גם בחלק A וגם בחלק B עבדנו עם IPv4

Part A

כדי להגדיר לו עם איזה congestion control הוא יעבוד לפי הפרמטר שהוא קיבל מהשורת פקודה, אפשר להשתמש בsetsockopt כדי להגדיר לו:

```
// define the congestion control algorithm used by a socket
if (setsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, algo, strlen(algo)) < 0) {
  perror("setsockopt");
  return 1;
}
```

שאר המימוש מוסבר באופן מפורט בקוד.

## Part B:

## הסבר על המימוש שלנו ל RUDP :

סוקט RUDP במימוש שלנו הוא סוקט שדרכו ניתן לשלוח ולקבל מידע בצורה אמינה על גבי UDP
המידע הנשלח דרך סוקט UDP הוא struct הבנויי בצורה הבאה:

```
typedef struct RUDP_flags {
  unsigned int SYN : 1;
  unsigned int ACK : 1;
  unsigned int DATA : 1;
  unsigned int FIN : 1;
} RUDP_flags;

typedef struct _RUDP {
  RUDP_flags flags;
  int seq_num;
  int checksum;
  int length;  // the length of the data
  char data[WINDOW_MAX_SIZE];
} RUDP;
```

בעזרת מבנה זה כל חבילה הנשלחת על גבי UDP מכילה header אשר מאפיין את סוג החבילה, מכיל נתונים כמו checksum ו sep_num בכדי לאמת את מהימנות ושלמות החבילה, וכמובן את המידע הנשלח.

הרעיון הכללי הוא ששליחת אישורים וכל צעד אחר לטובת מהימנות הפרוטוקול ממומש בתוך ה API.

לחיצת יד מתבצעת ע"י שליחת חבילה המסומנת כחבילת "פתיחת קשר" (flags.SYN =1)
והמתנה לקבלת אישור המסומן כחבילת אישור "יצירת קשר" ( flags.SYN=1 , flags.ACK=1 )

הערה: לאחר מספר נסיונות מוגדר של לחיצת יד ללא מענה מהצד השני הפונקציה תפסיק לנסות ותחזיר שגיאה.

לאחר לחיצת היד, שליחת קובץ תתבצע בשיטת STOPandWAIT , הפונקציה תחלק את הקובץ לחבילות data לפי גודל החלון שהגדרנו בקוד, תשלח את החבילה הראשונה, ולא תשלח את החבילה הבאה עד לקבלת אישור ( flags.DATA=1 , flags.ACK=1 ) , אם לא מתקבל אישור לאחר זמן TIMEOUT שהגדרנו מראש, פונקצית השלחה תשלח את הקובץ שוב.

פונקצית קבלת המידע שמימשנו, שולחת אישור לשולח עבור כל סוג חבילה בהתאמה, מכניסה את המידע למקום המתאים ומחזירה את הערכים הבאים:

אם אירעה שגיאה: -1

אם הגיעה הודעת יצירת קשר\ חבילה עם data שכבר הגיע בעבר: 0

חבילה המכילה מידע חדש לקליטה: 1

חבילה המכילה מידע חדש שהוא החלק האחרון בכלל הקובץ שנשלח: 2

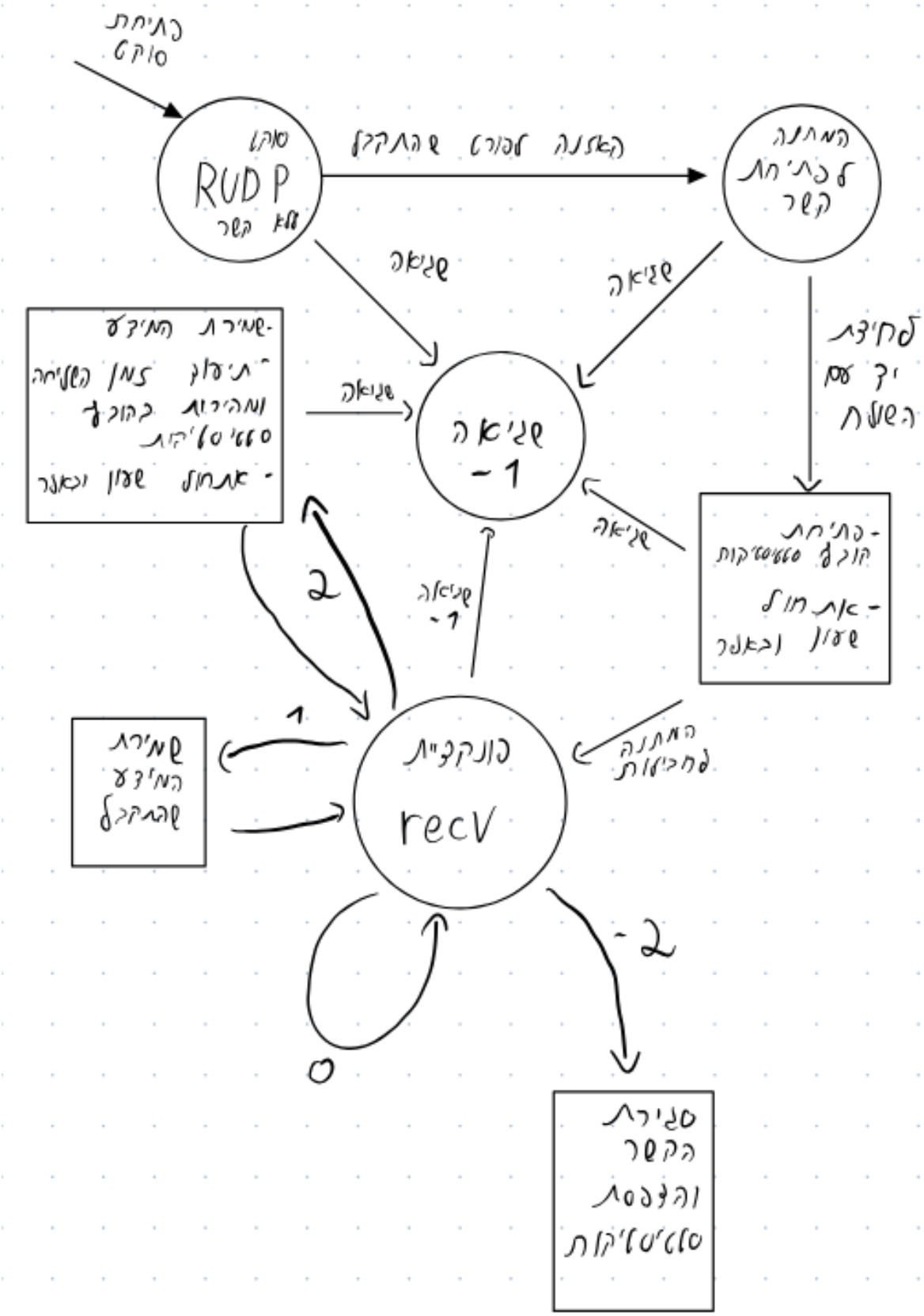
בקשה לסגירת קשר: -2

במידה ונתקבלה בקשה לסגירת הקשר, פונקציית קבלת המידע תשלח אישור ותחכה זמן מה בכדי לוודא שהאישור אכן הגיע ליעדו, ושלא נשלחה הודעת סגירה בשנית.

התמודדות עם TIMEOUT התבצעה בשני דרכים, הגדרת הסוקט ישירות כמו שהוצע במטלה, ובנוסף, לולאות אשר מוודאות שאם התקבל אישור על חבילה אחרת, נחזור ונאזין עד שנקבל את האישור המתאים או שיגמר הזמן המוגדר (TIMEOUT) . אם אכן הגענו ל TIMEOUT נשלח את החבילה בשנית.

שליחה המידע במימוש שלנו היא אומנם לא הכי מהירה או מתוחכמת, אבל ברמת אמינות גבוה, הנשמרת גם במקרה של אובדן חבילות.

לצורך הסבר יותר מפורט של מימוש הפרוטוקול נצרף דיאגרמת מצבים של ה Receiver



פירוט API כפי שמופיע בקובץ RUDP_API.h.

הערה: מימוש הפונקציות נעזר בפונקציות עזר שאינן נגישות לשולח ולמקבל ולכן אינן מופיעות בקובץ זה.
הצהרות ומימוש אותן פונקציות עזר מופיעים יחד עם מימוש ה API בקובץ RUDP_API.c

המקבל והשולח משתמשים בפונקציות כקופצא שחורה, ומשאירים את בדיקת המהימנות מימוש עצמו.

```
/**
 * Creating a RUDP socket
 * @return the socket number if success, -1 if failed.
 */
int RUDP_socket();

/**
 * opening a connection between two peers.
 * (a connection is established with 2 way handshake - SYN, SYN-ACK)
 *
 * @param socket - the socket to connect to.
 * @return 1 is success, 0 if failed.
 */
int RUDP_connect(int socket, char *ip, int port);

/**
 * Listening for incoming connection requests.
 * (a connection is established with 2 way handshake - SYN, SYN-ACK)
 *
 * @param socket - the socket to listen to.
 * @return 1 is success, 0 if failed.
 */
int RUDP_get_connection(int socket, int port);

/**
 * Sending data to the peer.
 *  The function should wait for an acknowledgment packet, and if it
didn't
 * receive any, retransmits the data.
 *
 * @param socket - the socket to send from.
 * @param data - the data to send.
 * @param data_length - the length of the data.
 * @return 1 is success, 0 if failed.
 */
int RUDP_send(int socket, char *data, int data_length);

/**
 * Receiving data from the peer.
```

```
 * will put the received data in the data parameter, and the length of
the data
 * in the data_length parameter.
 *
 * @param socket - the socket to receive from.
 * @param data - the data to receive.
 * @param data_length - the length of the data.
 * @return 1 is data, 0 if nondata, 2 if last data packet,
 *        -2 if close connection, -1 is failed.
 */
int RUDP_receive(int socket, char **data, int *data_length);

/**
 * Closing the RUDP socket.
 */
int RUDP_close(int socket);
```

את המימוש ניתן לראות בקבצי הקוד המצורפים, ובהמשך קובץ זה הסברים על הביצורים של הפרוטוקול והניתוחים הסטטיסטים.

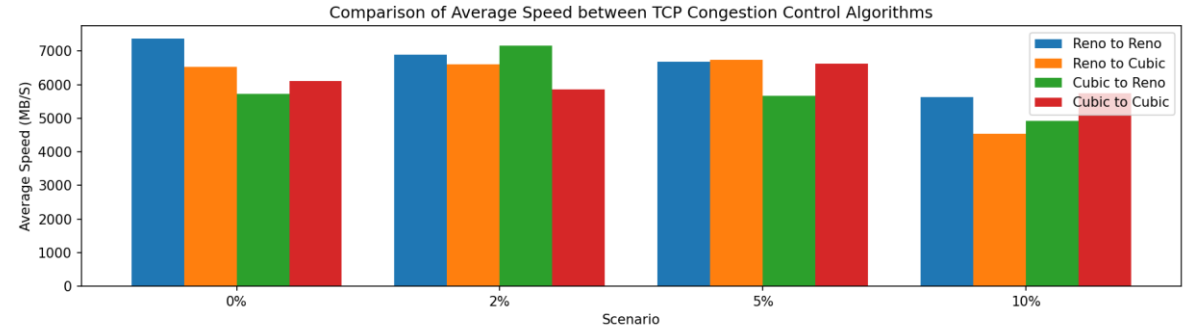
# Part C

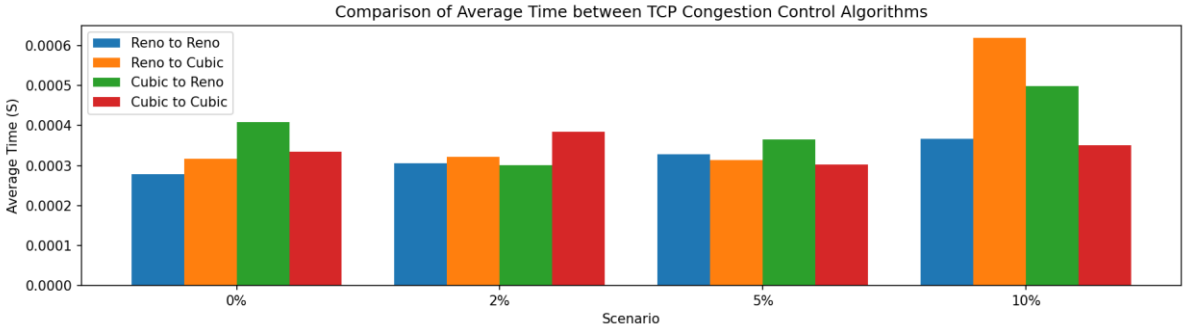
TCP reno & cubic

עשינו את החלק של הבונוס – הרצנו 4*4 פעמים, לכל ה4 אפשרויות את ה4 אפשרויות של packet loss, בכל הרצה שלחנו את הקובץ 5 פעמים.

בעזרת הפונקציה שסופקה בנספח C יצרנו קובץ גדול – 2MB בTCP_Sender ואותו שלחנו לTCP_Reciever .

לפני הכל יצרנו טבלאות של השוואה של המידע




בהתבוננות בטבלה ניתן לראות שכאשר איבוד המידע עולה תרחשים שבהם נעשה שימוש באלגוריתם reno הראו ירידה בביצועים, ומכך הגענו למסקנות הבאות:

cubic לcubic שמר על ביצועים יחסית יציבים, גם כשאיבוד המידע זינק. ללא איבוד מידע reno לreno מציג את הביצועים הגבוהים ביותר.

reno->reno

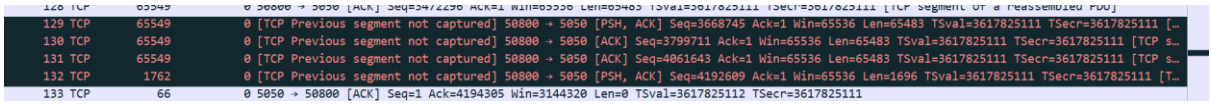
- 0% packet loss

אלו הנתונים שיצאו לנו מההרצה:

Run #1 Data: Time=0.000373 S ; Speed=5361.930295 MB/S
Run #2 Data: Time=0.000260 S ; Speed=7692.307692 MB/S
Run #3 Data: Time=0.000263 S ; Speed=7604.562738 MB/S
Run #4 Data: Time=0.000245 S ; Speed=8163.265306 MB/S
Run #5 Data: Time=0.000248 S ; Speed=8064.516129 MB/S
Average time: 0.000278 S
Average speed: 7377.316432 MB/S

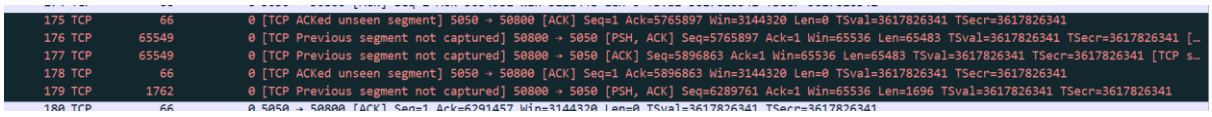
אפשר לשים לב למרות שהגדרנו 0% איבוד מידע, עדיין היה לנו קצת איבוד

למשל כאן הוא לא קיבל את הסגמנט הקודם.



| | | | |
|---|---|---|---|
| 129 | TCP | 65549 | 0 [TCP Previous segment not captured] 50800 → 5050 [PSH, ACK] Seq=3668745 Ack=1 Win=65536 Len=65483 TSval=3617825111 TSecr=3617825111 [… |
| 130 | TCP | 65549 | 0 [TCP Previous segment not captured] 50800 → 5050 [ACK] Seq=3799711 Ack=1 Win=65536 Len=65483 TSval=3617825111 TSecr=3617825111 [TCP s… |
| 131 | TCP | 65549 | 0 [TCP Previous segment not captured] 50800 → 5050 [ACK] Seq=4061643 Ack=1 Win=65536 Len=65483 TSval=3617825111 TSecr=3617825111 [TCP s… |
| 132 | TCP | 1762 | 0 [TCP Previous segment not captured] 50800 → 5050 [PSH, ACK] Seq=4192609 Ack=1 Win=65536 Len=1696 TSval=3617825111 TSecr=3617825111 [T… |
| 133 | TCP | 66 | 0 5050 → 50800 [ACK] Seq=1 Ack=4194305 Win=3144320 Len=0 TSval=3617825112 TSecr=3617825111 |

וכאן זה מציין שהWireshark לא קלט את הack של החבילה



| | | | |
|---|---|---|---|
| 175 | TCP | 66 | 0 [TCP ACKed unseen segment] 5050 → 50800 [ACK] Seq=1 Ack=5765897 Win=3144320 Len=0 TSval=3617826341 TSecr=3617826341 |
| 176 | TCP | 65549 | 0 [TCP Previous segment not captured] 50800 → 5050 [PSH, ACK] Seq=5765897 Ack=1 Win=65536 Len=65483 TSval=3617826341 TSecr=3617826341 [… |
| 177 | TCP | 65549 | 0 [TCP Previous segment not captured] 50800 → 5050 [ACK] Seq=5896863 Ack=1 Win=65536 Len=65483 TSval=3617826341 TSecr=3617826341 [TCP s… |
| 178 | TCP | 66 | 0 [TCP ACKed unseen segment] 5050 → 50800 [ACK] Seq=1 Ack=5896863 Win=3144320 Len=0 TSval=3617826341 TSecr=3617826341 |
| 179 | TCP | 1762 | 0 [TCP Previous segment not captured] 50800 → 5050 [PSH, ACK] Seq=6289761 Ack=1 Win=65536 Len=1696 TSval=3617826341 TSecr=3617826341 |

- 2% packet loss

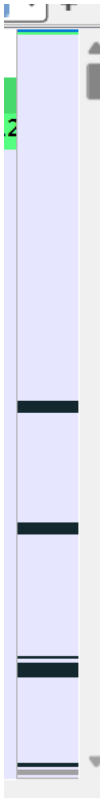
Run #1 Data: Time=0.000386 S ; Speed=5181.347150 MB/S
Run #2 Data: Time=0.000351 S ; Speed=5698.005698 MB/S
Run #3 Data: Time=0.000331 S ; Speed=6042.296073 MB/S
Run #4 Data: Time=0.000240 S ; Speed=8333.333333 MB/S
Run #5 Data: Time=0.000217 S ; Speed=9216.589862 MB/S
Average time: 0.000305 S
Average speed: 6894.314423 MB/S

אפשר לראות שלקח לנו קצת יותר זמן להעביר על קובץ, וגם הspeed הממוצע ירד.

אפשר לראות כאן שקיבלנו כאן ACK כפול, ושהוא פספס חלק מהACK. (היו עוד מקומות שבהן היו נפילות)

| | | | |
|---|---|---|---|
| 211 | TCP | 65549 | 0 56590 → 5050 [ACK] Seq=7797366 Ack=1 Win=65536 Len=65483 TSval=3618109591 TSecr=3618109591 |
| 212 | TCP | 65549 | 0 [TCP Previous segment not captured] 56590 → 5050 [ACK] Seq=7928532 Ack=1 Win=65536 Len=65483 TSval=3618109591 TSecr=3618109591 |
| 213 | TCP | 78 | 0 5050 → 56590 [ACK] Seq=1 Ack=7863049 Win=2978944 Len=0 TSval=3618109591 TSecr=3618109591 SLE=7928532 SRE=7994015 |
| 214 | TCP | 65549 | 0 [TCP Previous segment not captured] 56590 → 5050 [ACK] Seq=8059498 Ack=1 Win=65536 Len=65483 TSval=3618109591 TSecr=3618109591 |
| 215 | TCP | 78 | 0 [TCP Dup ACK 213#1] 5050 → 56590 [ACK] Seq=1 Ack=7863049 Win=2978944 Len=0 TSval=3618109591 TSecr=3618109591 SLE=7928532 SRE=8124981 |
| 216 | TCP | 65549 | 0 [TCP Previous segment not captured] 56590 → 5050 [ACK] Seq=8190464 Ack=1 Win=65536 Len=65483 TSval=3618109591 TSecr=3618109591 |
| 217 | TCP | 66 | 0 [TCP ACKed unseen segment] 5050 → 56590 [ACK] Seq=1 Ack=8386913 Win=2747776 Len=0 TSval=3618109591 TSecr=3618109591 |
| 218 | TCP | 1762 | 0 [TCP Previous segment not captured] 56590 → 5050 [PSH, ACK] Seq=8386913 Ack=1 Win=65536 Len=1696 TSval=3618109591 TSecr=3618109591 |
| 219 | TCP | 66 | 0 5050 → 56590 [ACK] Seq=1 Ack=8388609 Win=3144320 Len=0 TSval=3618109591 TSecr=3618109591 |

אלו הנפילות שהיו – כל שורה שחורה זה נפילה



- 5% packet loss

Run #1 Data: Time=0.000537 S ; Speed=3724.394786 MB/S
Run #2 Data: Time=0.000327 S ; Speed=6116.207951 MB/S
Run #3 Data: Time=0.000290 S ; Speed=6896.551724 MB/S
Run #4 Data: Time=0.000267 S ; Speed=7490.636704 MB/S
Run #5 Data: Time=0.000217 S ; Speed=9216.589862 MB/S
Average time: 0.000328 S
Average speed: 6688.876205 MB/S

גם כאן אפשר לראות ירידה בביצועים.

אפשר לראות גם שהיו הרבה יותר נפילות (כל שורה שחורה היא נפילה)

הפעם קיבלנו עוד נפילות מסוגים שונים

חוסר סדר בACK

| | | | | | |
|---|---|---|---|---|---|
| 20 | TCP | 32918 | 5050 | 32807 | 0 32918 → 5050 [ACK] Seq=327411 Ack=1 Win=65536 Len=32741 TSval=3618221449 TSecr=3618221449 |
| 21 | TCP | 32918 | 5050 | 32807 | 0 [TCP Out-Of-Order] 32918 → 5050 [PSH, ACK] Seq=163706 Ack=1 Win=65536 Len=32741 TSval=3618221449 TSecr=36182214 |
| 22 | TCP | 5050 | 32918 | 66 | 0 5050 → 32918 [ACK] Seq=1 Ack=360152 Win=982272 Len=0 TSval=3618221449 TSecr=3618221449 |

כאן גם קיבלנו ACK כפול, fast retransmission

| | | | | | | |
|---|---|---|---|---|---|---|
| 82 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=2293602 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618221450 |
| 83 | TCP | 5050 | 32918 | 66 | 0 | 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 |
| 84 | TCP | 32918 | 5050 | 65549 | 0 | [TCP Previous segment not captured] 32918 → 5050 [ACK] Seq=2490051 Ack=1 Win=65536 Len=65483 TSval=3618223037 T |
| 85 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=2555534 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618221450 |
| 86 | TCP | 5050 | 32918 | 78 | 0 | [TCP Dup ACK 83#1] 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 SLE |
| 87 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=2621017 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618221450 |
| 88 | TCP | 5050 | 32918 | 78 | 0 | [TCP Dup ACK 83#2] 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 SLE |
| 89 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=2686500 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618221450 |
| 90 | TCP | 5050 | 32918 | 78 | 0 | [TCP Dup ACK 83#3] 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 SLE |
| 91 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=2751983 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618223037 |
| 92 | TCP | 5050 | 32918 | 78 | 0 | [TCP Dup ACK 83#4] 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 SLE |
| 93 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=2817466 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618223037 [TCP segment of a re |
| 94 | TCP | 5050 | 32918 | 78 | 0 | [TCP Dup ACK 83#5] 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 SLE |
| 95 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=2882949 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618223037 [TCP segment of a re |
| 96 | TCP | 5050 | 32918 | 78 | 0 | [TCP Dup ACK 83#6] 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 SLE |
| 97 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=2948432 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618223037 [TCP segment of a re |
| 98 | TCP | 5050 | 32918 | 78 | 0 | [TCP Dup ACK 83#7] 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 SLE |
| 99 | TCP | 32918 | 5050 | 65549 | 0 | 32918 → 5050 [ACK] Seq=3013915 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618223037 [TCP segment of a re |
| 100 | TCP | 5050 | 32918 | 78 | 0 | [TCP Dup ACK 83#8] 5050 → 32918 [ACK] Seq=1 Ack=2359085 Win=3079040 Len=0 TSval=3618223037 TSecr=3618223037 SLE |
| 101 | TCP | 32918 | 5050 | 65549 | 0 | [TCP Fast Retransmission] 32918 → 5050 [ACK] Seq=2359085 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618 |
| 102 | TCP | 5050 | 32918 | 78 | 0 | 5050 → 32918 [ACK] Seq=1 Ack=2424568 Win=3013632 Len=0 TSval=3618223037 TSecr=3618223037 SLE=2490051 SRE=307939 |
| 103 | TCP | 32918 | 5050 | 65549 | 0 | [TCP Out-Of-Order] 32918 → 5050 [ACK] Seq=2424568 Ack=1 Win=65536 Len=65483 TSval=3618223037 TSecr=3618223037 |
| 104 | TCP | 5050 | 32918 | 66 | 0 | 5050 → 32918 [ACK] Seq=1 Ack=3079398 Win=2716928 Len=0 TSval=3618223037 TSecr=3618223037 |

- 10% packet loss

Run #1 Data: Time=0.000371 S ; Speed=5390.835580 MB/S
Run #2 Data: Time=0.000342 S ; Speed=5847.953216 MB/S
Run #3 Data: Time=0.000487 S ; Speed=4106.776181 MB/S
Run #4 Data: Time=0.000350 S ; Speed=5714.285714 MB/S
Run #5 Data: Time=0.000284 S ; Speed=7042.253521 MB/S
Average time: 0.000367 S
Average speed: 5620.420842 MB/S

גם כאן אפשר לראות ירידה בביצועים.

כאן רואים הכי הרבה נפילות (לא מפתיע)

הפעם קיבלנו שגיאה חדשה – TCP Retransmission שזה מציין שההודעה קיבלה timeout לקבלת ack אז היא שלחה את הסגמנט מחדש.

| | | | | | |
|---|---|---|---|---|---|
| 15 | TCP | 56020 | 5050 | 32807 | 0 [TCP Previous segment not captured] 56020 → 5050 [PSH, ACK] Seq=294670 Ack=1 Win=65536 Len=32741 TSval=36182976 |
| 16 | TCP | 5050 | 56020 | 78 | 0 [TCP Window Update] 5050 → 56020 [ACK] Seq=1 Ack=229188 Win=589440 Len=0 TSval=3618297093 TSecr=3618297093 SLE= |
| 17 | TCP | 56020 | 5050 | 32807 | 0 [TCP Previous segment not captured] 56020 → 5050 [PSH, ACK] Seq=360152 Ack=1 Win=65536 Len=32741 TSval=36182976 |
| 18 | TCP | 5050 | 56020 | 86 | 0 [TCP Window Update] 5050 → 56020 [ACK] Seq=1 Ack=229188 Win=720384 Len=0 TSval=3618297093 TSecr=3618297093 SLE= |
| 19 | TCP | 56020 | 5050 | 32807 | 0 56020 → 5050 [ACK] Seq=392893 Ack=1 Win=65536 Len=32741 TSval=3618297093 TSecr=3618297093 |
| 20 | TCP | 5050 | 56020 | 86 | 0 [TCP Window Update] 5050 → 56020 [ACK] Seq=1 Ack=229188 Win=851328 Len=0 TSval=3618297093 TSecr=3618297093 SLE= |
| 21 | TCP | 56020 | 5050 | 32807 | 0 56020 → 5050 [PSH, ACK] Seq=425634 Ack=1 Win=65536 Len=32741 TSval=3618297093 TSecr=3618297093 |
| 22 | TCP | 5050 | 56020 | 86 | 0 [TCP Window Update] 5050 → 56020 [ACK] Seq=1 Ack=229188 Win=982272 Len=0 TSval=3618297093 TSecr=3618297093 SLE= |
| 23 | TCP | 56020 | 5050 | 65548 | 0 [TCP Retransmission] 56020 → 5050 [PSH, ACK] Seq=229188 Ack=1 Win=65536 Len=65482 TSval=3618297093 TSecr=361829 |
| 24 | TCP | 5050 | 56020 | 78 | 0 5050 → 56020 [ACK] Seq=1 Ack=327411 Win=1113216 Len=0 TSval=3618297093 TSecr=3618297093 SLE=360152 SRE=458375 |
| 25 | TCP | 56020 | 5050 | 32807 | 0 [TCP Retransmission] 56020 → 5050 [ACK] Seq=327411 Ack=1 Win=65536 Len=32741 TSval=3618297093 TSecr=3618297093 |
| 26 | TCP | 56020 | 5050 | 32807 | 0 56020 → 5050 [ACK] Seq=458375 Ack=1 Win=65536 Len=32741 TSval=3618297093 TSecr=3618297093 |
| 27 | TCP | 5050 | 56020 | 66 | 0 5050 → 56020 [ACK] Seq=1 Ack=491116 Win=1375232 Len=0 TSval=3618297093 TSecr=3618297093 |

וכאן אפשר לראות עוד שגיאות שקרו:

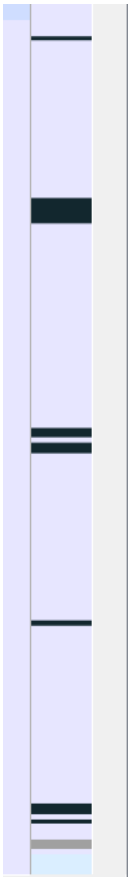
| | | | | | |
|---|---|---|---|---|---|
| 222 | TCP | 56020 | 5050 | 65549 | 0 [TCP Previous segment not captured] 56020 → 5050 [ACK] Seq=7666600 Ack=1 Win=65536 Len=65483 TSval=3618300021 T |
| 223 | TCP | 5050 | 56020 | 66 | 0 5050 → 56020 [ACK] Seq=1 Ack=7601117 Win=3079040 Len=0 TSval=3618300021 TSecr=3618300021 |
| 224 | TCP | 5050 | 56020 | 78 | 0 [TCP Window Update] 5050 → 56020 [ACK] Seq=1 Ack=7601117 Win=3112448 Len=0 TSval=3618300021 TSecr=3618300021 SL |
| 225 | TCP | 56020 | 5050 | 65549 | 0 56020 → 5050 [ACK] Seq=7732083 Ack=1 Win=65536 Len=65483 TSval=3618300021 TSecr=3618300021 |
| 226 | TCP | 5050 | 56020 | 78 | 0 [TCP Dup ACK 223#1] 5050 → 56020 [ACK] Seq=1 Ack=7601117 Win=3112448 Len=0 TSval=3618300021 TSecr=3618300021 SL |
| 227 | TCP | 5050 | 56020 | 86 | 0 [TCP Dup ACK 223#2] 5050 → 56020 [ACK] Seq=1 Ack=7601117 Win=3112448 Len=0 TSval=3618300021 TSecr=3618300021 SL |
| 228 | TCP | 56020 | 5050 | 65549 | 0 [TCP Fast Retransmission] 56020 → 5050 [ACK] Seq=7601117 Ack=1 Win=65536 Len=65483 TSval=3618300021 TSecr=3618 |
| 229 | TCP | 56020 | 5050 | 65549 | 0 [TCP Previous segment not captured] 56020 → 5050 [ACK] Seq=7928532 Ack=1 Win=65536 Len=65483 TSval=3618300021 T |
| 230 | TCP | 5050 | 56020 | 78 | 0 [TCP ACKed unseen segment] 5050 → 56020 [ACK] Seq=1 Ack=7928532 Win=2947456 Len=0 TSval=3618300021 TSecr=361836 |
| 231 | TCP | 5050 | 56020 | 66 | 0 [TCP ACKed unseen segment] 5050 → 56020 [ACK] Seq=1 Ack=8059498 Win=2914688 Len=0 TSval=3618300021 TSecr=361836 |
| 232 | TCP | 56020 | 5050 | 65549 | 0 [TCP Previous segment not captured] 56020 → 5050 [ACK] Seq=8059498 Ack=1 Win=65536 Len=65483 TSval=3618300021 T |
| 233 | TCP | 56020 | 5050 | 65549 | 0 56020 → 5050 [ACK] Seq=8124981 Ack=1 Win=65536 Len=65483 TSval=3618300021 TSecr=3618300021 |
| 234 | TCP | 5050 | 56020 | 66 | 0 5050 → 56020 [ACK] Seq=1 Ack=8190464 Win=3145728 Len=0 TSval=3618300021 TSecr=3618300021 |
| 235 | TCP | 56020 | 5050 | 65549 | 0 56020 → 5050 [ACK] Seq=8190464 Ack=1 Win=65536 Len=65483 TSval=3618300021 TSecr=3618300021 |
| 236 | TCP | 56020 | 5050 | 65549 | 0 56020 → 5050 [ACK] Seq=8255947 Ack=1 Win=65536 Len=65483 TSval=3618300021 TSecr=3618300021 |
| 237 | TCP | 5050 | 56020 | 66 | 0 5050 → 56020 [ACK] Seq=1 Ack=8321430 Win=3145728 Len=0 TSval=3618300021 TSecr=3618300021 |
| 238 | TCP | 56020 | 5050 | 1762 | 0 [TCP Previous segment not captured] 56020 → 5050 [PSH, ACK] Seq=8386913 Ack=1 Win=65536 Len=1696 TSval=36183006 |
| 239 | TCP | 5050 | 56020 | 78 | 0 [TCP Dup ACK 237#1] 5050 → 56020 [ACK] Seq=1 Ack=8321430 Win=3145728 Len=0 TSval=3618300021 TSecr=3618300021 SL |
| 240 | TCP | 56020 | 5050 | 65549 | 0 [TCP Out-Of-Order] 56020 → 5050 [ACK] Seq=8321430 Ack=1 Win=65536 Len=65483 TSval=3618300021 TSecr=3618300021 |
| 241 | TCP | 5050 | 56020 | 66 | 0 5050 → 56020 [ACK] Seq=1 Ack=8388609 Win=3078656 Len=0 TSval=3618300021 TSecr=3618300021 |

cubic ->reno

- 0%packet loss

Run #1 Data: Time=0.000757 S ; Speed=2642.007926 MB/S
Run #2 Data: Time=0.000305 S ; Speed=6557.377049 MB/S
Run #3 Data: Time=0.000238 S ; Speed=8403.361345 MB/S
Run #4 Data: Time=0.000307 S ; Speed=6514.657980 MB/S
Run #5 Data: Time=0.000439 S ; Speed=4555.808656 MB/S
Average time: 0.000409 S
Average speed: 5734.642591 MB/S

גם כאן למרות שהגדרנו 0% עדיין היו טיפה איבודים



- 2% packet loss

Run #1 Data: Time=0.000359 S ; Speed=5571.030641 MB/S
Run #2 Data: Time=0.000271 S ; Speed=7380.073801 MB/S
Run #3 Data: Time=0.000437 S ; Speed=4576.659039 MB/S
Run #4 Data: Time=0.000225 S ; Speed=8888.888889 MB/S
Run #5 Data: Time=0.000212 S ; Speed=9433.962264 MB/S
Average time: 0.000301 S
Average speed: 7170.122927 MB/S

כאן אפשר לראות שגיאה חדשה שלא הייתה קודם

נראה שאחרי שהשרת סגר את הקשר, הלקוח עדיין מנסה לשלוח מידע, ואז גם מנסה ליצור קשר מחדש ולא מצליח.

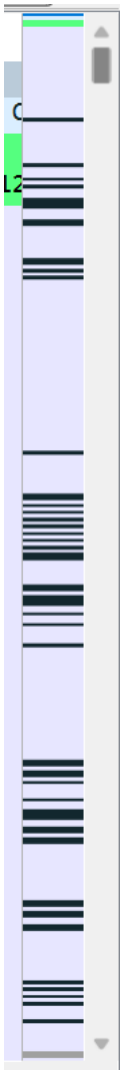
| | | | | | | |
|---|---|---|---|---|---|---|
| 256 | TCP | 5050 | 42386 | 66 | 0 | 5050 → 42386 [ACK] Seq=1 Ack=10485767 Win=3145728 Len=0 TSval=3618677503 TSecr=3618677503 |
| 257 | TCP | 5050 | 42386 | 66 | 0 | 5050 → 42386 [FIN, ACK] Seq=1 Ack=10485767 Win=3145728 Len=0 TSval=3618677504 TSecr=3618677503 |
| 258 | TCP | 42386 | 5050 | 66 | 0 | 42386 → 5050 [ACK] Seq=10485767 Ack=2 Win=65536 Len=0 TSval=3618677504 TSecr=3618677504 |
| 259 | TCP | 35250 | 5037 | 74 | 1 | 35250 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618682502 TSecr=0 WS=128 |
| 260 | TCP | 35250 | 5037 | 74 | 1 | [TCP Retransmission] 35250 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683528 TSecr=0 WS=1 |
| 261 | TCP | 5037 | 35250 | 54 | 1 | 5037 → 35250 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 262 | TCP | 35256 | 5037 | 74 | 2 | 35256 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683533 TSecr=0 WS=128 |
| 263 | TCP | 5037 | 35256 | 54 | 2 | 5037 → 35256 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 264 | TCP | 35260 | 5037 | 74 | 3 | 35260 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683536 TSecr=0 WS=128 |
| 265 | TCP | 5037 | 35260 | 54 | 3 | 5037 → 35260 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 266 | TCP | 35276 | 5037 | 74 | 4 | 35276 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683537 TSecr=0 WS=128 |
| 267 | TCP | 5037 | 35276 | 54 | 4 | 5037 → 35276 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 268 | TCP | 35284 | 5037 | 74 | 5 | 35284 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683538 TSecr=0 WS=128 |
| 269 | TCP | 5037 | 35284 | 54 | 5 | 5037 → 35284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 270 | TCP | 35300 | 5037 | 74 | 6 | 35300 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683541 TSecr=0 WS=128 |
| 271 | TCP | 5037 | 35300 | 54 | 6 | 5037 → 35300 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 272 | TCP | 35308 | 5037 | 74 | 7 | 35308 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683542 TSecr=0 WS=128 |
| 273 | TCP | 5037 | 35308 | 54 | 7 | 5037 → 35308 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 274 | TCP | 35320 | 5037 | 74 | 8 | 35320 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683543 TSecr=0 WS=128 |
| 275 | TCP | 5037 | 35320 | 54 | 8 | 5037 → 35320 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 276 | TCP | 35334 | 5037 | 74 | 9 | 35334 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683545 TSecr=0 WS=128 |
| 277 | TCP | 5037 | 35334 | 54 | 9 | 5037 → 35334 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 278 | TCP | 35340 | 5037 | 74 | 10 | 35340 → 5037 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3618683546 TSecr=0 WS=128 |
| 279 | TCP | 5037 | 35340 | 54 | 10 | 5037 → 35340 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |

- 5% packet loss

Run #1 Data: Time=0.000429 S ; Speed=4662.004662 MB/S
Run #2 Data: Time=0.000304 S ; Speed=6578.947368 MB/S
Run #3 Data: Time=0.000279 S ; Speed=7168.458781 MB/S
Run #4 Data: Time=0.000445 S ; Speed=4494.382022 MB/S
Run #5 Data: Time=0.000370 S ; Speed=5405.405405 MB/S
Average time: 0.000365 S
Average speed: 5661.839648 MB/S

גם כאן אפשר לראות ירידה בביצועים

והרבה יותר נפילות של המידע.



- 10% packet loss

Run #1 Data: Time=0.000772 S ; Speed=2590.673575 MB/S
Run #2 Data: Time=0.000256 S ; Speed=7812.500000 MB/S
Run #3 Data: Time=0.000575 S ; Speed=3478.260870 MB/S
Run #4 Data: Time=0.000623 S ; Speed=3210.272873 MB/S
Run #5 Data: Time=0.000265 S ; Speed=7547.169811 MB/S
Average time: 0.000498 S
Average speed: 4927.775426 MB/S

גם כאן אפשר לראות ירידה בביצועים ויותר איבוד מידע

reno->cubic

הפעם הsender יעבוד עם reno וה receiver יעבוד cubic.

- 0% packet loss

Run #1 Data: Time=0.000368 S ; Speed=5434.782609 MB/S
Run #2 Data: Time=0.000408 S ; Speed=4901.960784 MB/S
Run #3 Data: Time=0.000269 S ; Speed=7434.944238 MB/S
Run #4 Data: Time=0.000265 S ; Speed=7547.169811 MB/S
Run #5 Data: Time=0.000274 S ; Speed=7299.270073 MB/S
Average time: 0.000317 S
Average speed: 6523.625503 MB/S

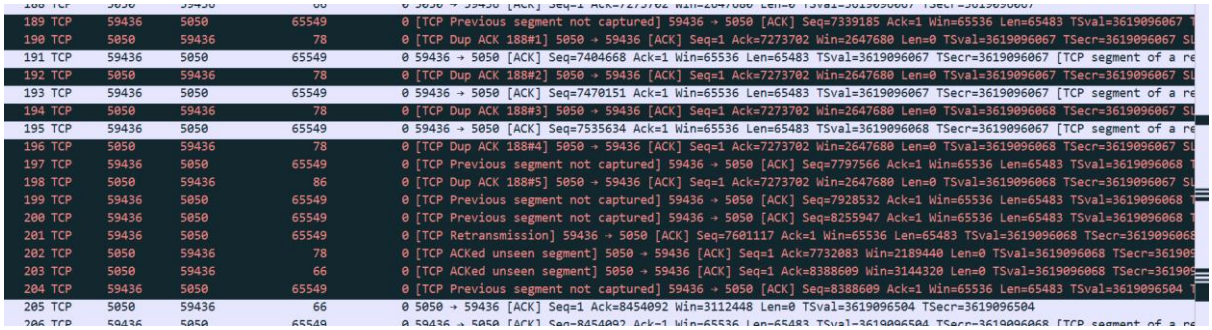
גם כאן היה כמות קטנה של איבוד מידע.

- 2% packet loss

Run #1 Data: Time=0.000330 S ; Speed=6060.606061 MB/S
Run #2 Data: Time=0.000229 S ; Speed=8733.624454 MB/S
Run #3 Data: Time=0.000434 S ; Speed=4608.294931 MB/S
Run #4 Data: Time=0.000241 S ; Speed=8298.755187 MB/S
Run #5 Data: Time=0.000378 S ; Speed=5291.005291 MB/S
Average time: 0.000322 S
Average speed: 6598.457185 MB/S

אפשר לראות ירידה בביצועים.

וגם כאן כל מיני שגיאות.



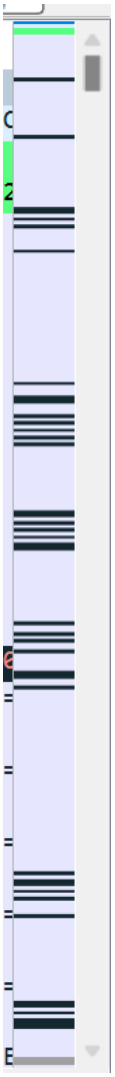
| | | | | | |
|---|---|---|---|---|---|
| 189 | TCP | 59436 | 5050 | 65549 | 0 [TCP Previous segment not captured] 59436 → 5050 [ACK] Seq=7339185 Ack=1 Win=65536 Len=65483 TSval=3619096067 T |
| 190 | TCP | 5050 | 59436 | 78 | 0 [TCP Dup ACK 188#1] 5050 → 59436 [ACK] Seq=1 Ack=7273702 Win=2647680 Len=0 TSval=3619096067 TSecr=3619096067 SL |
| 191 | TCP | 59436 | 5050 | 65549 | 0 59436 → 5050 [ACK] Seq=7404668 Ack=1 Win=65536 Len=65483 TSval=3619096067 TSecr=3619096067 [TCP segment of a re |
| 192 | TCP | 5050 | 59436 | 78 | 0 [TCP Dup ACK 188#2] 5050 → 59436 [ACK] Seq=1 Ack=7273702 Win=2647680 Len=0 TSval=3619096067 TSecr=3619096067 SL |
| 193 | TCP | 59436 | 5050 | 65549 | 0 59436 → 5050 [ACK] Seq=7470151 Ack=1 Win=65536 Len=65483 TSval=3619096067 TSecr=3619096067 [TCP segment of a re |
| 194 | TCP | 5050 | 59436 | 78 | 0 [TCP Dup ACK 188#3] 5050 → 59436 [ACK] Seq=1 Ack=7273702 Win=2647680 Len=0 TSval=3619096068 TSecr=3619096067 SL |
| 195 | TCP | 59436 | 5050 | 65549 | 0 59436 → 5050 [ACK] Seq=7535634 Ack=1 Win=65536 Len=65483 TSval=3619096068 TSecr=3619096067 [TCP segment of a re |
| 196 | TCP | 5050 | 59436 | 78 | 0 [TCP Dup ACK 188#4] 5050 → 59436 [ACK] Seq=1 Ack=7273702 Win=2647680 Len=0 TSval=3619096068 TSecr=3619096067 SL |
| 197 | TCP | 59436 | 5050 | 65549 | 0 [TCP Previous segment not captured] 59436 → 5050 [ACK] Seq=7797566 Ack=1 Win=65536 Len=65483 TSval=3619096068 T |
| 198 | TCP | 5050 | 59436 | 86 | 0 [TCP Dup ACK 188#5] 5050 → 59436 [ACK] Seq=1 Ack=7273702 Win=2647680 Len=0 TSval=3619096068 TSecr=3619096067 SL |
| 199 | TCP | 59436 | 5050 | 65549 | 0 [TCP Previous segment not captured] 59436 → 5050 [ACK] Seq=7928532 Ack=1 Win=65536 Len=65483 TSval=3619096068 T |
| 200 | TCP | 59436 | 5050 | 65549 | 0 [TCP Previous segment not captured] 59436 → 5050 [ACK] Seq=8255947 Ack=1 Win=65536 Len=65483 TSval=3619096068 T |
| 201 | TCP | 59436 | 5050 | 65549 | 0 [TCP Retransmission] 59436 → 5050 [ACK] Seq=7601117 Ack=1 Win=65536 Len=65483 TSval=3619096068 TSecr=3619096068 |
| 202 | TCP | 5050 | 59436 | 78 | 0 [TCP ACKed unseen segment] 5050 → 59436 [ACK] Seq=1 Ack=7732083 Win=2189440 Len=0 TSval=3619096068 TSecr=361909 |
| 203 | TCP | 5050 | 59436 | 66 | 0 [TCP ACKed unseen segment] 5050 → 59436 [ACK] Seq=1 Ack=8388609 Win=3144320 Len=0 TSval=3619096068 TSecr=361909 |
| 204 | TCP | 59436 | 5050 | 65549 | 0 [TCP Previous segment not captured] 59436 → 5050 [ACK] Seq=8388609 Ack=1 Win=65536 Len=65483 TSval=3619096504 T |
| 205 | TCP | 5050 | 59436 | 66 | 0 5050 → 59436 [ACK] Seq=1 Ack=8454092 Win=3112448 Len=0 TSval=3619096504 TSecr=3619096504 |

- 5% packet loss

Run #1 Data: Time=0.000251 S ; Speed=7968.127490 MB/S
Run #2 Data: Time=0.000470 S ; Speed=4255.319149 MB/S
Run #3 Data: Time=0.000238 S ; Speed=8403.361345 MB/S
Run #4 Data: Time=0.000300 S ; Speed=6666.666667 MB/S
Run #5 Data: Time=0.000310 S ; Speed=6451.612903 MB/S
Average time: 0.000314 S
Average speed: 6749.017511 MB/S

אפשר לראות ירידה קטנה בביצועים.

וגם הרבה נפילות מידע



- 10% packet loss

Run #1 Data: Time=0.001494 S ; Speed=1338.688086 MB/S
Run #2 Data: Time=0.000327 S ; Speed=6116.207951 MB/S
Run #3 Data: Time=0.000322 S ; Speed=6211.180124 MB/S
Run #4 Data: Time=0.000603 S ; Speed=3316.749585 MB/S
Run #5 Data: Time=0.000350 S ; Speed=5714.285714 MB/S
Average time: 0.000619 S
Average speed: 4539.422292 MB/S

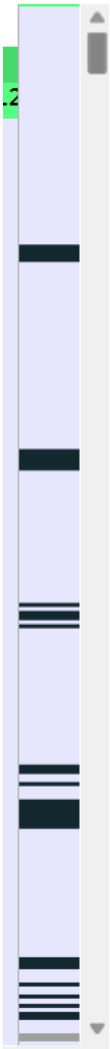
אפשר לראות שיש כאן הרבה נפילות מידע

cubic->cubic

הפעם שני הצדדים עובדים עם cubic.

- 0% packet loss

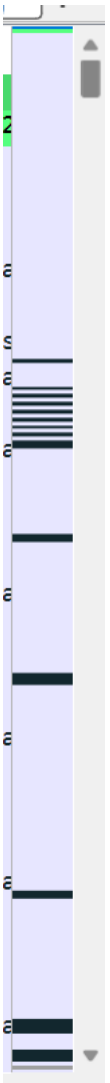
Run #1 Data: Time=0.000435 S ; Speed=4597.701149 MB/S
Run #2 Data: Time=0.000318 S ; Speed=6289.308176 MB/S
Run #3 Data: Time=0.000333 S ; Speed=6006.006006 MB/S
Run #4 Data: Time=0.000281 S ; Speed=7117.437722 MB/S
Run #5 Data: Time=0.000308 S ; Speed=6493.506494 MB/S
Average time: 0.000335 S
Average speed: 6100.791909 MB/S



- 2% packet loss

Run #1 Data: Time=0.000640 S ; Speed=3125.000000 MB/S
Run #2 Data: Time=0.000222 S ; Speed=9009.009009 MB/S
Run #3 Data: Time=0.000322 S ; Speed=6211.180124 MB/S
Run #4 Data: Time=0.000416 S ; Speed=4807.692308 MB/S
Run #5 Data: Time=0.000326 S ; Speed=6134.969325 MB/S
Average time: 0.000385 S
Average speed: 5857.570153 MB/S

גם כאן אפשר לראות ירידה בביצועים.



- 5% packet loss

Run #1 Data: Time=0.000251 S ; Speed=7968.127490 MB/S
Run #2 Data: Time=0.000470 S ; Speed=4255.319149 MB/S
Run #3 Data: Time=0.000238 S ; Speed=8403.361345 MB/S
Run #4 Data: Time=0.000300 S ; Speed=6666.666667 MB/S
Run #5 Data: Time=0.000310 S ; Speed=6451.612903 MB/S
Average time: 0.000314 S
Average speed: 6749.017511 MB/S

אפשר לראות ירידת בביצועים לעומת ה0% אבל עליה מה2% (אנחנו חושבים שזאת סטיית תקן לא מייצגת)

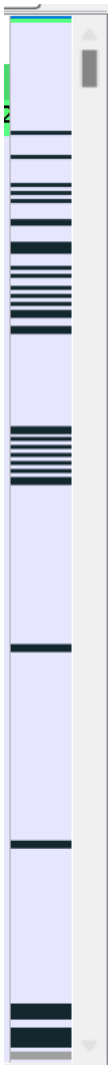
אלו הנפילות שהיו:

- 10% packet loss

Run #1 Data: Time=0.000388 S ; Speed=5154.639175 MB/S
Run #2 Data: Time=0.000336 S ; Speed=5952.380952 MB/S
Run #3 Data: Time=0.000388 S ; Speed=5154.639175 MB/S
Run #4 Data: Time=0.000323 S ; Speed=6191.950464 MB/S
Run #5 Data: Time=0.000319 S ; Speed=6269.592476 MB/S
Average time: 0.000351 S
Average speed: 5744.640449 MB/S

אפשר לראות ירידה בביצועים כאן.

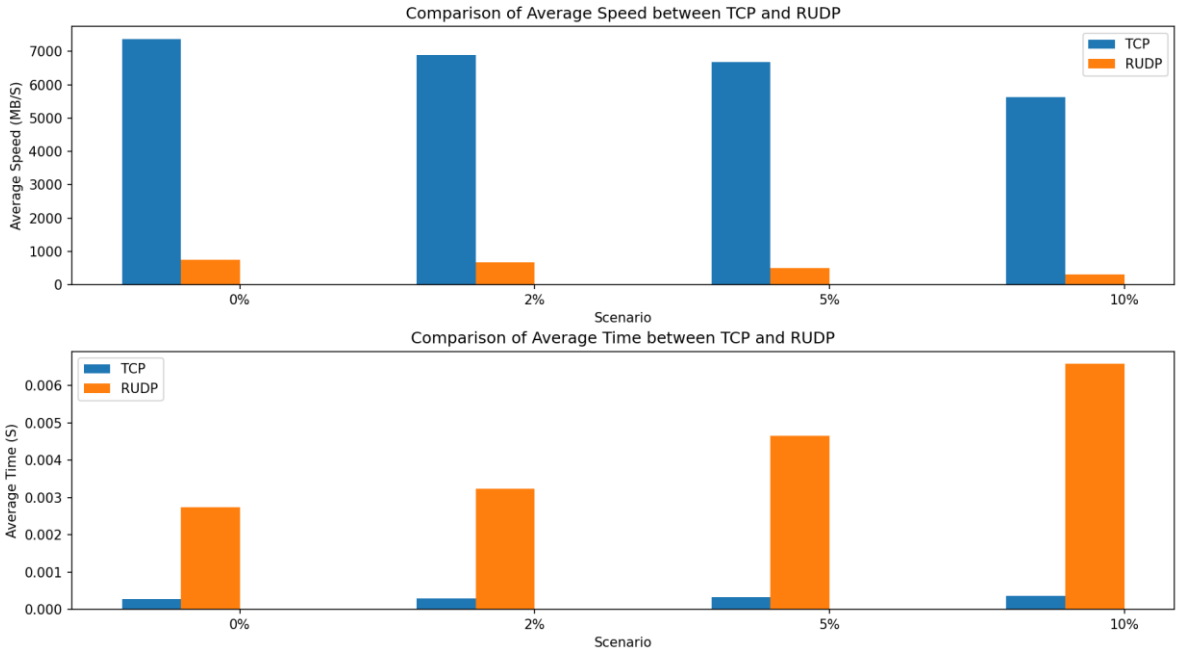
אלו השגיאות שקרו:



## שאלות של part C

1. כפי שצוין בניתוח הנתונים כשאין איבוד מידע reno ל reno נתן את הביצועים הטובים ביותר, אבל ככל שאיבוד המידע עלה, cubic נתן ביצועים יציבים יחסית לעומת reno שנתן ירידה חדה יותר.

.2




כמו בחלק א' השוונו את הביצועים של שני הפרוטוקולים עבור TCP בחרנו ב reno לreno.

כפי שניתן לראות TCP הרבה יותר יעיל מהRUDP שלנו, וTCP רגיל נתן ביצועים יותר טובים עבור איבוד מידע גבוה. (מהירות העברת המידע הממוצעת הייתה גבוהה יותר, והזמן הממוצע שלקח לשלוח את הקובץ הוא קטן יותר).

צפינו תוצאה זאת וכמו שציינו בpart B פרוטוקול הRUDP שלנו מממש flow control של stop and wait, שהוא פחות מהיר מהאחרים. בנוסף התמקדנו בהעברת מידע אמין מאשר העברה מהירה של המידע.

אלו ההדפסות שיצאנו לנו כאשר הרצנו את התוכנית:

1. 0% packet loss
Run #1 Data: Time=0.003152 S ; Speed=634.517766 MB/S
Run #2 Data: Time=0.002654 S ; Speed=753.579503 MB/S
Run #3 Data: Time=0.002192 S ; Speed=912.408759 MB/S
Run #4 Data: Time=0.003064 S ; Speed=652.741514 MB/S
Run #5 Data: Time=0.002600 S ; Speed=769.230769 MB/S

Average time: 0.002732 S
Average speed: 744.495662 MB/S

2. 2% packet loss
Run #1 Data: Time=0.002045 S ; Speed=977.995110 MB/S

Run #2 Data: Time=0.003164 S ; Speed=632.111252 MB/S
Run #3 Data: Time=0.003292 S ; Speed=607.533414 MB/S
Run #4 Data: Time=0.002854 S ; Speed=700.770848 MB/S
Run #5 Data: Time=0.004846 S ; Speed=412.711515 MB/S

Average time: 0.003240 S
Average speed: 666.224428 MB/S

3. 5% packet loss
Run #1 Data: Time=0.003035 S ; Speed=658.978583 MB/S
Run #2 Data: Time=0.005523 S ; Speed=362.122035 MB/S
Run #3 Data: Time=0.002590 S ; Speed=772.200772 MB/S
Run #4 Data: Time=0.007000 S ; Speed=285.714286 MB/S
Run #5 Data: Time=0.005102 S ; Speed=392.003136 MB/S

Average time: 0.004650 S
Average speed: 494.203762 MB/S

4. 10% packet loss
Run #1 Data: Time=0.004933 S ; Speed=405.432800 MB/S
Run #2 Data: Time=0.008443 S ; Speed=236.882625 MB/S
Run #3 Data: Time=0.005964 S ; Speed=335.345406 MB/S
Run #4 Data: Time=0.006439 S ; Speed=310.607237 MB/S
Run #5 Data: Time=0.007120 S ; Speed=280.898876 MB/S

Average time: 0.006580 S
Average speed: 313.833389 MB/S

3. אנחנו יודעים שקיים מימושים יעילים יותר ומהירים לRUDP, אך בהיתחשב במימוש שלנו, ובהתאם לניתוח התוצאות שהצגנו, תמיד נבחר להשתמש בTCP.

חלק D – שאלות פתוחות:

שאלה 1:

המקרה שבו שינוי הSSThreshold ישפיע בו במידה המירבית הוא מקרה 5 – "בקשר ארוך על גבי רשת אמינה עם RTT קטן.".

בעצם השינוי יגרום לזה שנשלח חבילות יותר גדולות יותר מהר – אם הקשר לא אמין או ש הRTT גדול, אז זה רק ייגרע כי אז timeout יגיע מהר יותר, ונרד בחזרה ל1 מהר יותר.

אבל אם הקשר ארוך, נוכל להגיע לשליחת חבילות צורה המירבית מהר יותר, כי הקשר אמין והRTT קטן אז נמצא את אותו threshold מהר.

שאלה 2:

אנחנו מתחילים את השליחה בגודל שליחה של 1, ואנחנו מתחילים תהליך של slow start, כלומר נכפיל את הגודל פי 2 בכל שליחה, בגלל שאין איבוד מידע אז הוא אף פעם לא יפיל את הגודל שליחה, ונמשיך לשלוח כל פעם פי שתיים מהפעם הקודמת. לכן כמות החבילות שנשלח היא $\log_2 s$, לכל חבילה ייקח RTT, לכן סך הזמן הוא בערך $logS \cdot RTT$ .

נחשב את כמות הסגמנטים שנשלחו -

$$\sum_{i=0}^{\log_2 S} 2^i = 2S - 1$$

וכל סגמנט הוא בגודל MSS

בסה"כ נקבל שהתשובה הראשונה היא הנכונה, כלומר התפוקה של הקשר תהיה בערך $2S \frac{MSS}{logS \cdot RTT}$.

שאלה 3:

*סיפרת הביקורת המינימלית היא* $X = 2$.

לפי הנתונים:

קצב תקשורת – 8Gbps

גודל כל חבילה – 2KBps

קצב התפשטות - $2 \cdot 10^8 m/sec$

מרחק בין התחנות – 1KM

חישובים:

את זמן ההשהיה נחשב $\frac{\text{מרחק}}{\text{קצב התפשטות}}$ כלומר $\frac{1000}{2 \cdot 10^8} = 0.0005$ שניות

זמן העברת חבילה אחת יהיה $\frac{\text{גודל החבילה}}{\text{קצב התקשורת}}$ כלומר $\frac{2 \cdot 10^3 \cdot 8}{8 \cdot 10^9} = 0.000002$

את הRTT נחשב ע"י זמן העברת חבילה אחת + 2 * זמן השהיה

לכן הRTT יהיה $2 \cdot 0.0005 + 0.000002 = 0.001002$ שניות.

אין איבוד מידע אז כל המידע הגיע ואין שידורים חוזרים, אז אנחנו יכולים להגדיל כמה שיותר את גודל החלון

$$window\ size = \frac{\text{קצב תקשורת} \cdot RTT}{\text{גודל חבילה}} = \frac{8 \cdot 10^9 \cdot 0.001002}{2 \cdot 8 \cdot 10^3} = 501$$