול נתונים מלוקוחות, ולא יכול לעבוד בקשנות.

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

החזרת תשובה ללוקה. Sendall מבטיח שהכל ישלח.

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

שליחת המידע לホוט ולפורט של הפרוקסוי.

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

שליחת הודעה על סגירת החיבור

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

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

בקלינט: השינויים הם בנית התפריט בעיקר אבל בהקשר של תקשורת הוא שולח את הבקשה לפרויקט:

```
# בקlijint ביטויים ליבורנום
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.")

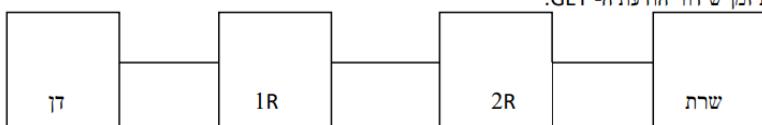
# מומלץ לזרוק
show_steps = input("Show steps? (y/n): ").strip().lower() == "y"
cache_result = input("Cache result? (y/n): ").strip().lower() == "y"
try:
    cache_control = int(input("Cache control (seconds, default 65535): ") or 2**16 -1 )
except ValueError:
    print("Invalid input for cache control.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

# בקlijint ביטויים ליבורנום
client( server_address=(api.DEFAULT_PROXY_HOST, api.DEFAULT_PROXY_PORT), expr, show_steps, cache_result, cache_co
```

שאלות מהדף

שאלה 1

(1) נניח שdone בדפדפן האינטרנט ניגש לאתר www.google.com. גודל הדרף שהדפדפן מורייד הוא F Kbyte ויחסו X תמונות כאשר כל אחת מהתמונות היא בגודל $4F$ Kbyte. הזמן והתמונות מאוחסנים באותו שרת. זמן העיבוד של כל אחד מהנתבים הוא d , קצב השידור R , אורך כל קו הוא s , מהירות התפשטות בקווים היא c ואפשר להזניח את זמן שידור הודעה - GET.



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

$$\text{RTT} : RTT = \left(3 \frac{x}{s} + d\right)$$

א :

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

ב.

$$T(\text{Persistant witout pipining}) = 2RTT + \frac{F}{R} + X \left(RTT + \frac{4F}{R}\right) :$$

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

הבדלים בין השיטות

HTTP non-persistent: כל אובייקט בחיבור נפרד, חיבור זמן עלפתיה וסירה של חיבורים, היכי אייט.

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

HTTP persistent with pipelining: אובייקטים במקביל באותו חיבור, היכי מהיר.

ג. הבדל בין HTTP POST ו-GET

הודעת get בעוברת שירות ב-*url* ואין בה כמעט מידע מה שהופך את ההעברה שלא למהירה מאוד. לעומת זאת ב-post עובר כל המידע מעמוד האינטרנט מה שהופך את זמן ההעברה שלא לאייטי משמעותית.

שאלה 2

- גודל הקובי 1GByte
- מהירות החיבור לשרת של המחשבים השונים:

המחשב	קצב העלאת נתונים	קצב הורדת נתונים
דנ	10MByte/sec	10MByte/sec
10 מחברי 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:

$$10*1000m/100m=100$$

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

ונחשב את שלב 2:

הקבוצה הראשונה של החברים מורידים מהсерבר בבת אחת ותופסים את כל קצב העלאה
הנתונים של השרת במשך הזמן שлокח להם כולם:

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

נשתמש בנוסחה כדי לדעת איזה זמן לוקח. לוקחים את המקיסום בין 102.4 ל 100

ברגע שהקבוצה הראשונה סיימה הקבוצה השנייה מורידה את הנתונים במשך:

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

לוקחים את המקיסום בין 128 ל 100 לפי המשוואה

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

הקובוצה השלישיית מנצלת בכל שנייה 2 מגה בית לכלי אחד במשך 128 שניות לכן בסוף התהליך הקובוצה השלישיית הורידה $2m=256$ sec * 128sec = 768sec נשאר להם להוריד $1024-256=768$ m

לכן לאחר מכן הקובוצה השלישיית מורידה במשך 768m / 4m = 192sec

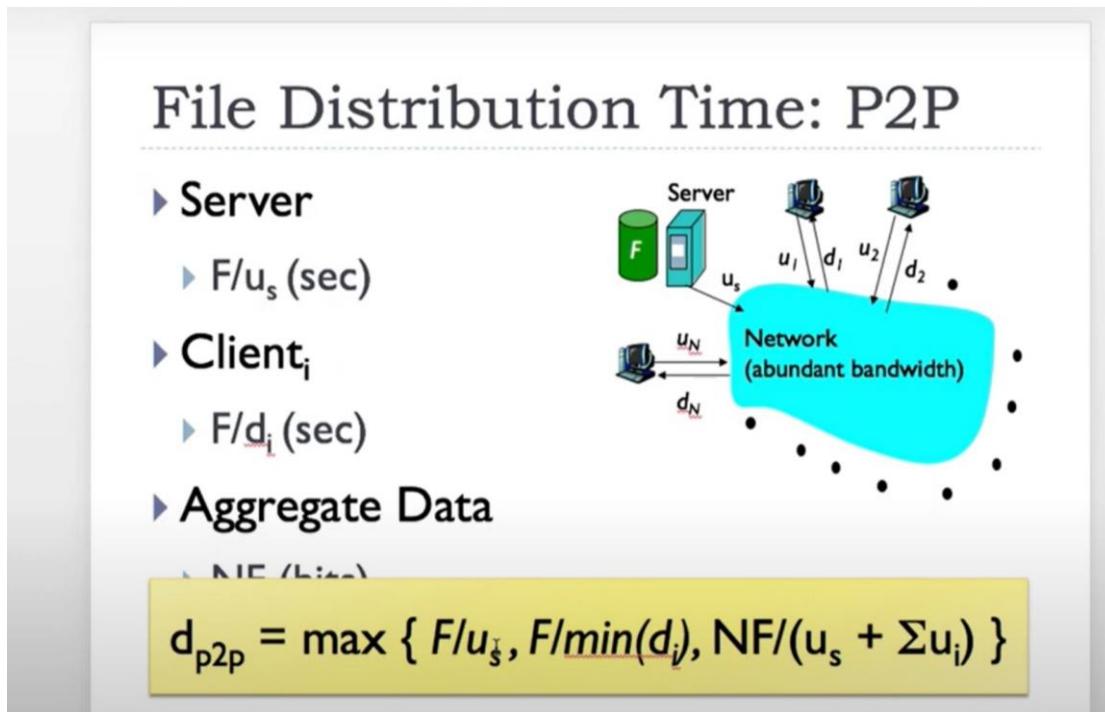
לוקחים את המיקסום בין 192 ל 100 לשונייה

לכן סהכ הזמן הכלול הוא חיבור של כל הזמןם כלומר:

$$102.4+128+192=422.4$$

P2P

ברשת P2P דני מעלה את הקובץ עם אחת וחבר אחד מוריד ובו זמינות משתף גם לחברים וכל חבר גם מוריד וגם מעלה יוצא מצב שכאיילו כולם מורידים את הקובץ בו זמינות כי כל אחד מוריד ומעלה שכן געשה את המיקסום מבין קצב העליה של דני קצב ההורדה של החבר שהכי מעכז וקצב העליה של כל החברים לפי כמות החברים



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

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

לכן התשובה היא 256

לכן עדיף להשתמש בק2P

סעיף ב

בסביבה הגורם המרכיב הוא קצב ההורדה של החברים כי הם מורידים בקצב הרובה יותר מאשר המרכיב של השירות ולכן כדי לתקן את זה היינו משנים את קצב ההורדה שייהי זהה לקצב העלה של השירות ואז היינו מקבלים

$$100+100=300 \text{ שניות סהכ}$$

במקרה השני השמאלי היה קצב ההורדה הכי נמוך לעומת 10 החברים האחרנים ולכן יש 2 אופציות:

במקרה הראשון שהוא פחות טוב (ביחס לשני) נעלם את קצב ההורדה של הקבוצה האחורונה לכך מציב שקצב ההורדה יהיה שווה לקצב העלה של כולם ביחד (כלומר החלק השלישי במשווהת המשמעות) כלומר עליה את קצב ההורדה שלהם ל-4.333 ואז נקבל $1024/4.333=236.307$

המקרה השני והיותר טוב מחלוקת לשני מקרים

מקרה 1: נרצה שקצב ההורדה של הלוקה הכי איטי יהיה זהה לקצב הלהלאה של השירות כלומר 102.4 נעשה את זה ע"י כך שקצב ההורדה של הלוקה הכי איטי יהיה M10 לשנייה. ואז הגורם המרכיב הוא קצב הלהלאה של כולם ביחד נקבל 236.307

במקרה השני נטפל גם בקצב העלה של כולם ונשנה אותו ל-10M לשנייה לכולם ואז נקבל $99.09 = (30 * 1024) / (10 + 30)$ בערך. קיבלנו ערך דומה פחות או יותר לקצב הלהלאה של השירות והזמן שנתקבל יהיה קצב הלהלאה של השירות.

סעיף ג

ברשותות P2P יש כמה בעיות עיקריות שצורך להתמודד איתן:

חיפוש קובץ: קשה לדעת מי מחזק את הקובץ שאתה מחפש. ב-*BitTorrent*-יש שהוא שנקרא "טרקים" שיעזר למחשב שלך לłączת החיבור שמחلك את אותו קובץ.

פיזור קובץ :

אם מחשב לא זמין אי אפשר להוריד את הקובץ ב-*BitTorrent*. הקובץ מחלוקת לחלקים קטנים שאפשר להוריד מספר מקורות גם אם המקור שלו הפסיק לעבוד הקבצים שהורדתי נשמרים אצל ואני יכול להמשיך את ההורדה ממקוור אחר

מהירות הורדה

אם אין מספיק שותפים להורידה המהירות יכולה להיות איטית ב-BITTORRENT אפשר להוריד חלקים שונים של הקובץ מספר מקורות בו זמן יותר 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 :

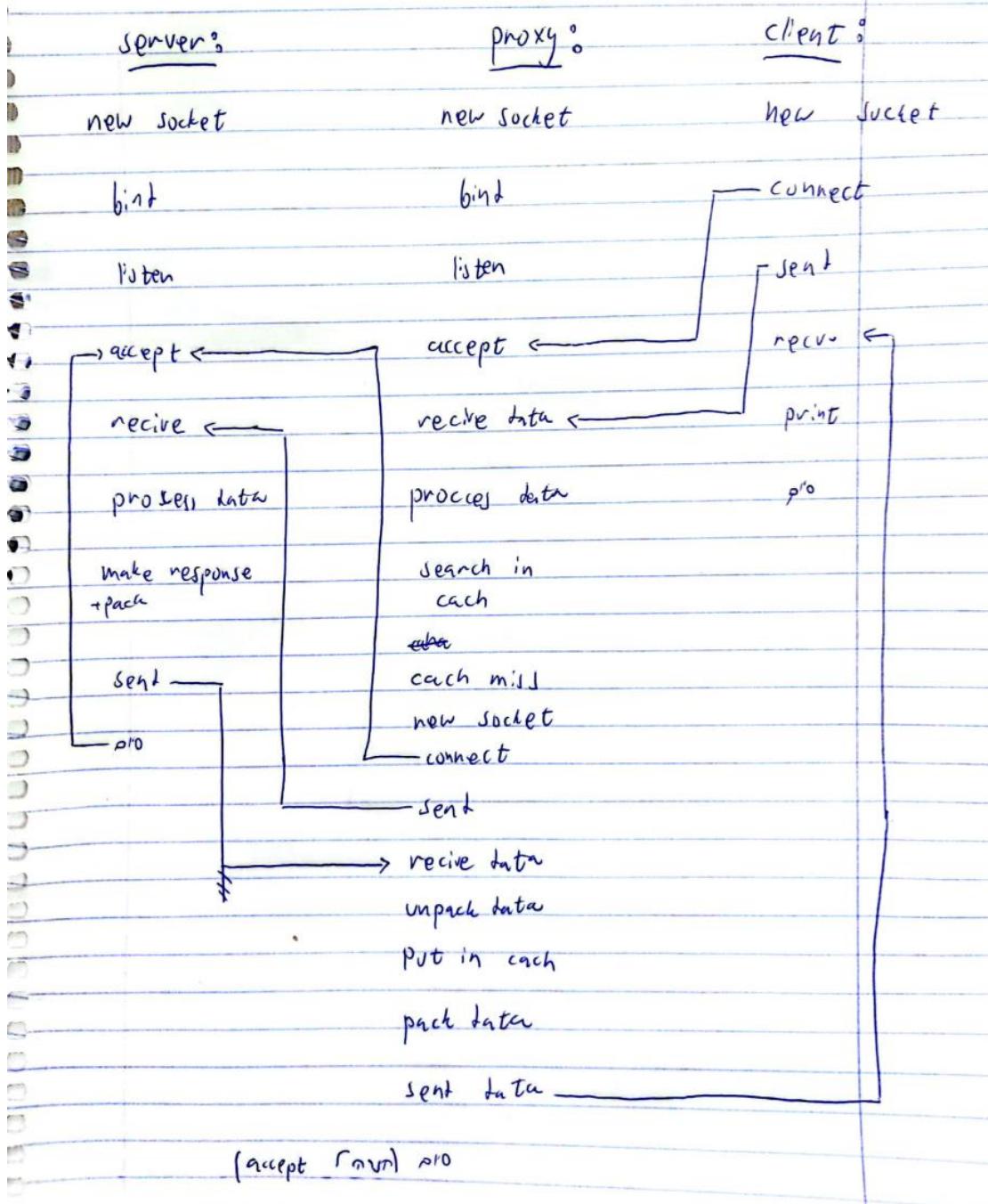
כל מחשב ברשת חשוף עם כתובת IP שלו ב-*BitTorrent* אפשר להסתייר את הכתובת

ניהול עומס :

לפעמים יש עומס על הרשות כי יש המון תעבורה בין כל המחשבים. ב BitTorrent-יש עדיפות למי שמשתף יותר, זהה מונע עומס ברשות.

תרשים:

MISS

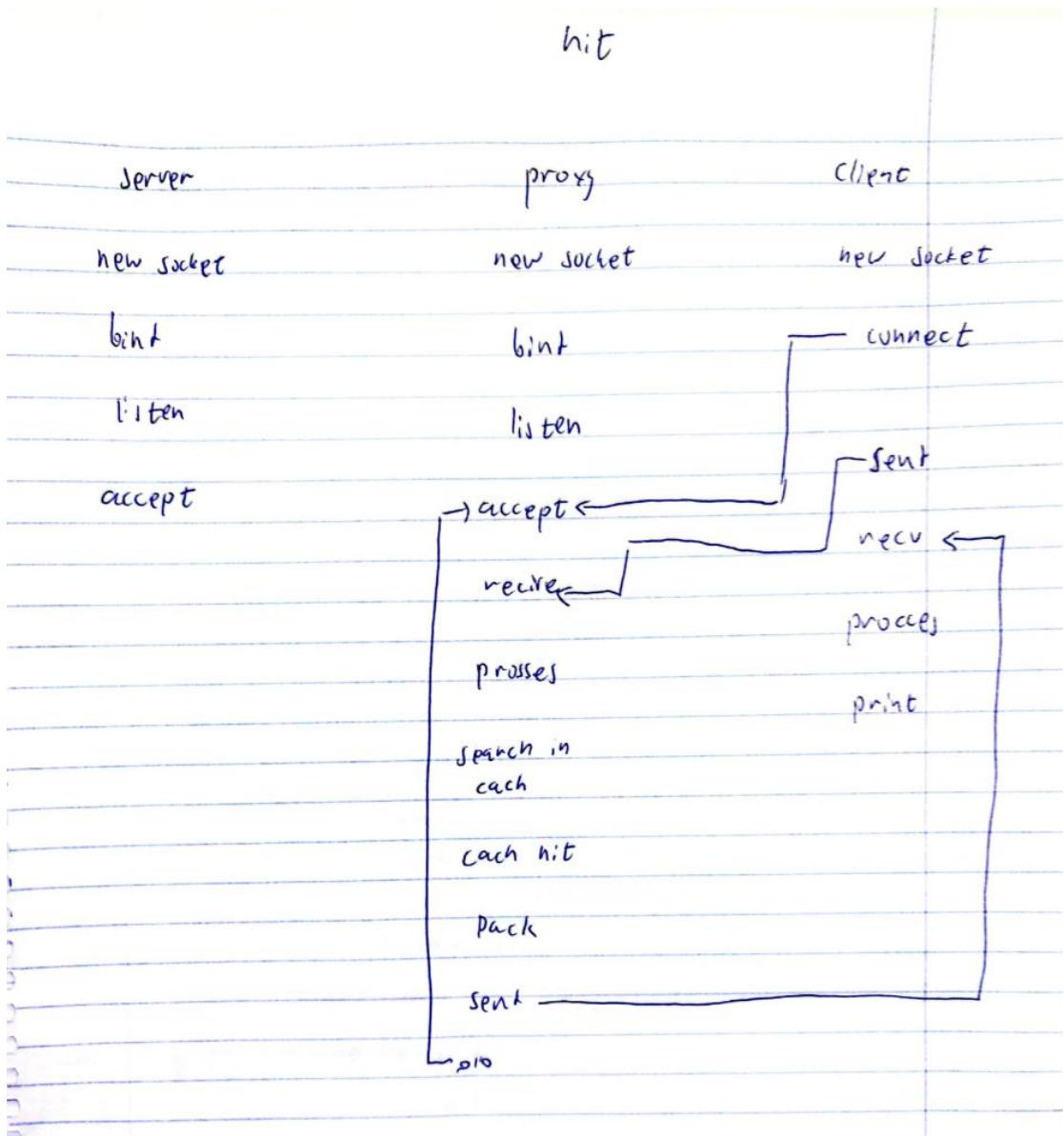


בהתחלת הקלינט פרוקס' וסרבר מופעלים כלומר יוצרים סוקט השרת עושה קישור לפורט ולקו של השרת והוא מאמין לבקשת אחת ומacha שמיישו יתחבר אליו אותו כNEL פרוקס' הקלינט מתחבר לפrox (CONNECT) הprox מתחבר אליו (ACCEPT) ומתאפשרת העברת נתונים בין הprox לקלינט הקלינט שולח את הבקשת שלו (SEND) הprox מקבל את הבקשת (RECV) מכון הprox עובד על הבקשת והקלינט מacha לקבל תשובה:

הפרויקט פותח את הבקשה כלומר ממיר אותה מביטים(PROCCES) זאת הוא מփש בCHASE . הוא לא ימצא את מה שהוא מבקש ב chach لكن הוא יפנה לשרת(ויחכה לתשובה ממנו):

הוא פותח סוקט חדש והוא מבקש להתחבר ל SERVER ל CONNECT (SERVER) השרת כמו שאמרנו בפתחה מחייב לבקשתו והוא מאשר אותה ברגע שהיא נשלחת אליו . לאחר מכן הפרויקט שלוח אליו את הנתונים שהקלינט ביקש ממנו (SEND) השרת מקבל את המידיע(RECV) הוא ממיר את המידע מביטים(PROCESS) והוא בונה את התשובה שלו(MAKE RESPONSE) וואז הוא אורץ את המידע לביטים(PACK) ושולח בחזרה לפרויקט(SEND) בשלב זה החיבור בין השרת לפרויקט נסגר והשרת חוזר להاذין לבקשתו התחרבות נוספת.

הפרויקט מחייב לבקשתו הוא מקבל אותה (RECV) בסיום התהליך של השרת הוא ממיר את המידע מביטים(UNPACK) ושם בהתאם ל HEADER CHACH שוב את המידע לביטים(PACK) וואז שלוח אותו בחזרה לקליינט(SEND) לאחר מכן נסגר החיבור בין הקלינט לפרויקט והפרויקט חוזר להاذין שוב לעוד בקשות חיבור . כפי שאמרנו הקלינט מחייב לבקשתו זו מגיעה מהפרויקט לאחר כל התהליך והוא מדפיס את התשובה שקיבל ונסגר .



CHACH HIT

בהתחלת הלקוח פרוקס וסרבר מופעלים כולם יוצרים Soket הסרבר עושה קישור לפורט ולוקן של הסרבר והוא מזין לבקשה אחת ומacha שמשה יתחבר אליו אותו כנהל פרוקס.

הלקוח בוחר את הבקשה שלו בMENU וambil לתחבר לשרת(CONNECT) השרת שמאזין מקבל את הבקשה(ACCEPT) ולאחר מכן הלקוח מעביר את הבקשה שלו לשרת(SEND) השרת מקבל את המידע(RECVE) הוא מmir אותו מביטים(PROCCES)

ומחפש בCLIENT לראות אם יש לי את המידע במקרה זה יש לו את המידע ולכן אורך את המידע(PACK) ושולח אותו ללקוח

(send) בשלב זה נסגר החיבור עם הלקוח וסרבר חוזר להזין לעוד בקשות חיבור הלקוח מקבל את המידע(RECVE) ששלח לו הפרוקס ממיר אותו מביטים(PROCES)

ולאחר מכן מדפיס אותו

שים לב שלאורך כל הזמן סרבר מזין אך לא היה בו שימוש