תומר סרוסי 209130269
בן כהן 208685784

מבוא לתקשורת מחשבים – תרגיל מעשי 1

1. שמות המגישים ות"ז

תומר סרוסי 209130269
בן כהן 208685784

2. הוראות שימוש והרצה

את הפרויקט כתבנו בשפת c. בקובץ המכווץ שלנו ישנם 4 קבצי קוד רלוונטיים, 2 קבצי הרצה, תיקייה בה הלוגים של הרצתנו, וקובץ ה-readme:
.server.c, server.h, client.c, client.h, sever.exe, client.exe, log files, readme.pdf

על מנת להריץ את התקשורת הדרושה בין הלקוח לשרת על אותו המחשב יש לבצע את הפעולות הבאות:

א. יש לקמפל את קוד השרת והלקוח באמצעות הפקודה הבאה:
gcc -g {server\client}.c -o {server\client}.exe -lwsock32
כאשר **{server\client}** נבחר בהתאם לקמפול קוד הסרבר או הלקוח.

ב. לאחר הקמפול יש להריץ את השרת עם הפרמטרים הנבחרים.

ג. לאחר הרצת השרת יש להריץ את הלקוח עם הפרמטרים הנבחרים, כאשר פרמטר ה-$serv\_ip$ יהיה "127.0.0.1" עבור חיבור מקומי.

3. תיאור הפרויקט ומבנה הקוד

בפרויקט זה מימשנו תור M/M/1 תוך כתיבת אמולציה פשוטה מבוססת TCP וזמן.
בנינו שני קבצי C ראשיים – קובץ client.c וקובץ server.c אשר מחוברים על ידי ערוץ TCP. *לכל אחד מן הקבצים, קיים גם קובץ header.

<u>לקוח</u>

- הלקוח מייצר ג'ובים בתהליך הסתברותי (פואסוני) – מימשנו זאת ע"י פונקציה בשם exp_wait_time אשר מקבלת את המשתנה μ ומגרילה את הזמן כדרוש.

```
double exp_wait_time(double mu)
{
    double u = (double)rand()/(double)RAND_MAX;
    return -1000*log(u)/mu;
}
```

תומר סרוסי 209130269
בן כהן 208685784

- כאשר ג'וב נוצר אצל הלקוח, נשלחת שורה אל השרת דרך ערוץ ה TCP. את יצירת הג'וב מימשנו באמצעות פונקציה בשם *job_creator* אשר יוצרת את הודעת הג'וב בפורמט הדרוש באמצעות *thread* ומחכה לחיווי חזרה מהשרת אם הג'וב הצליח (מקבל חיווי *'s'* מהשרת) או נכשל (מקבל חיווי *'f'* מהשרת)

```
DWORD WINAPI job_creator(void* data){
    double t = *(double*)data, creation_time, finish_time;
    int data_size;
    char job[MAX_LEN], ret;
    data_size = recv(connection_socket, &job, MAX_LEN, 0);
    sscanf(job, "%c%s", &ret, job);
    creation_time = atof(job);
    finish_time = (double)(clock())/CLOCKS_PER_SEC;
    if (ret == 'f')
    {
        WaitForSingleObject(mutex, INFINITE);
        total_drops++;
        ReleaseMutex(mutex);
        fprintf(tmp_file, "%f %f %f\n", creation_time, 0, 0);
    }
    else{
        fprintf(tmp_file, "%f %f %f\n", creation_time, finish_time, (finish_time - creation_time));
    }
    WaitForSingleObject(mutex, INFINITE);
    total_threads--;
    ReleaseMutex(mutex);
    ExitThread(0);
}
```

- מימשנו זאת באמצעות הזמן שהגרלנו והוסבר לעיל, תוך איטרציה בלולאה הבאה: לאחריה מתקבל shutdown והחיבור נסגר.

```
while ((double)(clock())/CLOCKS_PER_SEC <= T)
{
    sprintf(job, "%f\n", (double)(clock())/CLOCKS_PER_SEC);
    data_size = send(connection_socket, &job, (int)strlen(&job), 0);
    total_packets++;
    WaitForSingleObject(mutex, INFINITE);
    total_threads++;
    ReleaseMutex(mutex);
    CreateThread(NULL, 0, job_creator, &t, 0, NULL);
    t = exp_wait_time(lambda);
    Sleep(t);
}
```

- הלקוח ייצר יומנים (logs) בהם ירשמו זמני השרות של החבילות, תפוסת התור, ומספר חבילות שנזרקו. מימשנו זאת באמצעות שני קבצים – *tmp_file* ו*log_file*. כאשר *tmp_file* רושם את המידע בפורמט הדרוש עבור כל ג'וב כאשר הוא נוצר בזמן אמת ולאחר סיום כל האיטרציות, תוכן זה מועתק לקובץ הסופי *log_file* בו השורה הראשונה היא בפורמט המתואר.

<u>שרת</u>

- השרת משרת את הג'ובים בזמן ריצה הסתברותי (התפלגות מעריכית). מימשנו זאת באמצעות הפונקציה exp_wait_time בדומה למימוש בלקוח, רק שכעת מקבלת את המשתנה λ

תומר סרוסי 209130269
בן כהן 208685784

- כאשר ג'וב מגיע אל השרת, הוא מוכנס לתור FIFO (אם יש מקום) כאשר בכל רגע נתון, הג'וב הראשון בתור מקבל שרות. מימשנו את קבלת הג'ובים באמצעות חוט נוסף, כאשר הטיפול נעשה באמצעות מתודה בשם job_handler. כמו כן, נעזרנו במימוש של מנעולים (mutex) על מנת למנוע גישה בו-זמנית של שני החוטים לתור.

```
DWORD WINAPI job_handler(void* data){
    Queue* q = (Queue*)data;
    double t;
    char cur_job[MAX_LEN], file_name[20], *tmp, ch;
    FILE *log_file;
    while (!is_closed || q->size != 0)
    {
        WaitForSingleObject(mutex, INFINITE);
        tmp = dequeue(q);
        fprintf(tmp_file, "%f %d\n", (double)(clock())/CLOCKS_PER_SEC, q->size);
        ReleaseMutex(mutex);
        t = exp_wait_time(mu);
        Sleep(t);
        sprintf(cur_job, "%c%s", 's', tmp);
        send(connection_socket, cur_job, strlen(cur_job), 0);
    }
    fclose(tmp_file);
    tmp_file = fopen("server_tmp.log", "r");
    sprintf(file_name, "server_%d.log", run_id-1);
    log_file = fopen(file_name, "w");
    fprintf(log_file, "server_%d.log: seed=%d, mu=%f, QSize=%d\n", run_id-1, seed, mu, QSize);
    while((ch = fgetc(tmp_file)) != EOF)
        fputc(ch, log_file);
    fclose(tmp_file);
    fclose(log_file);
    remove("server_tmp.log");
    ExitThread(0);
}
```

את מימוש התור באמצעות struct של FIFO אשר מופיע בקובץ server.h ובאמצעות המתודות enqueue ו-dequeue להכנסת/הוצאת איבר מהתור.

```
typedef struct q_node {
    char* value;
    struct q_node* next;
} Q_node;

typedef struct queue {
    Q_node* header;
    Q_node* last;
    int size;
} Queue;

void enqueue(Queue* q, char* value);
char* dequeue(Queue* q);
```

- כאשר שרות של ג'וב מסתיים, נשלחת הודעה מהשרת חזרה ללקוח ובמידה ומגיע ג'וב כאשר התור מלא, הג'וב נזרק ונשלחת הודעה מתאימה מהשרת ללקוח. מימשנו זאת באמצעות שליחת המידע על ערוץ הTCP בתוספת האות 'f' לכישלון או 's' להצלחה.
- את מימוש הלוגים של השרת מימשנו בדומה למימוש בלקוח.

מקורות שנעזרנו בהם:

מתודה להעתקת מידע מקובץ אחד לאחר:

https://forgetcode.com/c/577-copy-one-file-to-another-file

יצירת חוטים בWindows:

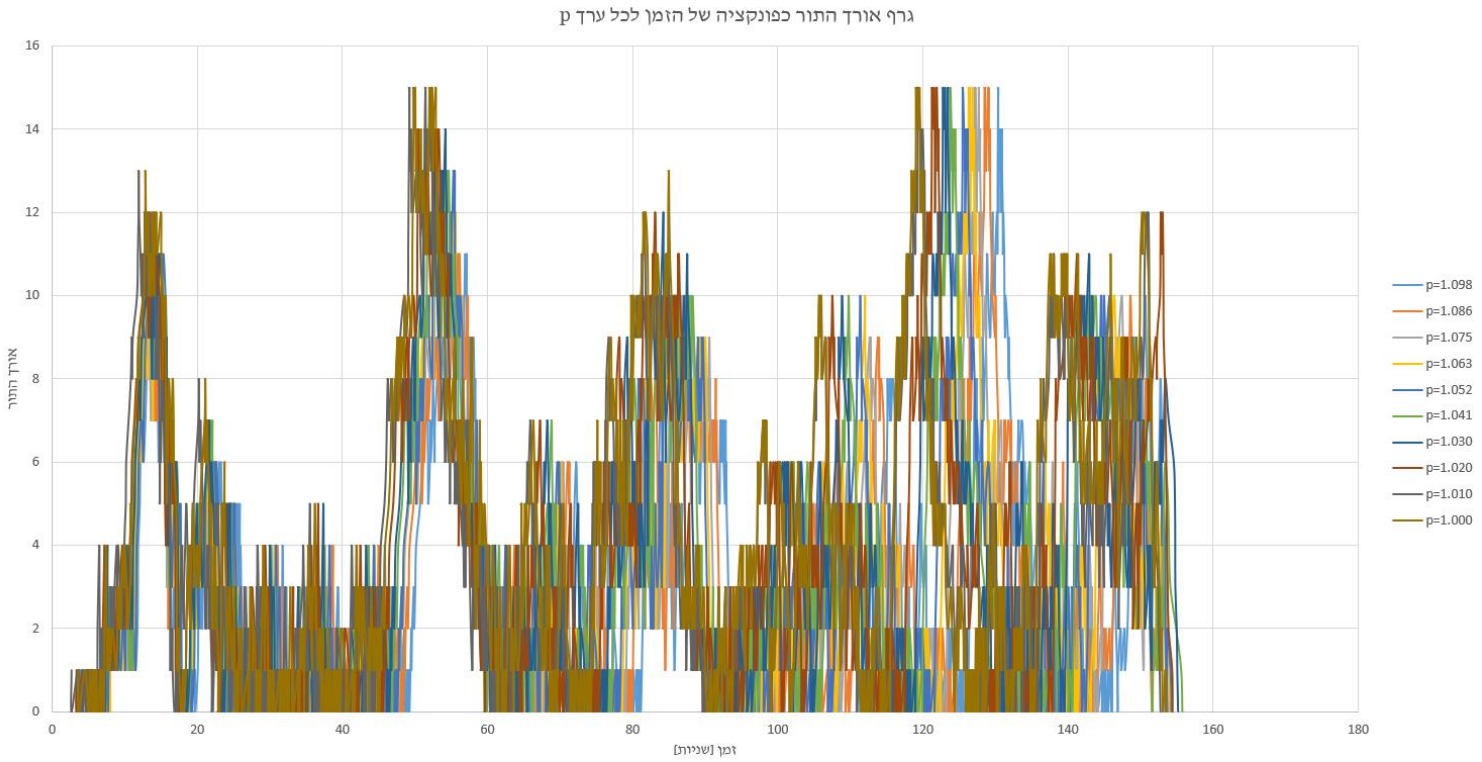
https://riptutorial.com/winapi/example/13881/create-a-new-thread

וכללי בספרייה windows.h למימוש המנעולים והחוטים בקוד שלנו.

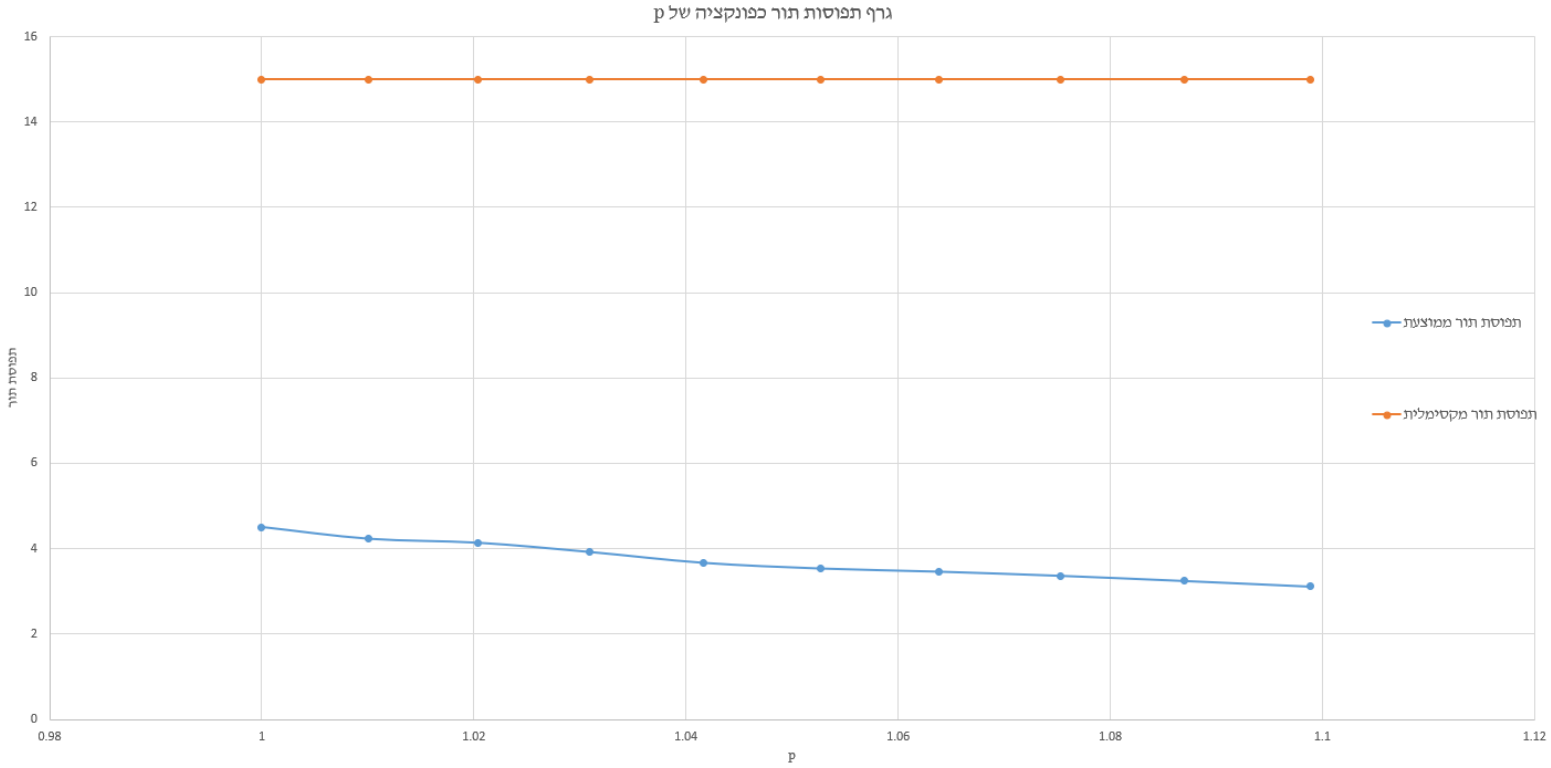
תומר סרוסי 209130269
בן כהן 208685784

הניסוי:

א.

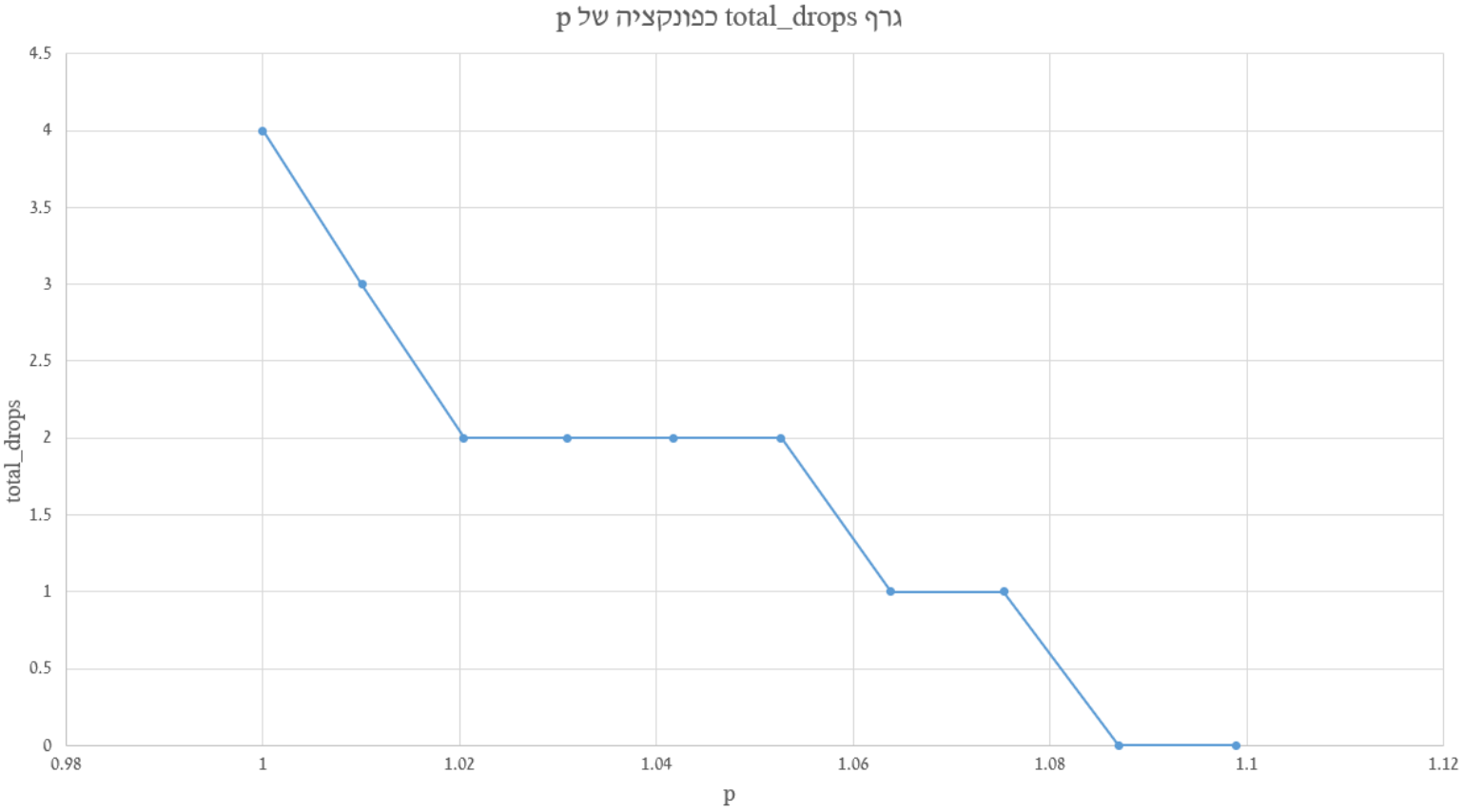


ב.

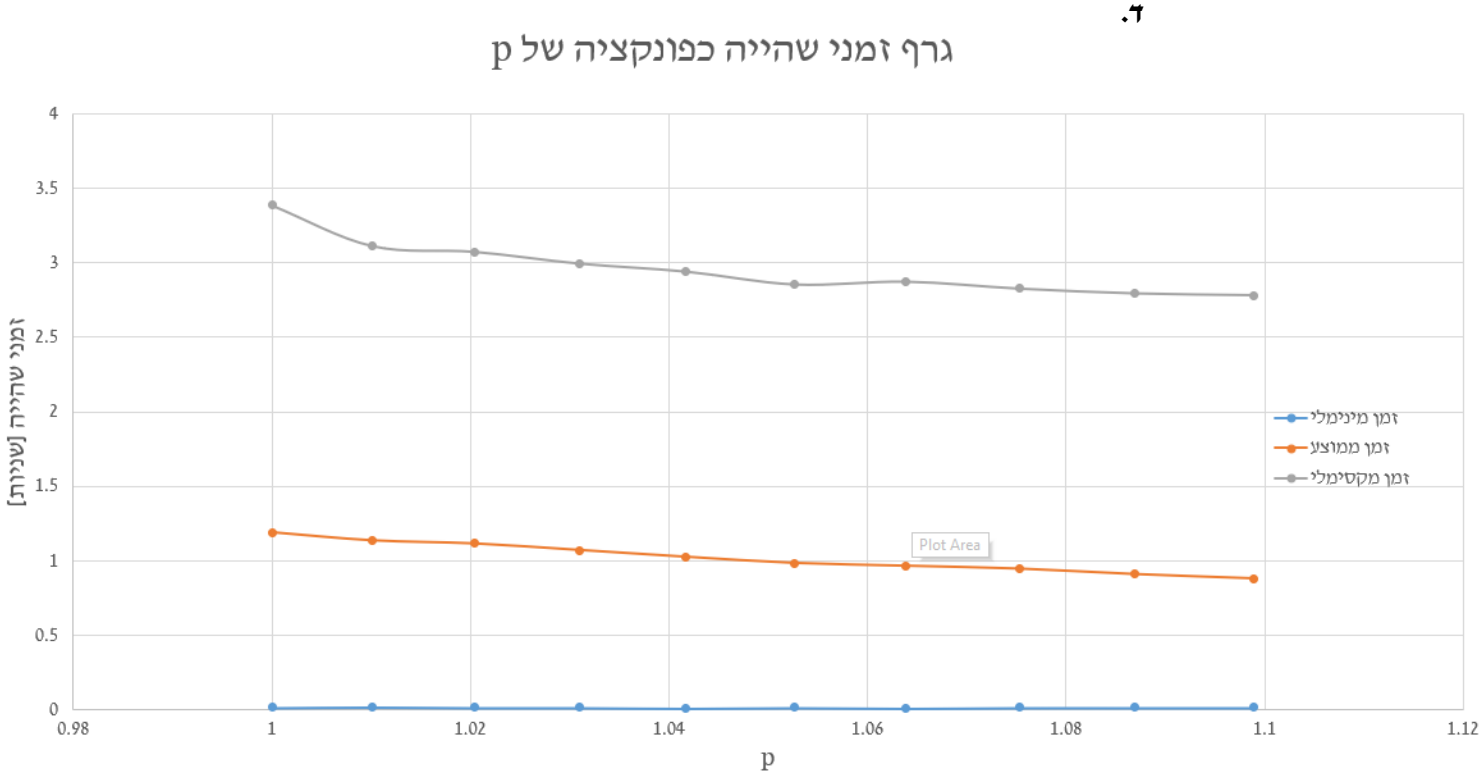


תומר סרוסי 209130269
בן כהן 208685784

ג.



ד.



תומר סרוסי 209130269
בן כהן 208685784

4. באגים ידועים ומגבלות שימוש:

בפרמטר ה-run_id של הלקוח ושל השרת לא ניתן לתת ערך 0. מגבלת שימוש זו הינה בהתאם להגדרות הפרמטרים שערך זה צריך להיות שלם חיובי. אולם, במהלך הניסוי נדרש מאיתנו לבצע הרצה עם ערך 0, ועל כן ראינו צורך לציין את מגבלה זו.