

שאלה 1 (בקרת גישה לערוצ וWIFI)

נתונה רשת אלחוטית ובה מספר גדול מאוד (אינסופי) של תחנות המשדרות לתחנת בסיס. תחנת הבסיס מודינה להודעות על גבי K תדרים שונים. הודעות שונות המשדרות באותו התדר עם חפיפה בזמן יתנגשו ולא יגיעו תקינות ליעד. עם זאת, הודעות המשדרות במקביל בתדרים שונים לא יפריעו זו לזו.

תחנות משדרות הודעות שזמן השידור שלתן הוא T_k שניות. הודעות אלו (חדשות וישנות) מופיעות במערכת לפי פילוג פואסוני עם קצב g הודעות בשניה.

- לאורך כל השאלה נתון כי זמני ההתפשטות בין התחנות השונות ובין התחנות לתחנת הבסיס זניחים.
- התחנות ותחנת הבסיס עושים שימוש ב프וטוקול Pure ALOHA כפי שנלמד בקורס.

בסעיפים הבאים, נתון כי כאשר מופיעה הודעה בתחנה כלשהי, היא מגירה באופן אחד אחד מכמה התדרים (באופן בלתי תלוי בין הודעות ובין תחנות), ומשדרת עלייה את ההודעה.

א. מהי התعبורה ברשת ביחסות של הודעות לשניה?

כפי שלמדנו התعبורה ברשת היא מدد בגודל של הודעות לשניה ומחושב לפי:

$$S = g \cdot P_{\text{succ}}$$

ישנם K תדרים שונים, ומהנתן שקצב ההודעות אחיד לכל תדר, קצב ההודעות המשדרות לכל תדר הוא $\frac{g}{K}$ הודעות בשניה. מכיוון שכל תדר מתנהג באופן אחיד, נוכל להתייחס לכל תדר כאיל מערכות נפרדת, ולבסוף נכפיל ב K כדי לקבל את התعبורה הכללית.

מנתנו שהודעות מופיעות במערכת לפי פילוג פואסוני עם קצב g הודעות בשניה | תחנות משדרות הודעות שזמן השידור שלתן הוא T_k שניות, וכך שמדובר ב프וטוקול Pure ALOHA, ההסתברות לכך ששידור בודד יצילח היא:

$$P_{\text{succ}} = e^{-2 \cdot kT \cdot \frac{g}{k}} = e^{-2 \cdot g \cdot T}$$

והتعبורה לשידור בודד היא:

$$S = \frac{g}{k} \cdot e^{-2gT}$$

נכפיל ב K ונקבל:

$$S = g \cdot e^{-2gT}$$

ב. מהי נצילות המערכת? ניתן (אך לא חובה) לבטא את התשובה כתלות בתעבורת, גם אם לא פתרתם את סעיף א'.

כפי שלמדנו נצילות המערכת הוא ממדים הנמדד ע"י נרמול בקיובל הקו:

$$n = \frac{S}{C}$$

הקיובל C מחושב כ-

$$\frac{k}{kT} = \frac{1}{T}$$

מה שمبטא את היחס בין מספר התדרים לבין הזמן לשידור הודעה אחת.

בסייף הקודם קיילנו שקצב התעבורה הכלול הוא:

$$S = g \cdot e^{-2gT}$$

נzieb בנוסחה ונקבל:

$$n = \frac{g \cdot e^{-2gT}}{\frac{1}{T}} = T \cdot g \cdot e^{-2gT} = T \cdot S$$

בסעיפים הבאים, נתן כי בזכות שדרוג טכנולוגי (הדומה ל-MA CSMA כפי שנלמד בקורס), כל תחנה פועלת באופן הבא:

כאשר מופיעה הודעה לנסיון שידור, התחנה בודקת את תפוזת התדרים השונים. ניתן להניח כי משך הבדיקה זניחה. אם קיימים תדרים פנוים, התחנה תגריל באופן אחד מהם לשדר בו את ההודעה. אחרת, נסיון השידור נכשל ולהודעה יוגרל זמן אקראי בעtid להגעה לנסיון שידור נוסף (ניתן להניח כי תהליך ההגעה של הודעות ישנות וחדשנות לנסיון שידור יותר פואסוני עם קצב g).

ג. מהי ההסתברות להצלחת נסיון שידור הודעה?

בסייף זה בלבד ניתן להשאיר בתשובהכם סכומים סופיים.

מהנתנו בשאלת, כאשר תחנה מנסה לשדר מחדש היא קודם בודקת את תפוזת התדרים, אם קיימים תדר פנוים התחנה תשדר אליו. אחרת, הניסיון נכשל.

כלומר, ההסתברות להצלחת שידור הודעה תליה בכך שהתדר שבו מתבצע השידור יהיה פנווי ולא יתקייםו התנגשויות עם הודעות אחרות המשודרות בתדרים אחרים.

מכאן, ההסתברות להצלחה, מחושבת כסכום ההסתברויות שאין התנגשות עבור כל מסדר אפשרי של תחנות שישדרו בהצלחה הקודם לכך:

$$P_{succ} = \sum_{i=0}^{i=k-1} P_i(kT)$$

כאשר, $(P_i(kT))$ היא ההסתברות ש- i תחנות שישדרו בהצלחה לפני התחנה הנוכחית. ההסתברות להצלחה של שידור מפולגת לפי פילוג פואסוני, שמתאים במיוחד לתיאור הסתברויות של מספר אירועים המתרחשים בפרק זמן נתון T בקצב הופעה g. בפרט, פרמטר לממד לפילוג פואסוני מייצג את קצב ההתרחשויות הממוצע של האירועים, והוא מוגדר כאן כ- gkT , שמשמעותו מספר השידורים הצפוי בתדר המסוים במהלך הזמן T.

נשתמש בנוסחתה ה- PMF של ההתפלגות הפואסונית, כפי שלמדנו בהסתברות, ונקבל:

$$P_{succ} = \sum_{i=0}^{i=k-1} P_i(kT) = \sum_{i=0}^{i=k-1} \frac{e^{-gkT} \cdot (gkT)^i}{i!}$$

ד. האם נצילות המערכת לאחר השדרוג גדולה, קטנה או שווה לזה שמצאתם בסעיף ב'?
הסבירו.

בפרוטוקול הקודם, מסעיף ב, כל תחנה ניסתה לשדר בכל תדר זמין. זה יצר כמות גבוהה של התנגשויות שהובילו לנצלות נמוכה יחסית בהשוואה למערכת לאחר השדרוג כך שהעומס בה היה מחולק בצורה טוביה יותר בקצב הבדיקה המקדים, מה שמקטין את ההסתברות להתנגשות ומעלה את סיכוי ההצלחה. למעשה, השיפור בנצלות נובע מהעליה בהסתברות ההצלחה P_{succ} , שימושו זהה יותר שידורים מוצלחים ופחות התנגשויות.

ונכל גם לחשב את נצילות המערכת לאחר השדרוג, ולהשוו אותה לנצילות לפני השדרוג.

קל לחשב את זה, יש בידינו את P_{succ} החדש, נעשה אותו תהליך של סעיף ב עם קצת שינויים.

לסיום, נצילות המערכת לאחר השדרוג גדולה יותר מהנצילות לפני השדרוג, וזה נובע מהפחתת התנגשויות ושיפור היכולת של המערכת לנצל את התדרים הפנויים בצורה אופטימלית.

ה. הוצע להחליף את הפרוטוקול בשימוש מ- Slotted ALOHA ל Pure ALOHA, עם חיריצים באורך זמן השידור של הودעה T_k , המסונכרים בין התחנות השונות ובין התדרים השונים. השדרוג שהוצע לפני סעיף ג' עדין בשימוש. האם הנצלות תגדל, תקטן או לא תשתנה בעקבות החלפת הפרוטוקול? הסבירו.

אם עברו מ Pure ALOHA ל Slotted ALOHA הניתנת עלולה להפגיע ולא להשתפר. למרות ש- Slotted ALOHA מצמצם את כמות התנגשויות בכך שהוא מאפשר שידור רק בתחלת חרץ זמן קבוע, הדבר גם מגביל את מספר התחנות שיכלות לשדר בכל רגע נתון. המשמעות היא שהמערכת יכולה להיות פחותה עיליה מאשר במצב של Pure ALOHA, שבו תחנות יכולות לשדר בכל זמן נתון ללא הגבלת חריצי זמן. בנוסף של דבר, המגבלה על שידור ב Slotted ALOHA- יכולה להוביל לכך שנצלות המערכת לא תשתperf ואולי תקטן בהשוואה במצב של Pure ALOHA. יותר מזה, עלותה להיווצר התנגשויות שלא היו קיימות קודם, כיוון שכל התחנות יכולות לשדר רק בתחלת חרץ זמן, יתכן מצב שבו מספר תחנות ינסו לשדר בדיק באותו חרץ זמן, מה שיוביל להתנגשות מיידית ביניהן, גם אם הן יכולות לשדר בהצלחה ב Pure ALOHA ללא מגבלת חריצי הזמן.

לכ, למרות שהשדרוג הקודם עדין זמן וממשיך לשפר את הסיכויים להצלחת השידור, המעבר ל Slotted ALOHA עשוי להפחית את נצילות המערכת עקב המגבלה על זמן השידור, דבר שעלול להגביל את מספר התחנות שיכלות לנצל את התדרים הפנויים בצורה אופטימלית.

שאלה 3 (תורת התורים)

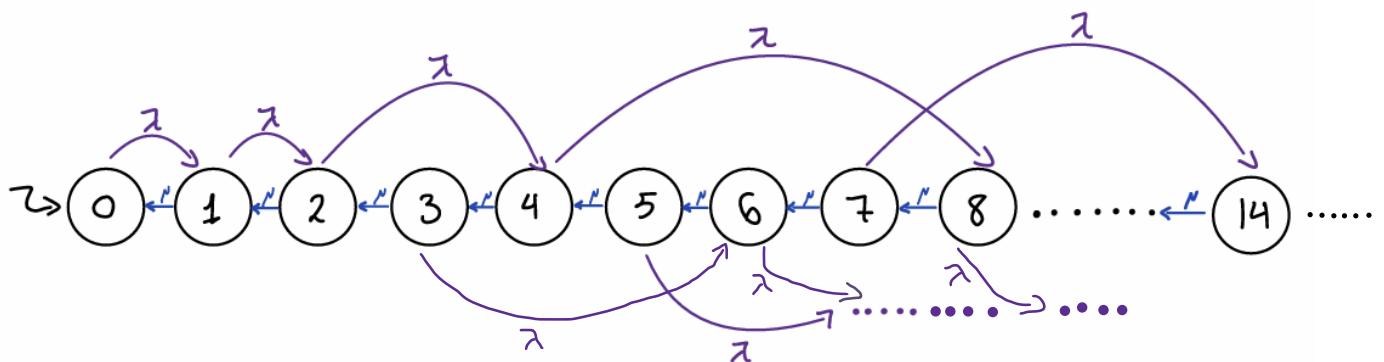
נתון תור FIFO עם הנתונים הבאים:

- הודיעות כניסה לתור בקבוצות כאשר פרק הזמן בין הגעות של קבוצות עוקבות מתפלג אקספוננציאלית עם פרמטר $\lambda > \mu$ הודיעות חדשות כניסה לתור באופן הבא:
 - כאשר $\lambda < \mu$ הודיעות בתור, ככלומר הودעה בזדמנות מקבלת שירות ו1 – a הודיעות נוספות נוספות ממתינות לשירות, קבוצה של a הודיעות תיכנס לתור. **לדוגמה:** אם במערכת יש 3 הודיעות, תגיע קבוצה של שלוש הודיעות.
 - אם $\lambda = \mu$ הודעה בזדמנות תיכנס למערכת.
 - יש שרף יחיד המטפל בהודיעות בזמן אקספוננציאלי עם פרמטר $\lambda > \mu$.

נניח תחילת כי התור אינסופי.

א. שרטטו דיאגרמת מצבים המתאימה לבעה זו

מערכת זו מתארת את כמות ה הודעות שנמצאות בתור המערכת, מצב 0 מייצג שאין הודעות בתור, מצב 1 מייצג שיש הודעה אחת, וכן הלאה.



דיאגרמה זו מתארת שכל הגעה היא קבוצה בגודל התור,(Clustered Queueing) כולם קפיצות כפולות. והיא אינסופית, ככלומר ניתן המשיך ולהוסיף הודיעות ללא הגבלה. הדיאגרמה ממשיכה באופן דומה לכל מצב נוסף, כך שהמעברים בין המצבים ממשיכים באותו אופן עם קצב ההכנסה λ וקצב הוצאה μ כפי שנעשה במצבים הקודמים.

ב. רשמו מערכת משוואות בעזרתן ניתן לחשב את ההסתברויות P_i לכל $i = 0, 1, 2, \dots$ כך שיהיו i הודיעות במערכת במצב יcit. אין צורך לפתור את מערכת המשוואות.

P_i : ההסתברויות שהמערכת נמצאת במצב שבו יש בדיקן i הודיעות בתור.
מהדיאגרמה ששרטטנו נסיק:

מקרה הבסיסי:

עבור $0 = i$:

$$\lambda \cdot P_0 = \mu \cdot p_1$$

עבור $1 = i$:

$$(\lambda + \mu) \cdot P_1 = \lambda \cdot P_0 + \mu \cdot P_2$$

עבורן כללי:

עבורן זוגי כר ש- $0 \neq i$:

$$(\lambda + \mu) * P_i = \lambda * P_{\frac{i}{2}} + \mu * P_{i+1}$$

עבורן אי-זוגי כר ש- $i \neq 1$:

$$(\lambda + \mu) * P_i = \mu * P_{i+1}$$

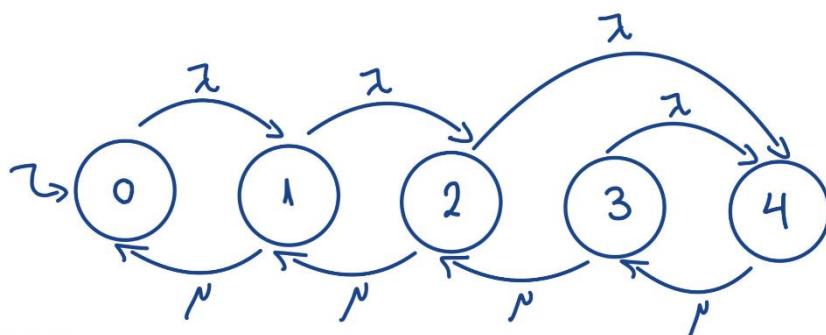
ג. האם כאשר $\lambda > \mu$ המערכת תהיה במצב יציב?

לא, במצב הרזה, המערכת לא תהיה במצב יציב.

כאשר $\lambda > \mu$, המשמעות היא שקצב הגעת ההודעות למערכת קטן מקצב הטיפול בהן. עם זאת, המערכת שבה ההודעות נכנסות בקצבות גדולות ובקפיאות כפולות, המערכת לא תהיה במצב יציב. זאת מכיוון שבכל פעם שנכנסת קבוצה של הודעות, היא מוסיפה כמות גדולה של הודעות לתור, והשתת לא מספיק לטפל בכלן לפני שקבוצה נוספת מגיעה. כתוצאה לכך, מספר ההודעות בתור ימשיך לגדול עם הזמן, מה שגורם למערכת להיכנס למצב של חוסר יציבות.

כעת נניח תור **סופי בגודל 4** (כאשר התור מלא הודעה אחת תהיה בתהליך שירות ושלוש הודעות ימתינו בתור לשירות)

- כאשר קבוצת הודעות מגיעה. הודעות אשר אין להם מקום בתור תיזרקנה (למשל אם המערכת 3 הודעות וקבוצה של 3 הודעות תגענה. הודעה אחת תיכנס לתור ושניים יזרקו)
- ד. שרטטו דיאגרמת מצבים המתאימה לבעה זו.



כל אחד מהמצבים הנ"ל מתאר מקרה כלשהו:

- 0 ← מייד על כר שאין הודעות בתור (תור ריק).
- 1 ← מייצג שיש הודעה אחת בתור.
- 2 ← מייצג שיש שתי הודעות בתור.
- 3 ← מייצג שיש שלוש הודעות בתור.
- 4 ← מייצג שיש ארבע הודעות בתור (תור מלא).

ה. רשמו מערכת משוואות בעזרתן ניתן לחשב את ההסתברויות P_i לכל $i = 0, 1, 2, \dots$ כך שיחיינו i הודות במערכת במצב יציב. פתרו את מערכת המשוואות.

במערכת תורים, במצב יציב, ישנו איזון בין קצב הכניסה של ההודעות למערכת לבין קצב הייציאה שלהן.

בשלב הראשון נרשום את משוואות האיזון עבור כל מצב במערכת.

$$\begin{aligned} P_0 \cdot \lambda &= P_1 \cdot \mu \\ P_1 \cdot \lambda &= P_2 \cdot \mu \\ P_2 \cdot \lambda &= P_3 \cdot \mu \\ P_3 \cdot \lambda + P_2 \cdot \lambda &= P_4 \cdot \mu \end{aligned}$$

כפי שלמדנו איחוד ההסתברויות שווה לאחד.

נשים לב ש- $0 = \dots = P_5$, וכך:

$$P_0 + P_1 + P_2 + P_3 + P_4 = 1$$

בשלב הראשון, נתחילה לבדוק את ההסתברויות מהשוואות שמצאים: מהמשואה הראשונה:

$$P_1 = \frac{\lambda}{\mu} P_0$$

מהמשואה השנייה:

$$P_2 = \frac{\lambda}{\mu} P_1 = \left(\frac{\lambda}{\mu}\right)^2 \cdot P_0$$

מהמשואה השלישית:

$$P_3 = \frac{\lambda}{\mu} P_2 = \left(\frac{\lambda}{\mu}\right)^3 \cdot P_0$$

מהמשואה הרביעית:

$$P_4 = \frac{\lambda \cdot (P_2 + P_3)}{\mu} = \frac{\lambda}{\mu} \cdot \left(\left(\frac{\lambda}{\mu}\right)^2 + \left(\frac{\lambda}{\mu}\right)^3 \right) \cdot P_0$$

לאחר מכן, נציב את כל ההסתברויות האלו במשואה, ונקבל:

$$P_0 + \frac{\lambda}{\mu} P_0 + \left(\frac{\lambda}{\mu}\right)^2 \cdot P_0 + \left(\frac{\lambda}{\mu}\right)^3 \cdot P_0 + \frac{\lambda}{\mu} \cdot \left(\left(\frac{\lambda}{\mu}\right)^2 + \left(\frac{\lambda}{\mu}\right)^3 \right) \cdot P_0 = 1$$

ונמצא את P_0 כגורם משותף:

$$P_0 \cdot \left(1 + \frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2 + 2 \cdot \left(\frac{\lambda}{\mu}\right)^3 + \left(\frac{\lambda}{\mu}\right)^4 \right) = 1$$

ובודד את P_0 ונקבל:

$$P_0 = \frac{1}{\left(1 + \frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2 + 2 \cdot \left(\frac{\lambda}{\mu}\right)^3 + \left(\frac{\lambda}{\mu}\right)^4 \right)}$$

לבסוף, לאחר שמצאנו את P_0 , נוכל להשתמש בו כדי לחשב את $P_1/P_2/P_3/P_4$ בעזרת המשוואות שנגזרו בשלב הראשון.

מחשב את P_1 לדוגמה, ושאר ההסתברויות נחשב אותן באופן דומה.

$$P_1 = \frac{\lambda}{\mu} P_0 = \frac{\lambda}{\mu} \cdot \frac{1}{\left(1 + \frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2 + 2 \cdot \left(\frac{\lambda}{\mu}\right)^3 + \left(\frac{\lambda}{\mu}\right)^4\right)}$$

1. מהם התנאים למקצב יציב?

המערכת תגיע למקצב הרכינה אולם קצב החזאה נ. בתנאי זה, מספר ההודעות בתור לא יגדל ללא גבול והמערכת תתכנס למקצב יציב שבו ההודעות נכנסות ויוצאות בצורה מאוזנת. במקרה שלנו, מספר הממצבים במערכת הוא סופי, לכן התנאי הזה יבטיח שהמערכת לא תעבור את הגבול של מספר הממצבים ותישאר יציבה.

2. מהו הקצב הממוצע של הודעות אשר נדרקוט מהמערכת?

נבחן ש- הודעות נדרקוט מהמערכת כאשר התור מלא, ואין אפשרות להכניס הודעות חדשות. כלומר, המערכת נמצאת במצבים שבהם התור כבר מלא ואין מקום להודעות נוספות, וכך כל הودעה חדשה שmagua נדרקת. לפיה הדיאגרמה של מסעיף ד', הודעות נדרקוט כאשר אנחנו נמצאים במצבים 3 או 4, עבור מצב 3, ההנחה היא שזוג הודעות נדרקוט כי הודעה אחת כניסה והטור כבר מלא, וכך נכפיל את הסתברות P_3 בקצב הגעה ובמספר ה Hodoutes הנדרקוט. באותו אופן נתייחס למצב הרביעי, ההנחה כאן היא ש-4 הודעות נדרקוט, וכך נכפיל את הסתברות P_4 בקצב הגעה ובמספר ה Hodoutes הנדרקוט. כלומר, נקבל שהקצב הכלל של ה Hodoutes שנדרקוט מהמערכת ניתן על ידי:

$$P_3 \cdot 2\lambda + P_4 \cdot 4\lambda$$

ונכל המשיך ולהוציא את גורם משותף, ולפשט את הביטוי עוד יותר בלהציב את P_3 ו- P_4 שמצאו במסעיף ה.

3. מהו מושך הזמן הטיפול הממוצע במערכת של ה Hodoute אשר נכנסה לתור?

כדי לחשב את זמן הטיפול הממוצע במערכת של ה Hodoute אשר נכנסה לתור, נשתמש במשפט ליטל. נחשב את מספר ה Hodoutes הממוצע במערכת על ידי שקלול הסתברויות שכל ה Hodoute נמצאת במצבים השונים בטור:

$$E(N) = P_1 + 2P_2 + 3P_3 + 4P_4$$

לאחר מכן, נחשב את הקצב הממוצע של ה Hodoutes הנכנסות למערכת תוך התחשבות בכך שהמערכת יכולה להיות מלאה במצבים מסוימים:

$$E(\lambda) = \lambda \cdot (P_0 + P_1 + 2P_2 + P_3)$$

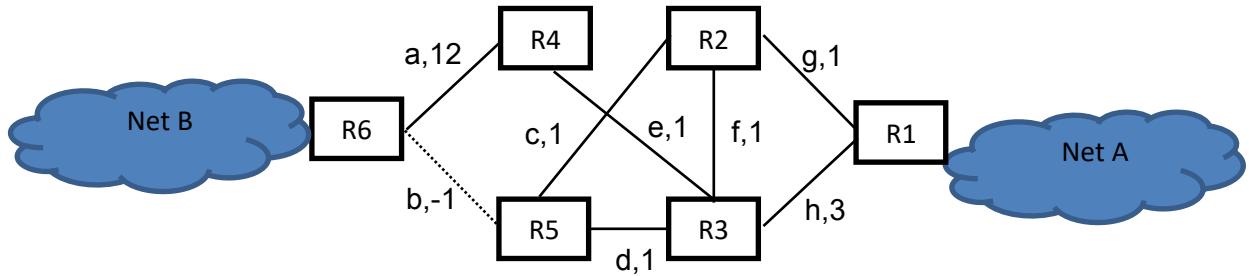
לפי משפט ליטל:

$$E(T) = \frac{E(N)}{E(\lambda)} = \frac{P_1 + 2P_2 + 3P_3 + 4P_4}{\lambda \cdot (P_0 + P_1 + 2P_2 + P_3)}$$

לסיום, זמן הטיפול הממוצע בה Hodoute הנכנסת לתור נקבע על ידי מספר ה Hodoutes הממוצע במערכת חלקי' קצב ה Hodoutes הנכנסות למערכת בהצלחה, וזאת בהתאם למשפט ליטל, שmbtich שהמערכת תישאר מאוזנת ויעלה.

שאלה 4 (ניתוב)

התבונן/ני ברשת הבאה:



ברשת הנתונה הורץ פרוטוקול ניתוב **link state** המבוסס מסלולים קצרים ביותר. (אות ליד הקשת מסמן את זהות הקשת והמספר את משקל הקשת. כולם 12, a מייצג את קשת a בעל משקל 12).

א. הצע אלגוריתם למציאת מסלולים קצרים ביותר עבור פרוטוקול הניתוב. נמק/י

```

INITIALIZE-SINGLE-SOURCE( $G, s$ )
1   for each vertex  $v \in G.V$ 
2      $v.d = \infty$ 
3      $v.\pi = \text{NIL}$ 
4    $s.d = 0$ 

-----
RELAX( $u, v, w$ )
1   if  $v.d > u.d + w(u, v)$ 
2      $v.d = u.d + w(u, v)$ 
3      $v.\pi = u$ 

-----
BELLMAN-FORD( $G, w, s$ )
1   INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   for  $i = 1$  to  $|G.V| - 1$ 
3     for each edge  $(u, v) \in G.E$ 
4       RELAX( $u, v, w$ )
5     for each edge  $(u, v) \in G.E$ 
6       if  $v.d > u.d + w(u, v)$ 
7         return FALSE
8   return TRUE

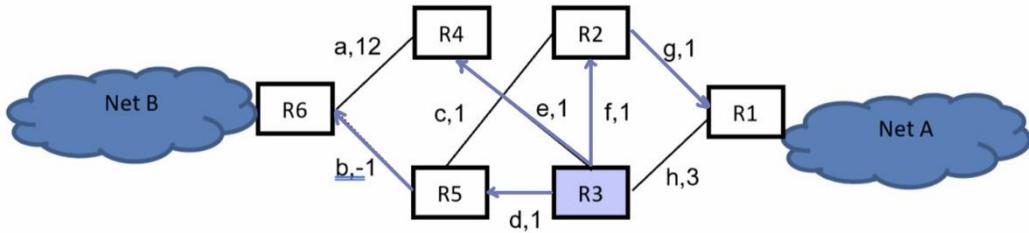
```

מכיון שישנה קשת עם משקל שלילי ברשת (הקשת b), אני מציעה להשתמש באלגוריתם בלמן-פורד, מכיוון שהוא מתאים למציאת מסלולים קצרים ביותר גם במקרים שבהן יש קשותות עם משקלים שליליים. אם לא היה ברשת קשותות שליליות הייתה מעדיפה דיקטורה כי הוא יותר יעיל בזמן ריצה מאשר בלמן-פורד.

בסוף הרצת הפרוטוקול כל אחד מהנתבים מחשב עץ פורש (מסלולים קצרים ביותר) אל שאר הנתבים.

ב. מהו העץ פורש המוחושב מנתב R3?

עץ פורש המוחושב מנתב R3 הוא אוסף של המסלולים הקצרים ביותר מ-R3-ל יתר הנתבים. כמובן הוא תת-גרף קשור וחסר מעגלים, הכלול את כל הנתבים, ו R3 הוא הנקודה ההתחלתית.



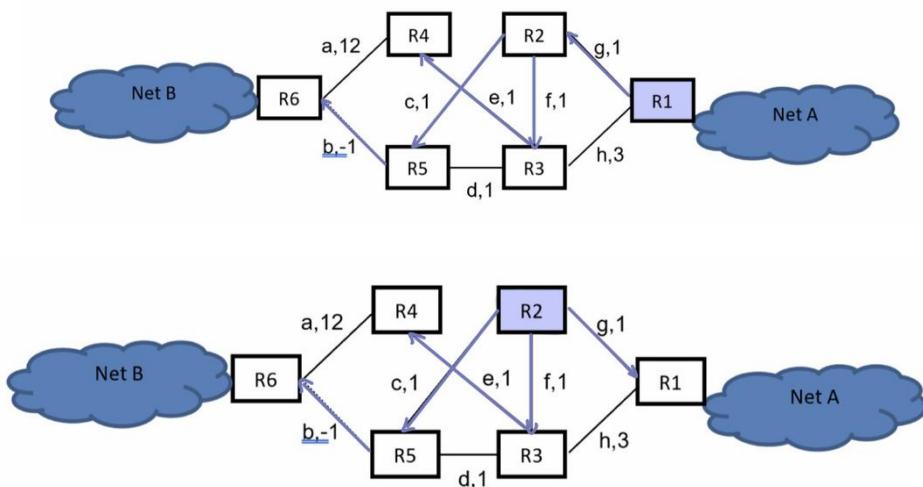
ג. האם העץ הפורש המוחושב ע"י כל אחד מהנתבים הינו עצ פורש מינימלי? נמק/י

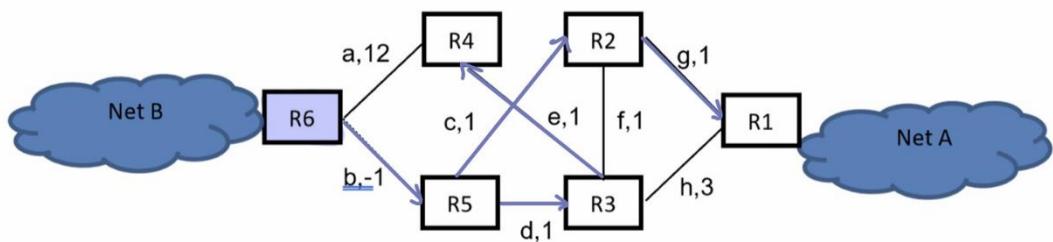
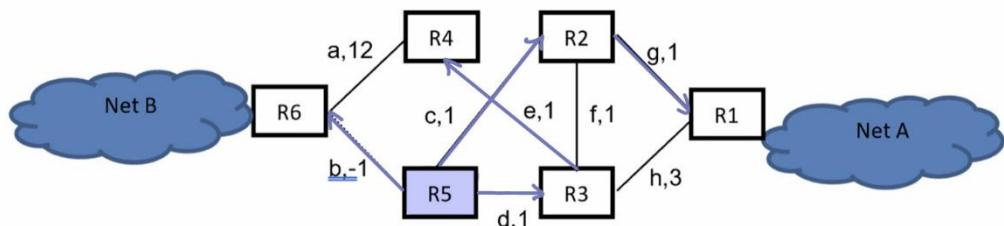
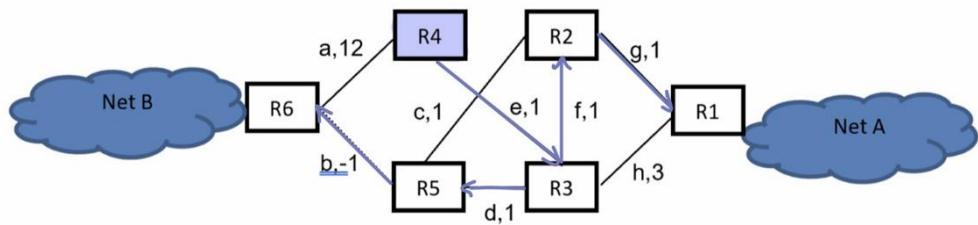
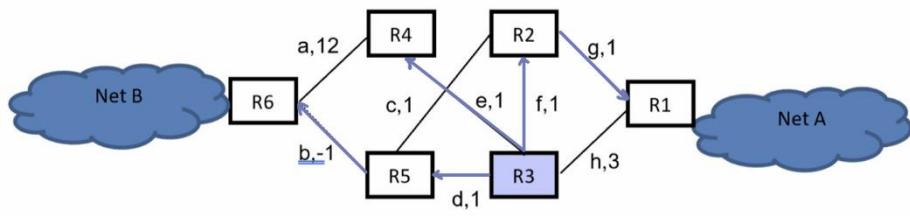
העץ הפורש המוחושב על ידי כל אחד מהנתבים ברשות, אינו בהכרח עצ פורש מינימלי, כמובן, הוא לא תמיד יהיה העץ בעל המשקל הכלול הנמוך ביותר שמחבר את כל הצמתים ברשות. הסיבה לכך היא שאלגוריתם בלמן-פורד מתמקד במציאת המסלול הקצר ביותר מנקודות מוצא נתונה לכל אחד מרצטטים האחרים, ולאו דווקא בבחירה עצ פורש מינימלי. במקרה שלנו, אחרי שמצאת את העצים הפורשים מכל נtab בסעיף (ד), התגלה שככל אחד מהעצים האלה הוא עצ פורש מינימלי. הסיבה לכך היא שהקשתות שנבחרו מחברות את כל הצמתים בצורה כזו שהמשקל הכלול של הקשתות הוא הנמוך ביותר האפשרי. במקרה זה, כל אחד מהנתבים ברשות הצלח לחשב עצ פורש מינימלי, למראות שאין הכרח לכך מבחינה תאורטית.

נגידר את סוג הקשתות הבאים:

- 1) קשת קריטית – הינה קשת שמשתתפת בעץ הפורש של כל אחד מהנתבים.
 - 2) קשת חשובה - הינה קשת שמשתתפת בעץ הפורש של נtab כלשהו ואינה קריטית.
 - 3) קשת גיבוי – הינה קשת שלא משתתפת בשום עצ פורש.
- ד. עברו כל אחד מהקשתות בגרף, מהו סוג הקשת? נמק/י

כדי לסוג את הקשתות, נדרש את העץ הפורש מכל נתיב.





קשתות קרייטיות – g / e / b : קשתות אלו, כפי שניתן לראות משתתפות בכל העצים.

קשתות גיבוי – h / a : קשתות אלו לא משתתפות בשום עץ.

קשתות חשובות – c / f / d :

c – קשת זו לא קרייטית (לא מופיעה בעץ של R4) , והיא מופיעה למשל בעץ של R6 .

f – קשת זו לא קרייטית (לא מופיעה בעץ של R6) , והיא מופיעה למשל בעץ של R4 .

d – קשת זו לא קרייטית (לא מופיעה בעץ של R2) , והיא מופיעה למשל בעץ של R6 .

ה. בעת הניחו כי קוו על המקווקו נפל. תארו הילך הפצת נפילת הקו לכל הנتابים בראשת. כמה הודיעות נשלחות בראשת על מנת לתקן את טבלאות הניתוב? אילו נתבים יישנו את טבלאות הניתוב? נמק!

1. זיהוי הכשל והפצת המידע:

R5 שולח ל R4 דרך הקשת a ו- R5 שולח ל R2 דרך הקשת c ול R3 דרך הקשת d. ההודעות מועברות מהנתבים R6 ו- R5 לנתקים השכנים לה. השכנים מעריכים זמינה יותר, ולכן יש לעדכן את טבלאות הניתוב בהתאם. אולם, אם לא ניתן לשלוח מessage מ- R5 ל- R6, על מנת לא לפגוע בפעולת R5, ניתן לשלוח מessage מ- R5 ל- R3 דרך הקשת d.

2. הפצת המידע לשכנים נוספים:

לאחר שהנתבים הראשונים (R1, R2, R3) קיבלו את הودעות ה LSA על נפילות הקשת ב, הם ממשיכים להפיץ את המידע לשכנים שלהם, כדי לוודא שהמידע הגיע לכל הנתבים בראשת R4. מעביר את ההודעה ל R3 דרך הקשת e, מעביר ל R1 דרך הקשת d וLR3 דרך הקשת f, וLR3 מעביר ל R1 דרך הקשת h.

בפרוטוקול Link State, ישנו כל חשיבותו שמוני מנתב להעביר את המידע חזרה לשכנים קיבל את המידע, מה שמנוע יצירת לויאות ומיעיל את תהליך הפצת המידע.

3. עדכון טבלאות הניתוב:

לאחר שהמידע על נפילת הקשת b התפשט בכל הרשות, הנتابים מתחילה לעדכן את טבלאות הניתוב שלהם בהתאם למידע החדש. הנتابים שנפגעו ישירות מההசל, ככלומר R5 ו R6, יעדכו מיד את הטבלאות שלהם, מכיוון שהיו מחוברים ישירות לקשת b. בוגוסף R3, ישנה את המסלול ל R6, כך שיתבצע CUT דרכ R5 במקום R4. יעדכן את המסלול שלו ל R6 לנتاب ישיר דרכ הקשת a ו R2. ישנה את המסלול ל R6 כך שייעבור דרך R3 במקום R5. נתב R1 לא יבצע שינוי, מכיוון שהמסלולים שלו לא הושפעו מנפילת הקשת b.

לאחר סיום תהליך העדכון, כל הנتابים שהושפעו ישירות או בעקבות מהנפילה, עדכנו את טבלאות הניתוב שלהם כדי לשקף את השינויים ברשות. בסך הכל, נשלחו 7 הודעות LSA ברשות כדי לודא שכל הנتابים מעודכנים. **הנתבים שעדכנו את טבלאות הניתוב שלהם הם R5, R6, R2, R3, R4, R5, R6**, בעוד שנטב R1 לא ביצע שינוי כלשהו. תהליך זה מבטיח שהרשות תחזור לתפקיד מלא בצוראה מהירה ויעילה לאחר נפילת הקשת.

icut ha-nihiho ci birshet ha-netanya (col keshet d) horz protokol ni-tuwb **distance vector**. ham ha-protokol yitcanu v-tbalot ha-nihiho yi-tibzu? b-makrah v-la-hizi'u dror la-takan at ha-protokol. nmk/

טבלאות הניתוב בראשת הנתונה, הכוללת את הקשת s עם משקל שלילי, לא יכולו להתייצב בפרוטוקול Distance Vector. הדבר נובע מכך שמשקל שלילי בקשת עלול להוביל לביעות חמורות בניטו. קשת זו עלולה לגרום לנODEים לדוח שוב ושוב על עליות נמוכות יותר ויותר ליעדים אחרים. אלגוריתם Distance Vector מתבסס על עדכוני עלות ממחזור למחזור, וכך אשר קיימת קשת עם משקל שלילי, הערכיהם בטבלאות הניתוב יכולים להמשיך ולהתעדכן כלפי מטה בצורה אינסופית. לדוגמה, כאשר R6 מעדכן את R5 לגבי מסלול זול יותר לעד מסויים דרך הקשת s , בתורו שולח עדכון דומה חזרה ל R6 וחוזר חלילה. תהליך זה

מכניס את הנתבים לולאה אינסופית של עדכנים, וגורם לכך שטבלאות הניתוב לא יתיצבו.

הצעת פתרון:

כדי לפתור בעיה זו ולהימנע מלולאות אינסופיות, ניתן ליישם את מנגנון Poison Reverse Split Horizon בטכניתה זו, כאשר נתב מקבל מידע על מסלול דרך נתב אחר, הוא מחזיר את המידע עם ערך אינסופי, שמצוין שהמסלול איינו תקף . לדוגמה, אם R6 שולח ל R5 את ה distance vector שלו, R5 מזהה שהמסלול ל R5 דרך R6 נלמד ממנו עצמו, ולכן מחזיר את המידע עם ערך אינסופי. בדרך זו נמנעים מלולאות של לולאות אינסופיות, והפרוטוקול מצליח להתייצב בצורה נכונה.

שאלה 5 (ARQ)

תחנה A שולחת הודעות לתחנה B בעזרת Stop and Wait משופר. כדי לשפר את הפרטוקול הוחלט שהתחנה A תshedר כל פעם עם שתי הודעות עוקבות ברכף, בק לאחר שתחנה A יודעת מה קרה עם כל אחת משתי ההודעות, תחנה A מחליטה איזה הודעות לשדר ומתחילה לשדר אותן מיד:

- אם להודעה הראשונה מבין שתי ההודעות פקע $timeout$ תחנה A תshedר שוב את שתי ההודעות (אחרי שתחנה A יודעת מה קורה עם ההודעה השנייה).
- אם להודעה השנייה בחילון פקע $timeout$ והתקבל חיווי עבור ההודעה הראשונה, תחנה A תשדר את ההודעה השנייה ואת ההודעה הבאה אחרת.
- אם התקבל חיווי עבור שתי ההודעות, תחנה A תעבור לשדר את שתי ההודעות הבאות.

עוד נתונים:

- זמן התהפכות בכל ציון הוא t_p
- זמן שידור כל הודעה הוא t_i
- $2t_p \geq t_i$
- זמן העבודה בכל התחנות T ניחים
- זמן שידור-ACK נניחים
- לתחנה A תמיד יש מה לשדר
- בהסתברות p הודעות נפלות, $1 < p < 0$
- חיויים מגיעים תמיד
- זמן $timeout$ הוא זהה עבור כל הודעה ונמדד מסוף שידור הודעה

א. מהו ה- $timeout$ המינימלי שתמיד מונע שידורים מיותרים?

ה- $timeout$ המינימלי שמנע תמיד שידורים מיותרים הוא $2t_p$ שזה פעמיים זמן ההתהפכות של הודעה לכל ציון. הסיבה לכך היא שתחנת A צריכה להמתין מספיק זמן לקבלת אישור חוזרת מתחנת B לפני שתshedר שוב את הודעה. אם ה- $timeout$ קצר מדי, תחנת A עלולה לשדר את הודעה מחדש ללא צורך, מה שיוביל לשידורים מיותרים. $2t_p$ מבטיח שההודעה ואישורה יגיעו בזמן לפני שידור נוסף יבוצע.

בעצמי ועד סוף השאלה הפרוטוקול משתמש ב- $timeout$ שמצאו בסעיף א'.
ב. מהי נצילות הפרטוקול?

בהתבהה שה- $timeout$ מהסעיף הקודם הקודם הוא אכן נכון,
 $Timeout = 2t_p$
נצילות הפרטוקול כפי שמדובר היא היחס בין זמן שידור אופטימי לבין הזמן הכלול הנדרש לשידור ולהמתנה לאישורים.

זמן הנדרש להשלמת מחזור שידור אחד, כולל גם את קבלת האישור חוזרת הוא:

$$T = 2t_i + 2t_p$$

כעת, נעבור לחשב את הזמן שבו הפרטוקול משדר נתונים בצורה אופטימית:

$$(1 \cdot t_i + (1 - 1) \cdot (1 - p)) + ((1 - p)^2 \cdot 2t_p)$$

נסביר איך הגיענו לנוסחה זו,
הסתברות שהשידור הראשון עבר בהצלחה ללא צורך בשידור חוזר היא $(1 - p)^2$,
והשידור מתבצע במקרה שני זמן השידור. לכן, קיבל את הביטוי $(1 - p)^2 \cdot 2t_p$.

בעוד שהסתברות שהשידור הראשון נכשל, אך השידור השני מצליח היא $(p - 1) \cdot p$, והזמן לשידור במקרה זה הוא t_i .
לסיכום,

$$= \frac{((1-p)^2 \cdot 2t_i) + ((1-p) \cdot p \cdot t_i)}{2t_i + 2t_p} \text{ ניצולת}$$

ג. מהו מספר השידורים הממוצע של החבילה השנייה מתחילה הרצת הפרוטוקול?

בכל פעם ששידור הודעה נדרש להישלח שוב בעקבות כישלון, יש הסתברות של ק לכישלון נוספת וסתירות של $p - 1$ להצלחה. כאשר לוקחים בחשבון את הצורך לשדר את הודעה ממספר פעמים עד שהיא תצליח להגיע ליעדה, מספר השידורים הממוצע הדרוש לשם כך הוא:

$$\frac{1}{1-p}$$

כאשר נדרשים לשדר מחדש הודעה נוספת לאחר הכישלון הראשון, כל כישלון נוסף מוסיף ב ממוצע:

$$\frac{p}{1-p}$$

לשידורים הדורשים. לפיכך, מספר השידורים הכולל הנדרש כדי להבטיח שההודעה תעבור בהצלחה, כולל כל השידורים החזריים, הוא:

$$\frac{p+1}{1-p}$$

ד. מהו מספר השידורים הממוצע של הודעה השלישייה מתחילה הרצת הפרוטוקול?
רמז: כשההודעה השלישייה נשלחת בפעם הראשונה, ניתן להבדיל בין המקרה שבו הודעה נשלחת ראשונה למקרה שבו הודעה שנייה מבין שתי ההודעות שנשלחו.

מהרמז שנותן, נבחן שאנו צריכים להגיד את מספר השידורים הממוצע בשני מקרים שונים:

המקרה הראשון - שבו הודעה שנייה נשלחת בהצלחה;
במקרה זה, הודעה השלישייה נשלחת בהסתברות של ,

$$\frac{1}{1-p}$$

שהיא ההסתברות לכך שההודעה תצליח בשידור הראשון שלו.

המקרה השני - שבו הודעה שנייה נכשלה:
במקרה זה, מספר השידורים הממוצע הנדרש להצלחה הוא:

$$\frac{2}{1-p} - 1$$

כי הודעה צריכה להישלח שוב עד שתתקבל בהצלחה.

לכן, מספר השידורים הממוצע של הודעה השלישייה מתחילה הרצת הפרוטוקול הוא:

$$(1-p)\left(\frac{1}{1-p}\right) + p \cdot \left(\frac{2}{1-p} - 1\right)$$

נפשט ביטוי זה ונקבל:

$$= \frac{1-p}{1-p} + \frac{2p}{1-p} - p = 1 + \frac{2p}{1-p} - p$$

ה. בהנחה שחיוים יכולים ליפול, כמה ביטים צריך להקצות לשדה המספר? הסבירו על ידי דוגמה נגדית למה בית אחד פחות מהתשובה לא מספיק

במצב שבו חיויים יכולים ליפול, יש להקצות לפחות 2 ביטים לשדה המספר כדי להבטיח שתחנתת הקליטה תוכל לבדוק בין הودעה חדשה לבין הודעה חוזרת. לדוגמה, אם נשתמש רק בבית אחד לשדה המספר, התחנה לא תוכל לבדוק בין שתי הודעות עוקבות אם החוויי להודעה הראשונה נופל. נניח שההודעה הראשונה נשלחה עם מספר 0, והתחנה הקולטת קיבלה אותה והחזירה חוווי. בעת ההודעה השנייה נשלחת עם מספר 1, אך החוויי שלה נופל ולא מגיע לשולח. השולח, שלא קיבל את החוויי, ישדר את ההודעה שוב עם מספר 1. תחנתת הקליטה, שכבר קיבל הודעה עם מספר 1, לא תוכל לבדוק אם מדובר בהודעה חדשה או בשידור חוזר, ותחשוב שהיא קיבלה את אותה הודעה שוב.

לעומת זאת, עם 2 ביטים, ניתן להקצות מספרים בין 0 ל-3, מה שמאפשר לתחנות לאסוף ברור את ההודעות ולהימנע מבלבול במקרה של נפילת חוווי. לפחות מ-2 ביטים לא יספיק, מכיוון שלא יהיה מספיק טווח למספרים כדי לבדוק בין ההודעות במקרה של כישלון חוווי.

לסיכום, לפחות מ-2 ביטים לא יספיקו, מכיוון שלא יהיה מספיק טווח למספרים כדי לבדוק בין ההודעות במקרה של נפילת חוווי, מה שיכל להוביל לבלבול ולטעויות בתקשות.

שאלה 6 (TCP)

MRIADOK מעוניין להקים מעבדה חדשה באוניברסיטה. לצורך כך, הוא רכש שני מחשבים חדישים וקיים ביניהם באמצעות כבל רשת מתќדם. MRIADOK מתכוון לבצע ניסויים רגילים במיוחד, ועל כן חשוב לו לדעת במדויק מהם הפרמטרים של הרשת. לאחר מדידות ושיקולים רבים, MRIADOK מחליט לעבוד תחת ההנחהות הבאות:

- המחשבים מחוברים ביניהם בכבל שמהירות התפשטות דרכו היא 7 (במטרים לשניה).
- אורך הcabl בין שני המחשבים הוא d מטרים.
- קיבול הקו בין המחשבים הוא C (bihichot של בית לשניה).
- לאחר קבלת הודעה, לכל מחשב נדרש זמן עיבוד של $\frac{[sec]}{KB}$, לפני יכול לבצע פעולות נוספות (כגון מענה להודעה).
- אלא אם צוין אחרת, הודעות וחיוויים מתකבים תמיד באופן תקין.

לצורך ציול המערכת, MRIADOK מעוניין לשלוח הודעה ממחשב אחד למחשב השני, ולאחר מכן בחרה במחשב ממנו לשלוח ההודעה.

A. על מנת לבצע את הניסוי הראשון שלו כראוי, MRIADOK יכול להקצות בדיקות 2 שניות מהרגע שימושו בית הראשוני מהמחשב השולח, ועד שנקלט הבית האחרון של החיווי באותו המחשב. מהו אורך ההודעה המקורי L בביטים, אשר יאפשר שלוחה הודעה וקבלת חיוויי בפרק זמן זה? הניחו כי גודל החיוויי זהה לגודל ההודעה.

כדי לחשב את אורך ההודעה המקורי L בביטים,

ראשית, נחשב את הזמן שהולך להעביר את הביטים מהמחשב הראשון למחשב השני:

$$t_i = \frac{L}{C}$$

בנוסף לזמן העברת הביטים, ניקח בחשבון את הזמן שהולך לפריים לעبور את המרחק בין שני המחשבים בקו התקשרות:

$$t_p = \frac{d}{v}$$

כמו כן, t_i ו- t_p נמדדים בשניות.

כדי להשלים את התהליך, علينا גם להתחשב ב t_{proc} :

$$t_{proc} = \frac{t_{KB} \cdot L}{1000}$$

כאשר t_{KB} הוא הזמן הנדרש לעיבוד הודעה לפני השידור.

כדי לחשב את הזמן הכולל הנדרש להודעה להגעה מהמחשב הראשון למחשב השני ולקבל אישור בחרה, נשתמש בנוסחה הבאה:

$$\tau = t_{RRT} = 2t_i + t_{proc} + 2t_p = \frac{2L}{C} + \frac{t_{KB} \cdot L}{1000} + \frac{2d}{v}$$

לבסוף נבודד את L מהמשוואה הנ"ל,

נעביר אגפים:

$$\tau - \frac{2d}{v} = \frac{2L}{C} + \frac{t_{KB} \cdot L}{1000}$$

ונמצא את L כגורם משותף:

$$\tau - \frac{2d}{v} = \frac{L \cdot (2000 + t_{KB} \cdot C)}{1000 \cdot C}$$

נבודד את L :

$$L = \left(\tau - \frac{2d}{v} \right) \cdot \frac{1000 \cdot C}{(2000 + t_{KB} \cdot C)}$$

את L זהה, הוא אורך ההודעה המקורי לשולח בתנאים הנתונים, מבליל לגורם לאובדן נתונים או שגיאות בתקשורת בין שני המחשבים.

מעתה והלאה הניחו כי כמות המידע (ללא headers) העובר בכל הודעה הוא $L = 1KB$ ושבচিহ্নস্থান এই মান কেন করা হবে? (רק header).

בעבור הניסוי השני, מריידוק מעוניין לשולח $4KB$ של מידע ממחשב A למחשב B. לצורך כך, הוא מעוניין לשות שימוש ב프וטוקול TCP, כפי שנלמד בקורס. השימוש בשתי הtablאות הבאות את הפרטים עבור כל אחת מההודעות שנשלחות בין המחשבים, מתחילת הקמת תקשורת ועד סכום המידע עבור למחשב B (כולל סגירת תקשורת) .. ניתן להזניח את זמני העיבוד, כמו כן, בסעיף זה בלבד הניחו שהפרמטרים של פרוטוקול TCP ($rwnd$, $cwnd$, $ssthresh$) גדולים מספיק ולא מגבלים את תחולות ההודעות.

שימוש לב: מספר השורות בטבלה אינו מעיד בהכרח על מספר ההודעות הדרשיות.

הודעות ממחשב A למחשב B:

Fin	Ack	Syn	Ack Number	Sequence Number	תוקן	#
0	0	1	0	1001	הקמת תקשורת	1
0	1	0	2001	1002	היקמת תקשורת + מידע	2
0	1	0	2001	1002	שליחת הודעה בגודל 7	3
0	1	0	2001	2002	שליחת הודעה בגודל 7	4
0	1	0	2001	3002	שליחת הודעה בגודל 7	5
0	1	0	2001	4002	שליחת הודעה בגודל 7	6

1	1	0	2001	5002	התחלת תהליך סיום ההקשר	7
0	1	0	2002	5003	ACK מגיב לsegueira ההקשר	8

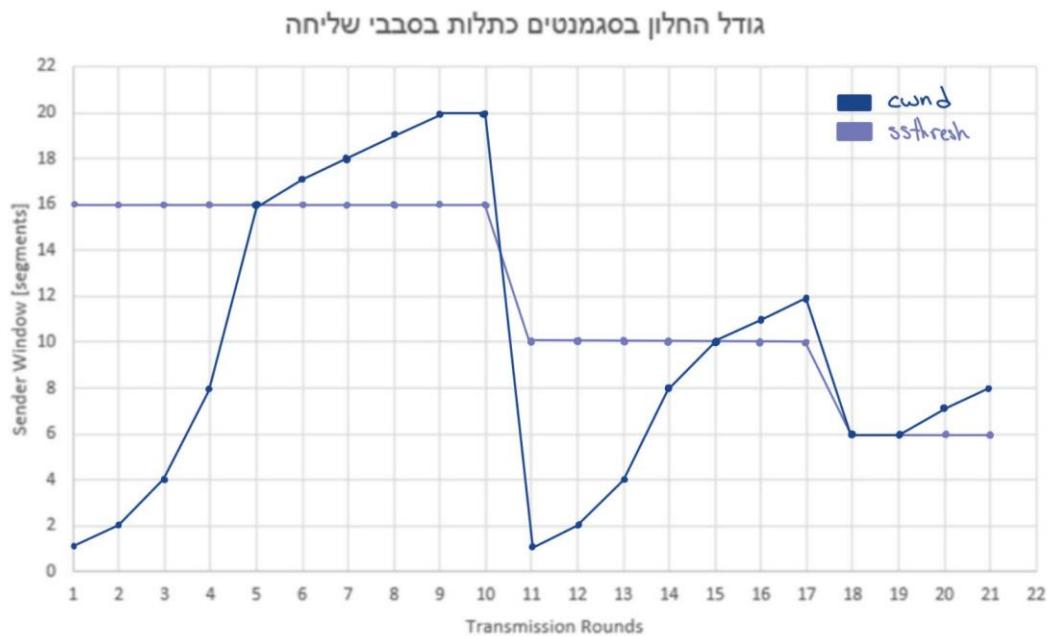
הודעות מחשב B למחשב A:

Fin	Ack	Syn	Ack Number	Sequence Number	תוכן	#
0	1	1	1002	2000	הקמת קשר	1
0	1	0	2002	2001	אישור קבלת נתוניים	2
0	1	0	3002	2001	אישור קבלת נתוניים	3
0	1	0	4002	2001	אישור קבלת נתוניים	4
0	1	0	5002	2001	אישור קבלת הנתוניים	5
0	1	0	5003	2001	שליחת ACK בתגובה להודעה ה F/N	6
1	1	0	5003	2001	שליחת F/N כדי לסגור את הקשר	7

ג. בהמשך, מעוניין מריאדוק לאסוף מידע באופן רציף באמצעות מחשב A, ולהעביר אותו באמצעות פרוטוקול TCP למחשב B. נתוני הפרטוקול:

- בזמן $t = 0$, מתקיים $ssthresh = 16KB$
- בזמן $t = 0$, חלון הגודש הינו בגודל $cwnd = 1KB$
- חלון הקליטה במחשב B הינו בגודל קבוע של $rwnd = 20KB$
- בסעיף זה בלבד ניתן להניח כי מריאדוק החליף את כרטיסי הרשות בכרטיסים מהירים משעווותית, ולכן $t_p \ll t_i$, כלומר זמן המחזור (RTT) הם בקירוב זהים.

شرطנו על- גבי הגרף הבא את $ssthresh$ ואת גודל חלון השילחה בחבילות, כתלות בסביבי השילחה (עד וכול סבב 21), אם נתון שבסבב שליחה 10 כל ההודעות בחלון נפלו, וכן על ההודעה הראשונה בסבב 17 התקבלו 3 חיויים כפולים (dupack).



:Transmission Rounds 1-5

ה $cwnd$ גדל בצורה אקספוננציאלית, כפי שמצופה ב프וטוקול TCP במצב של **Start Stop**. ניתן לראות שבسبب החמייש, ה $cwnd$ הגיע ל-16 סגמנטים, שהוא בדיקת הערך של ה- $ssthresh$.

:Transmission Rounds 6-10

לאחר הגיעו לה $ssthresh$ ה $cwnd$ ממשיך לגודל בצורה לינארית עד שהוא מגיע לערך של 20 סגמנטים בסביבה-10.

זהו גודל חלון השילחה המקורי.

:Transmission Round 11

בסביבה-11, יש נפילה חדה בגודל החלון מ-20 סגמנטים ל-1 סegment בעקבות תקללה במשלוח.

:Transmission Rounds 12-17

גידול חדש לאחר נפילה, ה $cwnd$ מתחילה לגודל שוב בצורה לינארית. הגידול הזה נמשך עד סיבב 17, בו ה $cwnd$ הגיע לערך של כ-12 סגמנטים.

:Transmission Round 18

שוב נראית נפילה חדה בגודל ה $cwnd$. מה שמצויב על כך שקרה כשלון נוסף במשלוח.

:Transmission Rounds 19-21

ה $cwnd$ מתחילה לגודל שוב בצורה לינארית, וממשיכו לגודל יכירה עד סיבב 21.

הסעיפים הבאים אינם מלאים בחלוקת הקודמים של השאלה.

במהלך המתנה לסיום הניסוי, מריידוק מחבר את המחשב הנייד שלו לנット בעדשה (אשר אינו מחובר למחשבים A ו-B לעיל) על-מנת לגלוש באינטרנט. הוא שולח בקשה DHCP ומתקבל את הנתונים הבאים:

- כתובת IP עבור המחשב הנייד: 192.168.0.4
- 192.168.0.1 :Default Gateway
- 255.255.255.0 :Subnet Mask
- כתובת IP של שרת DNS: 192.168.5.2

ד. תארו את התקשרות בין המחשב לשרת DHCP עד להקצת כתובת ה-IP וקבלת הנתונים הנ"ל?

שלב 1: גילוי (DHCP Discover)

כשהמחשב מתחבר לרשת וудין אין לו כתובת IP הוא משדר הודעה DHCP Discover הוועדה זו משודרת לכל המכשירים ברשת המקומיות באמצעות broadcast, עם כתובת IP של היעד 255.255.255.255. כתובת IP של המחשב השולח היא 0.0.0.0, שכן עדין לא הוקצתה לו כתובת IP. בשלב זה, המחשב מחפש שרת DHCP זמינים ברשת.

שלב 2: הצעה (DHCP Offer)

בתגובה להודעת ה-DHCP Discover שרת DHCP מגיב בהודעת DHCP Offer. הודעה זו כוללת כתובת IP מוצעת למחשב, לצד פרטים נוספים כמו ה-default gateway והרשת הנחוצה subnet mask וכתובת שרת DNS. ההצעה נשלחת למחשב כדי להקצתה לו את הגדרות הרשת הנחוצות.

שלב 3: בקשה (DHCP Request)

לאחר קבלת ההצעה, המחשב שולח הודעה בשם DHCP Request הוועדה זו משודרת שוב ברשת המקומיות כbroadcast ומציין כי המחשב מבקש להשתמש בכתובת IP שהוענזה לו ובשאר ההגדרות שנשלחו אליו. בשלב זה הוא הבקשה הרשמית מהשרת לאשר את ההגדרות.

שלב 4: אישור (DHCP Acknowledgment)

לאחר קבלת הודעה DHCP Request מהמחשב, שרת DHCP מאשר את הקצאת כתובת IP וההגדרות על ידי שליחת הודעה DHCP Acknowledgment (ACK). הסופי מאפשר למחשב להתחיל להשתמש בכתובת ה-IP וההגדרות שקיבל.

ה. בהנחה שהמחשב הנייד של מריאדוק הוא המכשיר הראשון המחבר לרשת, וכן לנット יש משק אחד בלבד, כמה מכשירים נוספים יכול היוטר יכול מריאדוק לחבר לרשת? נמקו.

בהנחה שהרשת משתמשת ב- Subnet Mask המאפשרת 256 כתובות IP, קיימות בסך הכל 256 כתובות אפשריות בטוחה הכתובות זהה. עם זאת, לא כל הכתובות זמינים לשימוש עבור מכשירים נוספים ברשת כי:

- כתובות אחת שמורה עבור הרשת עצמה (Network Address), והוא הראשונה בטוויה הכתובות.
- כתובות נוספת שמורה עבור השידור לרשת כליה (Broadcast Address), והוא האחרון בטוויה הכתובות.
- כתובות אחת משמשת כתובות שער ברירת מחדל (Default Gateway) ליציאה לרשותות אחרות.
- כתובות אחת כבר הוקצתה למחשב של מריאדוק, כך שגם היא אינה זמינה.

לכן, מתוך 256 הכתובות, נשארות 252 הכתובות שימושיות. המשמעות היא שברשת זו מריאדוק יכול לחבר עד **252 מכשירים נוספים** בנוסף למכשיר הראשון שכבר קיבל כתובת IP .

. אם מריאדוק יכול לגלוות את כתובת ה-MAC של שרת ה-DNS? נමכו.

לא, מריאדוק לא יכול לגלוות את כתובת ה-MAC של שרת ה-DNS .

הוא יכול לגלוות את כתובת ה MAC של שרת DNS רק אם השרת נמצא באותה רשת מקומית (LAN) . במקרה זה, ניתן להשתמש ב프וטוקול ARP כדי למצוא את כתובת ה- MAC של השרת בהתבסס על כתובת ה IP שלה.

לעומת זאת, אם המחשב של מריאדוק והשרת נמצאים ב subnets- שונים, פרוטוקול ARP לא יוכל לעבור ל subnets המקומי של המחשב. במקרה זה, הבקשה שמריאדוק שולחת תונטו ב תחילת אל ה Default Gateway של הרשת, מה שיגרום לכך שתכתובת ה MAC היחידה שמריאדוק יראה תהיה זו של ה Default Gateway ולא של שרת ה DNS .

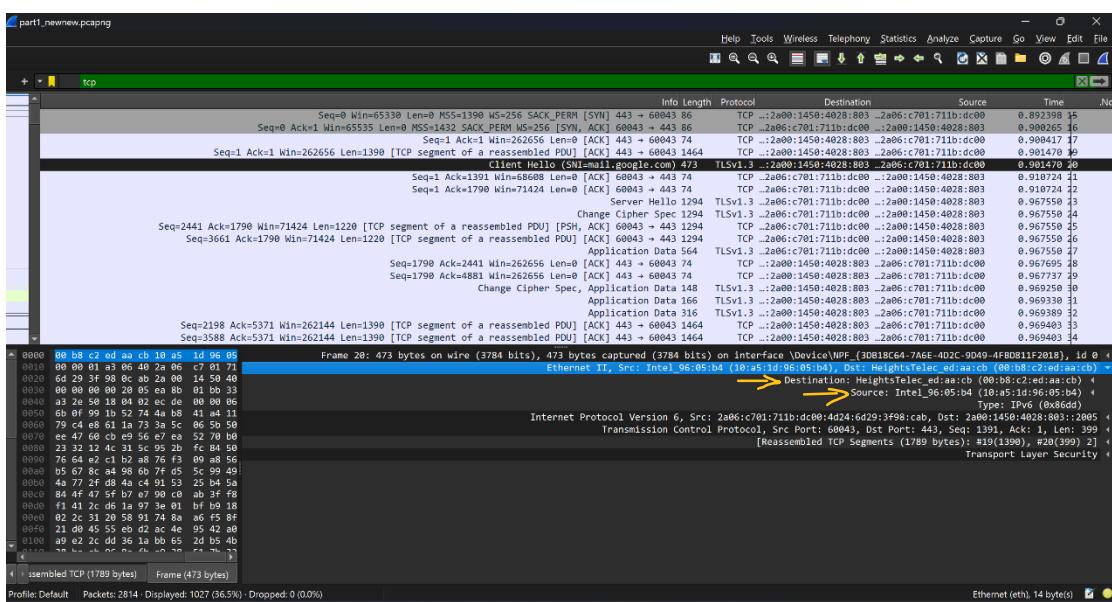
שאלת מעבדה

בשאלה זו נתמקד בפרוטוקול TCP.
עליכם להשתמש עם אפליקציה ה-wireshark לניתוח החבילות השונות העוברות ברשת. עבורי מהשאלות עליכם:

- לצף צילומי מסך של ה-wireshark המאמתים את תשובתכם.
- לספק קבצי pcapן במקומות הנדרשים.
- לנקק את תשובתכם היבט.

חלק א'

בחרו אפליקציה כלשהי המשמשת ב프וטוקול TCP והקליטו ב-wireshark capture. ספקו קובץ pcapng המכיל את הקלטת התעבורה.
א. רישמו את כתובות המקור וכתובת היעד עבור שכבת ה- MAC. למי שיוכות כתובות אילו?



כתובת MAC של המקור: **10:a5:1d:96:05:b4**
זו היא כתובת ה- MAC של כרטיס הרשת במחשב שלי.

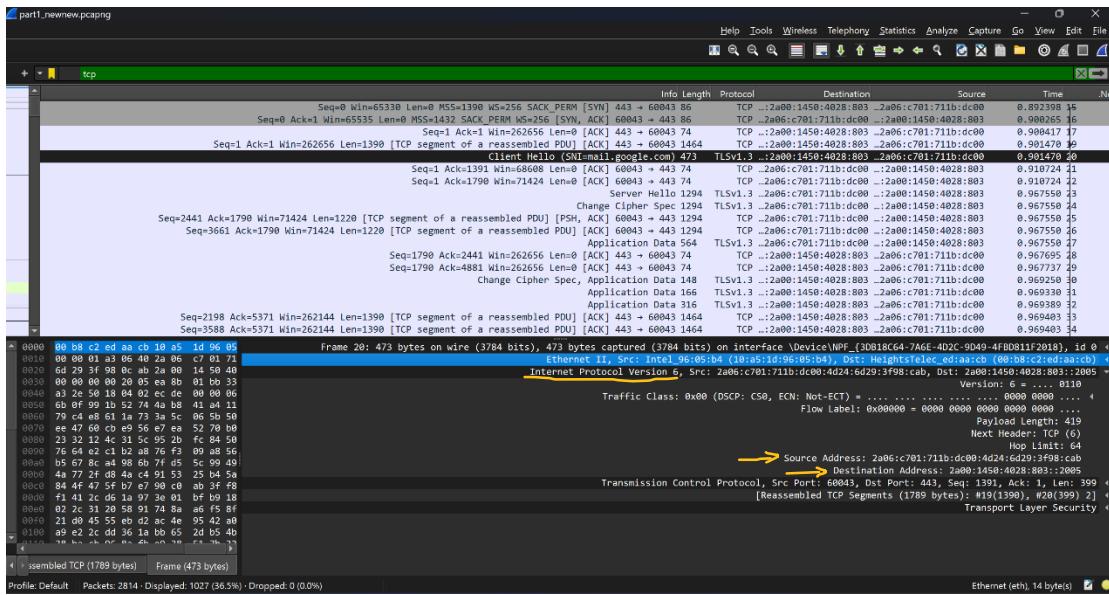
Physical Address : 10-A5-1D-96-05-B4

כתובת MAC של היעד: **00:b8:c2:ed:aa:cb**
זו היא כתובת ה- MAC של הנగב (router) שלי, שתפקידוCIDR בתקשורת זו.

Interface: 10.0.0.8 --- 0x7	Internet Address	Physical Address	Type
	10.0.0.1	00-b8-c2-c7-d4-83	dynamic
	10.0.0.2	00-b8-c2-c7-d4-05	dynamic
	10.0.0.138	00-b8-c2-ed-aa-cb	dynamic

**Default Gateway : fe80::2b8:c2ff:feed:aacb%7
10.0.0.138**

ב. רישמו את כתובות המקור וכתובת היעד עבור שכבת ה- IP. למי שייכות כתובות אילו?



כתובות אלה הן כתובות IPv6:

כתובת ה- IP של המקור היא: **2a06:c701:711b:dc00:4d24:6d29:3f98:cab**
זו היא כתובת IPv6 (זמןית) של המחשב שלך.

```
Wireless LAN adapter    Wi-Fi:

Connection-specific DNS Suffix . : local
IPv6 Address. . . . . : 2a06:c701:711b:dc00:cf4:e11f:ce41:9772
Temporary IPv6 Address. . . . . : 2a06:c701:711b:dc00:4d24:6d29:3f98:cab
```

כתובת ה- IP של היעד היא: **2a00:1450:4028:803::2005**
כתובת זו שייכת לשרת של Google.
זה היה את זה על ידי הדומיין **1e100.net**.

```
C:\Users\adans>nslookup 2a00:1450:4028:803::2005
Server: UnKnown
Address: 2a06:c701:711b:dc00:2b8:c2ff:feed:aacb

Name: tlv04s06-in-x05.1e100.net
Address: 2a00:1450:4028:803::2005
```

ג. איזה אפליקציה בחרתם להשתמש? הראו כי היא אכן משתמשת ב-TCP. מה ה-port שלה?

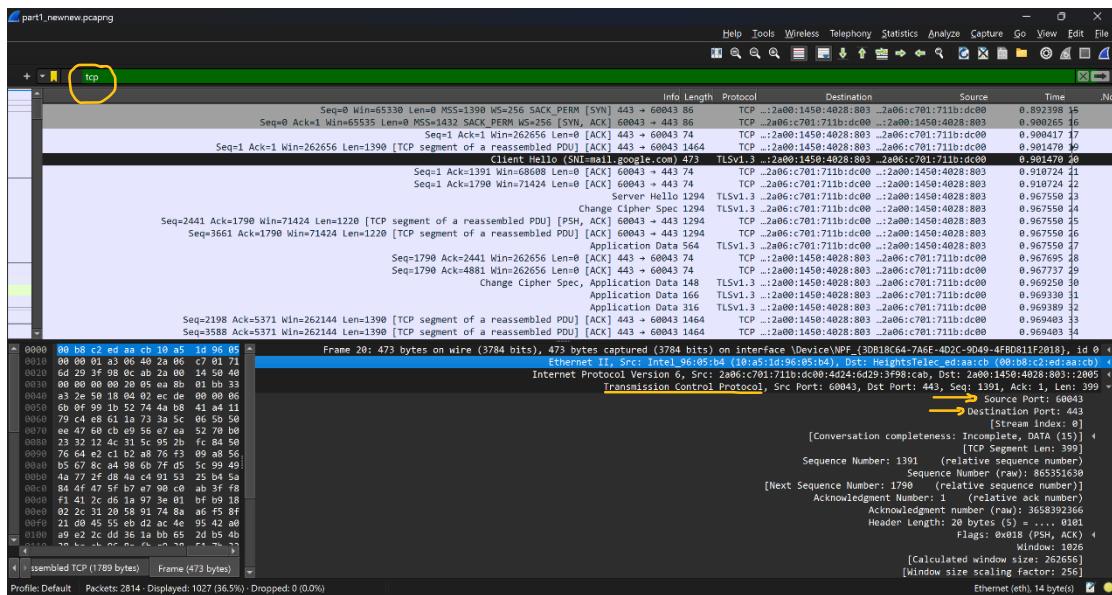
אני השתמשתי בדף אינטרנט **chrome**.

"**TCP**, כי ניתן לראות את השדה "TCP", מה שמשמעותוTCP מה שמשמעותוTCP".

פורט מקור – 60043 : זה הפורט דינמי שהוקצה באופן אקרים על ידי המערכת כדי ליצר את החיבור לשרת.

פורט יעד – 443 : מה שמעיד על כך שהחיבור הוא חיבור HTTPS, המשמש לתקשורת מאובטחת בין דפדפן אינטרנט לשרת.

לטס'יקום, הדפדפן מתקשר עם שרת אינטרנט בצורה מאובטחת באמצעות פרוטוקול HTTPS דרך הפורט 443, בעוד שהמערכת הקצתה את הפורט 60043 כדי לנוהל את החיבור הזה מצד שלו.



ד. הראו את תהליך הקמת הקשר בTCP? מה הנ吐נים המועברים בתהליך הקמת קשר זה? הסבירו אותו.

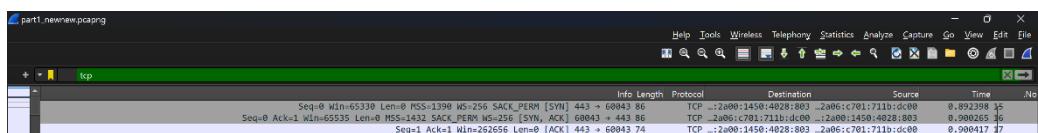
. Three-way-Handshake המשמש בTCP.

- שלב 1 : **SYN** - חיבור מס' 15, אני שולחת הודעת SYN לשרת כדי לפתוח את החיבור.
- מס' רצף 0. (אני מנסה להקים חיבור עם השרת דרך הפורט 443).

- שלב 2 : **SYN-ACK** - חיבור מס' 16, השרת מגיב ומאשר את ההודעה על ידי שליחת SYN-ACK בחזרה אליו.
- מס' רצף 0 ומספר אישור 1. (השרת מאשר את בקשת החיבור ומסנכרן את המספרים)

- שלב 3 : **ACK** - חיבור מס' 17, אני שולחת הודעת ACK לשרת כדי לאשר את קבלת הודעת SYN-ACK.
- מספר רצף 1 ומספר אישור 1.

(החיבור הוקם בהצלחה, וניתן להתחילה בתקשורת נתונים).



חלק ב'

ב חלק זה של התרגיל נספק קבצי PCAP עבור wireshark בהם העברנו תעבורת על פס צר, ונראה כיצד מගיב פרוטוקול-TCP במצב שבו יש צורך להעביר מידע מעבר לקיבולת של הרשת.

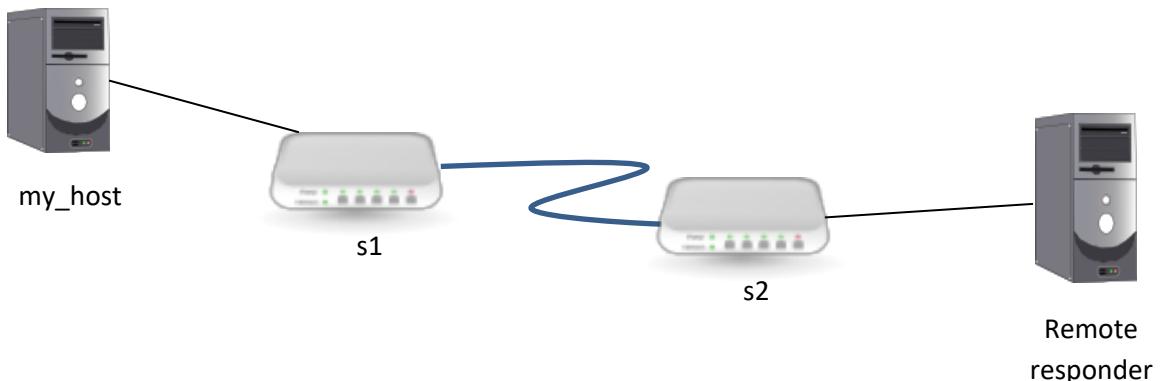
הצירור הבא ממחיש את הטופולוגיה שאיתה עבדנו.

הטופולוגיה מורכבת מ-2 מחשבים ו-2 מתגים.

בין 2 המתגים עובר קו ב מהירות של 500Kbps. זהו רוחב פס מאד צר יחסית לשאר הרשת. כר "יצרנו צוואר בקבוק" שיגרום לאובדן חבילות במידה וניצר תעבורת גדולה מהקיבולת של קו זה.

התחנה ממנה הרצנו את המדידות היא `host_my`. התחנה עמה תקשrnנו היא ה- `remote responder`

העברת המידע נעשית באמצעותiperf המותקן בשני המחשבים באמצעותו ניתן לשלוט על פרמטרי TCP וקצב השידור.



תעבורה בקצב נמוך

תחליה ננתה תעבורה בקצב נמוך (12Kbps) בין `my_host` ל-`remote responder`, שמייצר מעת חבילות אותן ננתה.

תוצאות התעבורה המוצרתה ע"י iperf מוקלטות בקצב בשם `tcp_12k.pcapng` אותו יש לנתח.

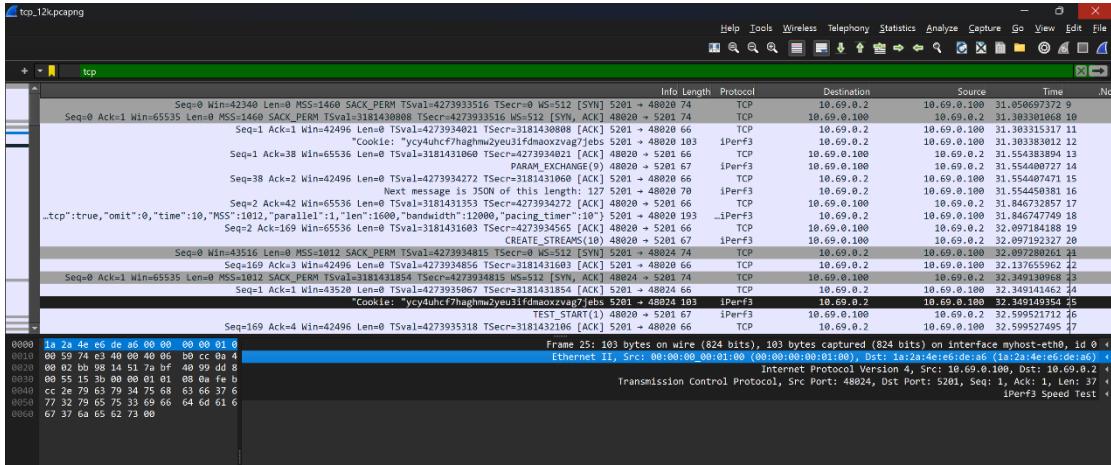
שימוש לב: כאשר מרים iperf, התכנית פותחת 2 קשרי TCP, כאשר בשניהם פורט היעד הוא 5201. הקשר TCP הראשון הוא ערוץ בקרה, ועל הקשר השני (בדרך כלל ה-`port` של Source שלו הוא מספר גבוה יותר) רצה תעבורת הבדיקה. רק הקשר השני מעוניין אותנו.

אחרי שאיתרתם את תעבורת הבדיקה, פלטרו את wireshark כך שתראו רק את הקשר הספציפי זהה, ע"י בחירה של חבילה אחת מהקשר, כפטור עכבר ימני ולחיצה על "Follow TCP Stream". ניתן לסגור מיידית את החלון הנוסף שנפתח, ולהתמקד חזרה בחולון הראשי.

שימוש לב שנשלחו בתחילת כמה חבילות להקמת הקשר. בחבילות אלו לא נשלחת תעבורה, ושליחת התעבורה מתבצעת רק כאשר הקמת הקשר מסתיימת.

א. מה כתובת-ip של המחשב `my_host` ושל ה-`remote responder`?

כתובת ה-IP של המחשב **my_host**
כתובת ה-IP של ה **remote responder**



לאחר ניתוח התעבורה, זיהוינו Ci התקשרות בין המחשבים הקיימים באמצעות כל הבדיקה iP. בתמונה ניתן לראות את החבילות שנלכדו, כאשר כתובות ה IP של _host remote responder זההה כ-10.69.0.100, וכתובות ה IP של my _host זההה כ-10.69.0.2. זיהוי זה ה证实 על החבילות שנשלחו והתקבלו ביניהם על גבי פורט 5201, אשר שימש את iPPerf3 במהלך הבדיקה.

ב. מהו הכוון התעבורה הנשלחת ? מייזה מחשב לאיזה מחשב?

ניתן לראות שהתüberורה נשלחה מהמחשב `my_host` שכתובת ה IP שלו היא 10.69.0.100, אל המחשב `remote responder` שכתובת ה IP שלו היא 10.69.0.2 זהה כיון התüberורה, הכוללת את הנתונים שנשלחו מכתובת 10.69.0.100 אל כתובת 10.69.0.2.

כשר מוקם קשור TCP, שני הצדדים מעבירים בינהם פרמטרים המאפשרים את הקשר. הפרמטרים הללו נמצאים בחלק ה-Options של ה-TCP Header. פיתחו את ה-TCP Header.

ג. מה ה-MSS של המחשב השולח? מהו גודל ה-payload עבור ערך זה?

כפי שניתן לראות, ה-MSS של המחשב השולח הוא 1012 bytes.

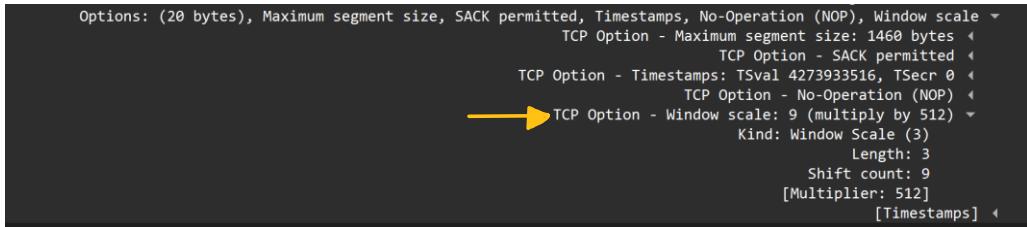
```
Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale ▾  
          TCP Option - Maximum segment size: 1012 bytes ▾  
          Kind: Maximum Segment Size (2)  
          Length: 4  
          → MSS Value: 1012  
          TCP Option - SACK permitted ▾  
TCP Option - Timestamps: TSval 3181431854, TSecr 4273934815 ▾  
          TCP Option - No-Operation (NOP) ▾  
          TCP Option - Window scale: 9 (multiply by 512) ▾  
          [Timestamps] ▾  
          [SEQ/ACK analysis] ▾
```

Header Length: 40 bytes (10) = 1010

$$\text{payload} = \text{MSS} - \text{TCP_Header} = 1012 - 40 = 972$$

גודל ה payload המרבי שיכל להישלח במערכת זו הוא 972 bytes.

ד. בהקמת הקשר מהו ה scaling factor של חלון הקבלה? מהו אומר? ומדוע יש צורך בערך זה?



ערך ה scaling factor של חלון הקבלה הוא 9. ה scaling factor הזה אומר שבכל פעם שמתאפשר ערך בשדה חלון הקבלה במסגרת חיבור ה TCP, יש להכפיל אותו ב- 512 (2 בחזקת 9) כדי לחשב את גודל חלון הקבלה האמתי. זה אומר שבפועל, חלון הקבלה יכול להיות הרבה יותר גדול מאשר שופיע בשדה עצמו, וזה חיוני כדי לתמוך בקצבים גבוהים של תעבורת רשת. יש צורך ב Scaling Factor זה כדי לאפשר גודל חלון קבלה גדול יותר, במיוחד במקרים של רוחב הפס. מהירויות שבהם חלון קבלה רגיל של 65,535 ביטים אינן מספיק לניצול מלא של רוחב הפס. ללא ה Scaling Factor יתכן שהחיבור לא יוכל לנצל את מלאו הפוטנציאלי שלו ברשומות מודרניות עם מהירותים גבוהות.

עיברו לתחנית capture.

ה. כמה חבילות נשלחות לצורך סיום הקשר? אילו Flags דלקים בכל אחת מהן?

ניתן לזהות 4 חבילות נשלחות לצורך סיום הקשר.

Seq=1 Ack=16938 Win=105984 Len=0 Tsv1=3181442356 TSecr=4273944918 [FIN, ACK] 48024 → 5281 66	TCP	10.69.0.100	10.69.0.2	42.849767114	73
Seq=16938 Ack=2 Win=43520 Len=0 Tsv1=4273945572 TSecr=3181442356 [ACK] 5281 → 48024 66	TCP	10.69.0.2	10.69.0.100	42.853638905	77
Seq=16938 Ack=2 Win=43520 Len=0 Tsv1=4273946402 TSecr=3181442356 [FIN, ACK] 5281 → 48024 66	TCP	10.69.0.2	10.69.0.100	43.684259388	85
Seq=2 Ack=16939 Win=105984 Len=0 Tsv1=3181443440 TSecr=4273946402 [ACK] 48024 → 5281 66	TCP	10.69.0.100	10.69.0.2	43.934585491	88

הדגלים הדלקים בכל חבילה:

חביבה 73: FIN, ACK

חביבה 77: ACK

חביבה 85: FIN, ACK

חביבה 88: ACK

בהתבסס על הניתוח של החבילות שנמצאו, ניתן לראות כי תחילת סיום הקשר בוצע בצורה תקינה עם ארבע חבילות, כאשר דגלי FIN ו ACK דלקים בהתאם לציפוי בכל אחת מהחבילות המעורבות.

ו. איך אפשר לדעת כמה בתים עברו בסך הכל מצד אחד לצד השני? ציינו את כמות הבתים ב-

2 הכוונים, לא כולל Headers.

כינון ראשון: 15301 bytes
 $(final_seq = 16038) - (first_seq = 0) - 1 - (23 * 32) = 15301$
 החסרטוי אחד כי בתחילת תחילת הקשר אנו מוסיפים אחד.
 יש 23 הודעות אחרי הקמת הקשר זהה TCP_HEADER.

Seq	Ack	Win	Len	Tsval	TSecr	Info	Length	Protocol	Destination	Source	Time	No
Seq=1638	Ack=1	Win=43528	Len=1000	Tsval=4273936385	TSecr=3181432686	[ACK]	5201 → 48824 1866	TCP	10.69.0.2	10.69.0.100	33.666597715	36
Seq=2638	Ack=1	Win=43528	Len=60	Tsval=4273936385	TSecr=3181432696	[PSH, ACK]	5201 → 48824 666	TCP	10.69.0.2	10.69.0.100	33.666606268	37
Seq=3238	Ack=1	Win=43528	Len=1000	Tsval=4273937452	TSecr=3181433423	[ACK]	5201 → 48824 1066	TCP	10.69.0.2	10.69.0.100	34.733596916	40
Seq=4238	Ack=1	Win=43528	Len=60	Tsval=4273937452	TSecr=3181433423	[PSH, ACK]	5201 → 48824 666	TCP	10.69.0.2	10.69.0.100	34.733601379	41
Seq=4838	Ack=1	Win=43528	Len=1000	Tsval=4273938518	TSecr=3181434498	[ACK]	5201 → 48824 1866	TCP	10.69.0.2	10.69.0.100	35.799595778	44
Seq=5438	Ack=1	Win=43528	Len=60	Tsval=4273938518	TSecr=3181434498	[ACK]	5201 → 48824 666	TCP	10.69.0.2	10.69.0.100	35.799595778	45
Seq=6438	Ack=1	Win=43528	Len=1000	Tsval=4273938585	TSecr=3181435550	[ACK]	5201 → 48824 1066	TCP	10.69.0.2	10.69.0.100	36.866598554	46
Seq=7438	Ack=1	Win=43528	Len=60	Tsval=4273938585	TSecr=3181435550	[PSH, ACK]	5201 → 48824 666	TCP	10.69.0.2	10.69.0.100	36.866600990	49
Seq=8038	Ack=1	Win=43528	Len=1000	Tsval=4273940652	TSecr=3181436623	[ACK]	5201 → 48824 1866	TCP	10.69.0.2	10.69.0.100	37.933597567	52
Seq=9938	Ack=1	Win=43528	Len=60	Tsval=4273940652	TSecr=3181436623	[PSH, ACK]	5201 → 48824 666	TCP	10.69.0.2	10.69.0.100	37.933599977	53
Seq=9638	Ack=1	Win=43528	Len=1000	Tsval=4273941718	TSecr=3181437698	[ACK]	5201 → 48824 1066	TCP	10.69.0.2	10.69.0.100	38.999596737	56
Seq=10638	Ack=1	Win=43528	Len=60	Tsval=4273941718	TSecr=3181437698	[PSH, ACK]	5201 → 48824 666	TCP	10.69.0.2	10.69.0.100	38.999598816	57
Seq=11238	Ack=1	Win=43528	Len=60	Tsval=4273942785	TSecr=3181438756	[PSH, ACK]	5201 → 48824 1066	TCP	10.69.0.2	10.69.0.100	38.999598816	58
Seq=12238	Ack=1	Win=43528	Len=60	Tsval=4273942785	TSecr=3181438756	[PSH, ACK]	5201 → 48824 666	TCP	10.69.0.2	10.69.0.100	38.999598816	59
Seq=13238	Ack=1	Win=43528	Len=1000	Tsval=4273943499	TSecr=3181439082	[ACK]	5201 → 48824 1066	TCP	10.69.0.2	10.69.0.100	41.133608996	65
Seq=14438	Ack=1	Win=43528	Len=1000	Tsval=4273944918	TSecr=3181440890	[ACK]	5201 → 48824 1066	TCP	10.69.0.2	10.69.0.100	42.199593865	68
Seq=15438	Ack=2	Win=43528	Len=60	Tsval=4273945572	TSecr=3181442356	[ACK]	5201 → 48824 66	TCP	10.69.0.2	10.69.0.100	42.199595597	69
Seq=16038	Ack=2	Win=43528	Len=60	Tsval=4273946492	TSecr=3181442356	[FIN, ACK]	5201 → 48824 66	TCP	10.69.0.2	10.69.0.100	42.199595597	70
Seq=16938	Ack=2	Win=43528	Len=60	Tsval=4273946492	TSecr=3181442356	[FIN, ACK]	5201 → 48824 66	TCP	10.69.0.2	10.69.0.100	43.684259388	71

כינון שני: 0 bytes

$$1 - 0 - 1 = 0$$

Seq	Ack	Win	Len	Tsval	TSecr	Info	Length	Protocol	Destination	Source	Time	No
Seq=1	Ack=1638	Win=69632	Len=0	Tsval=3181432696	TSecr=4273935668	[ACK]	5201 → 48824 5201	TCP	10.69.0.100	10.69.0.2	33.899736665	15
Seq=1	Ack=1638	Win=71680	Len=0	Tsval=3181434323	TSecr=4273936385	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	33.216620794	18
Seq=1	Ack=1638	Win=73728	Len=0	Tsval=3181434323	TSecr=4273936385	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	33.216634496	19
Seq=1	Ack=4238	Win=75776	Len=0	Tsval=3181434499	TSecr=4273937452	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	34.983634499	42
Seq=1	Ack=4838	Win=77824	Len=0	Tsval=3181434499	TSecr=4273937452	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	34.983638590	43
Seq=1	Ack=5838	Win=79872	Len=0	Tsval=3181435550	TSecr=4273938518	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	36.049635829	46
Seq=1	Ack=6438	Win=81920	Len=0	Tsval=3181435550	TSecr=4273938518	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	36.049646669	47
Seq=1	Ack=7438	Win=83968	Len=0	Tsval=3181435550	TSecr=4273938518	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	37.1165640726	50
Seq=1	Ack=8038	Win=85065	Len=0	Tsval=3181435623	TSecr=4273938585	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	38.1165640726	51
Seq=1	Ack=8638	Win=86960	Len=0	Tsval=3181437699	TSecr=4273940652	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	38.183625485	55
Seq=1	Ack=9238	Win=88960	Len=0	Tsval=3181437699	TSecr=4273940652	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	38.183625485	56
Seq=1	Ack=9838	Win=91648	Len=0	Tsval=3181438756	TSecr=4273941718	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	39.249617771	58
Seq=1	Ack=11238	Win=93696	Len=0	Tsval=3181438756	TSecr=4273941718	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	39.249622105	59
Seq=1	Ack=12238	Win=95744	Len=0	Tsval=3181439823	TSecr=4273942785	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	40.216614670	62
Seq=1	Ack=12838	Win=97792	Len=0	Tsval=3181439823	TSecr=4273942785	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	40.216618642	63
Seq=1	Ack=13438	Win=101880	Len=0	Tsval=3181440890	TSecr=4273944852	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	41.383638086	66
Seq=1	Ack=14038	Win=103936	Len=0	Tsval=3181441196	TSecr=4273944918	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	42.400680591	70
Seq=1	Ack=16038	Win=105984	Len=0	Tsval=3181441956	TSecr=4273944918	[ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	42.409611146	71
Seq=1	Ack=16938	Win=105984	Len=0	Tsval=3181442356	TSecr=4273944918	[FIN, ACK]	48824 → 5201 66	TCP	10.69.0.100	10.69.0.2	42.849767114	73

ב-jeffrey הגדרנו שליחת ייחודת מידע של 1600 בתים.

ז. המחשב השולח שולח כל פעם 2 חבילות אחת אחרי השניה. למה?

במצב שבו jeffrey מוגדר לשולח ייחודת מידע בגודל של 1600 בתים, והמגבלה בפועל של MSS היא קטנה יותר. במקרה זה, ה-TCP מבצע חלוקה אוטומטית של המידע לחבילות בגודל שמתאים למוגבל המותר. המטרה היא לוודא שכל הנתונים נשלחים בצורה נכונה, 1012 בתים והשנייה 580 בתים). המטרה היא לוודא שגם כל הנתונים נשלחים בצורה נכונה, בהתאם למוגבלות MSS שנקבעו בחיבור. באופן זה, ה-TCP יכול להעביר את המידע בצורה יעילה, תוך כדי התאמת לרוחב הפס ולאילוצי הרשת.

ח. בחלק מהחבילות מופיע flag push. למה דואק בחבילות האלו?

	Info	Length	Protocol	Destination	Source	Time	No
Seq=169 Ack=5 Win=42496 Len=0 Tsvl=427393568 Tscr=3181432322 [ACK]	5201 → 48024 66	TCP	10.69.0.100	10.69.0.2	32.849599201	30	
Seq=38 Ack=1 Win=43520 Len=1000 Tsvl=427393568 Tscr=3181432322 [ACK]	5201 → 48024 66	TCP	10.69.0.100	10.69.0.2	32.849599201	31	
Seq=1839 Ack=1 Win=43520 Len=690 Tsvl=427393568 Tscr=3181432322 [ACK]	5201 → 48024 66	TCP	10.69.0.100	10.69.0.2	32.849599201	32	
Seq=1 Ack=1038 Win=67584 Len=8 Tsvl=31814323606 Tscr=427393568 [ACK]	48024 → 5201 66	TCP	10.69.0.100	10.69.0.2	33.009731682	34	
Seq=1 Ack=1638 Win=69632 Len=8 Tsvl=31814323606 Tscr=427393568 [ACK]	48024 → 5201 66	TCP	10.69.0.100	10.69.0.2	33.009731665	35	
Seq=1638 Ack=1 Win=43520 Len=1000 Tsvl=4273936385 Tscr=3181432606 [ACK]	5201 → 48024 1066	TCP	10.69.0.2	10.69.0.100	33.666597715	36	
Seq=2638 Ack=1 Win=43520 Len=600 Tsvl=4273936385 Tscr=3181432606 [PSH, ACK]	5201 → 48024 666	TCP	10.69.0.2	10.69.0.100	33.666598028	37	
Seq=1 Ack=2638 Win=71680 Len=0 Tsvl=3181433423 Tscr=4273936385 [ACK]	48024 → 5201 66	TCP	10.69.0.100	10.69.0.2	33.916629794	38	
Seq=1 Ack=3238 Win=73728 Len=0 Tsvl=3181433423 Tscr=4273936385 [ACK]	48024 → 5201 66	TCP	10.69.0.100	10.69.0.2	33.916634906	39	
Seq=4238 Ack=1 Win=43520 Len=690 Tsvl=4273937452 Tscr=3181433423 [PSH, ACK]	5201 → 48024 666	TCP	10.69.0.2	10.69.0.100	34.733681379	40	
Seq=1 Ack=4238 Win=690 Tsvl=4273937452 Tscr=3181433423 [PSH, ACK]	48024 → 5201 66	TCP	10.69.0.2	10.69.0.100	34.733681379	41	
Seq=1 Ack=4238 Win=75776 Len=0 Tsvl=3181434490 Tscr=4273937452 [ACK]	48024 → 5201 66	TCP	10.69.0.100	10.69.0.2	34.983634499	42	
Seq=1 Ack=4838 Win=77824 Len=0 Tsvl=3181434490 Tscr=4273937452 [ACK]	48024 → 5201 66	TCP	10.69.0.100	10.69.0.2	34.983638598	43	
Seq=4838 Ack=1 Win=43520 Len=1000 Tsvl=4273938518 Tscr=3181434490 [ACK]	5201 → 48024 1066	TCP	10.69.0.2	10.69.0.100	35.799595778	44	
Seq=5838 Ack=1 Win=43520 Len=600 Tsvl=4273938518 Tscr=3181434490 [PSH, ACK]	5201 → 48024 666	TCP	10.69.0.2	10.69.0.100	35.799598464	45	
Seq=1 Ack=5838 Win=79872 Len=0 Tsvl=3181435556 [ACK]	48024 → 5201 66	TCP	10.69.0.100	10.69.0.2	36.049635829	46	
Seq=1 Ack=6438 Win=81920 Len=0 Tsvl=3181435556 Tscr=4273938518 [ACK]	48024 → 5201 66	TCP	10.69.0.100	10.69.0.2	36.049646609	47	
Seq=6438 Ack=1 Win=43520 Len=1000 Tsvl=4273939585 Tscr=3181435556 [ACK]	5201 → 48024 1066	TCP	10.69.0.2	10.69.0.100	36.866598654	48	
Seq=7438 Ack=1 Win=43520 Len=0 Tsvl=4273939585 Tscr=3181435556 [PSH, ACK]	5201 → 48024 666	TCP	10.69.0.2	10.69.0.100	36.866600950	49	

הדגל PSH מופיע בחלק מהחבילות כי הוא מבטיח שהנתונים יעברו ליישום המקלט באופן מיידי, מבלי להמתין לנ נתונים נוספים. במצב שבו `zero` שלוח יחידת מידע בגודל 1600 בתים, והמגבלה של MSS עומדת על 1012 בתים (כפי שמצאו בסעיף ג) הנתונים מתפצלים לשתי חבילות. החבילה השנייה, המכילה את יתרת 588 הbytes, נשלחת עם דגל PSH כדי להורות למקבל לעבד את הנתונים ברגע שהם מתקיים, ולא לאגור אותם בזיכרון ביןיהם. דגל זה קרייטי כאשר יש צורך בעיבוד מיידי של המידע, כמו במקרים בהם כל קטע מידע חשוב ויש להעבירו ליישום ללא עיכוב.

פרוטוקול TCP על פס צר

התרגיל הזה וגם התרגילים הבאים מדגים את הדרך בה מתבצעת בקרת תעבורת-TCP וኒיצר תעבורת מעלה הפס הצר.

על מנת להבין את התרגיל הבא, הנה רקע קצר על המנגנוןים ב망ת Ethernet שלוקחים חלק בשידור וROLONETים ליטיסו'.

לכל מtag או נתב יש תור יציאה, שבו חבילות ממתיינות לתורן כדי להיות משודרות. לצורך העניין נניח שהזו תור FIFO וקצב המילוי שלו הוא גדול מקצב השידור על הקוו, על מנת לא לאלבד חבילות כאשר יש burst (כלומר שידור קצר בקצב מאד מהיר) או כאשר יש שידור בו זמינות מכמה מקורות יחד.

בניסוי שלנו אנחנו נפשط מад את העניינים כדי שתוכלו לעקוב. לשם כך הגדרנו את מtag 1s עם צד אחד עם רוחב סרט גבוה הצד השני עם פס צר. כאשר נשדר מכוון הפס הרחב, תור היציאה של הפס הצר יתחל להתמלא.



בכל מtag ישנה האפשרות לkneg את גודל התור לערך אופטימלי. תור שהוא קצר מדי יכול לגרום לאבדן חבילות מיותר בעומס רגעים. תור שהוא ארוך מדי יגרום לזמן שידור ארוכים שגם הם בעיתים. הפתרון גם תלוי בסוג הרשת. ברשותות ארגוניות למשל אפשר לחוות עם תור יחסית ארוך. ברשותות של data center נkneg תורים הרבה יותר קצרים, ונעדי' לזרוק חבילות מאשר לתת להם לחכות. אנחנו קבענו תור מסוג FIFO הנicho כי התור בגודל **25 חבילות**.

שאלה נוספת היא מתי נחליט לזרוק חבילות. הגישה הטריוויאלית היא של Tail Drop. ככלمر לתור להתמלא עד הסוף, ואם אין מקום אז לזרוק את החבילה. הבעיה בגישה זו היא שאם נניח התור מתמלא בקצב כפול מקצב השידור, מעתה והלאה כל חבילה שנייה תזירק, מה שיגרום לבזבוז משאבי.

גישה יותר טובה מתחכמת נקראת RED. בגישה זו, קובעים סף, נניח חצי מאורך התור המקורי, וכאשר חווים סף זה חבילה יכולה להזירק בהסתברות נמוכה, נניח 0.05. באופן זה תזירק רק חבילה אחת, והחbillות שאחריה ייקלטו בתור בהסתברות של 0.95. כאשר הצד השולח יבין שנדרקה חבילה, זה יرمז לו להאט את הקצב, וכך העומס על התור ירגע. כך ניתן לפחות את בעיית עומס התור במחיר של אבדן חבילות לא גדול. אנחנו בניסוי משתמשים בשיטה של Tail Drop, מכיוון שהיא יותר קלה להבנה ולמעקב.

ט. בהינתן הקו של 500Kbps, עם RTT כולל של 250 מייל, כמה bytes in flight ייתן הקו זהה, ללא התורים? כמה חבילות עם MTU זה יתרוגם?

נסדר את הנתונים:

- קצב הקו(kbps) : bandwidth = 500,000 bits per second
- RTT כולל: 250 מייל-שניות = 0.25 שניות.
- 1514 bytes : MTU

$$\text{bytes in flight} = \text{bandwidth} \cdot \text{RTT}$$

נzie בנוסחה ונקבל:

$$\text{bytes in flight} = \frac{500,000 \cdot 0.25}{8 \text{ bits per byte}} = 15,625 \text{ bytes}$$

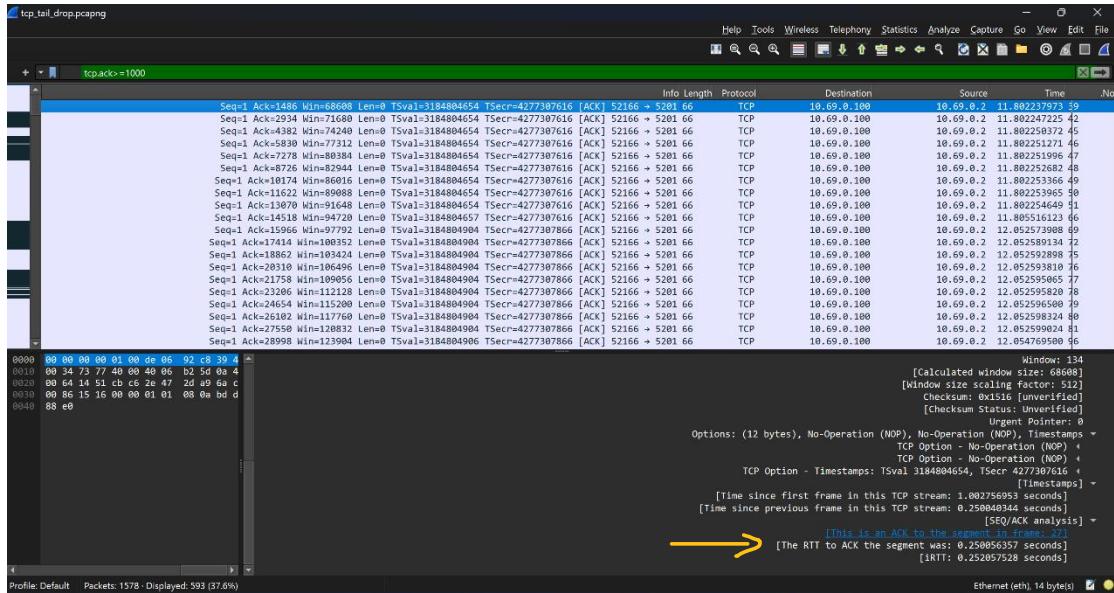
כעת, כדי לחשב כמה חבילות בגודל MTU ניתן לשדר במהלך זמן ה RTT, משתמש בנוסחה:

$$\text{number of packets} = \frac{\text{bytes in flight}}{\text{MTU}} = \frac{15,625 \text{ bytes}}{1514 \text{ bytes}} = 10.32$$

כעת ננתח תעבות TCP בין my_host ל-remote responder, שמייצרת ע"י iperf למשך 20 שניות. תוצאות התעבורה מוקלטות בקובץ בשם tcp_tail_drop.pcapng אותו יש לנתח.

. צינו את ה-RTT(Round Trip Time) של הבטים האלף, 20 אלף, 40 אלף, 60 אלף. לצורך כך יש להסתכל על חבילת ACK שחזרה מה-responder עם הערך הקרוב ביותר למספר הרצוי, ושגדל ממנו. בחבילה זו, פיתחו תחת TCP את SEQ/ACK Analysis, ושם יש את החישוב. הסבירו מדוע ה-RTT מתארך ככל שהזמן מתקדם. היכן בדיקת חבילות מתעכבות?

חבילת ACK עוברת 1000 בתים, היא חבילה הראשונה בתמונה. וכפי שנitin לראות ה- RTT הוא : 0.250056357 seconds



ה- RRT עברו 20,000 בתים הוא : 0.250343781 seconds

ה- RRT עברו 40,000 בתים הוא : 0.446419013 seconds

ה- RRT עברו 60,000 בתים הוא : 0.535979704 seconds

ה RTT מتأרך כל שזמן מתקדם בגל גודש בתורי הרשות. בתחילת העברת הנתונים, ה RTT היה נמוך יחסית והוא בסביבות 0.25 שניות עברו 1000 ו-20,000 בתים. אולם, לאחר 40,000 בתים, ה RTT עלה באופן משמעותי ל-0.446 שניות ולאחר מכן ל-0.535 שניות עברו 60,000 בתים.

הסיבה העיקרית להתרIOR של RTT היא גודש בתורי FIFO במתגים. כאשר התור מתמלא ונוצר עומס בתורים, החבילות נאלצות להמתין בתור לפני שהן משודרות, מה שmobiel להארכת זמן ההמתנה והגדלת ה RTT. מיקום העיכוב הוא בתורים של המתגים ובמקטעי רשת שביהם יש גודש. ככל שהעומס בתורים גובר, החבילות מתעכבות יותר זמן בתור, מה שגורם להארכת ה RTT. בנוסף, במצבים של גודש חמוץ יותר, יתכן שהתחילה להתרכש מצבים של Tail Drop, שבהם חבילות נזרקות עקב אחר מקומ בתור, מה mobiel לשידורים חוזרים ולהגברת העומס בתור, וכך מתרחש מעגל העיכובים והארכת ה RTT.

א. בשלב מסוים מתחילה ללכט לאיבוד חבילות. מה האינדיקציה שה Wireshark מקבל שמננו הוא מבין שחבילה הלכה לאיבוד?

ב Wireshark, האינדיקציה לכך שחבילה הלכה לאיבוד היא דיה של שידור חוזר. בנוסף, Wireshark עשוי לזהות אישורים כפולים (Duplicate ACKs), שבהם נשלחים ים זרים עברו אותו מסטר סידורי. במקרים כאלה, השולח עשוי לבצע שידור חוזר מהיר (Fast Retransmit) של החבילה האבודה. הידועות כמו [TCP segment lost] מהוות גם הן סימן לכך שחבילה לא נקלטה או הלכה לאיבוד.

- ניתן להשתמש במסננים ב Wireshark כדי לזהות בהירות אירועים כמו שידור חוזר ואיבוד חבילות, וביצוע ניתוח ממוקד יותר של התüberה ברשות.

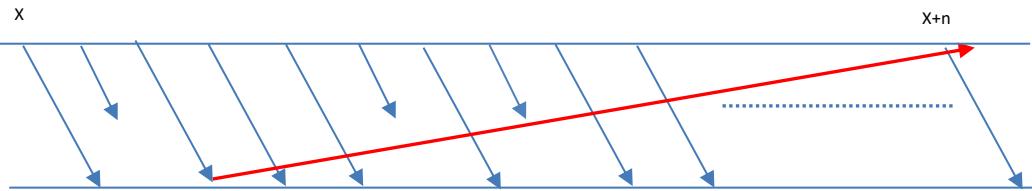
תחילה נכנסה לנסה מה קורה בראשת רגע לפני שחבילות מתחילה למכת לאיבוד.

יב. מהו הבית הראשון ב-stream שלא הגיע לעדנו בניסיון ראשון?

88366. הבית הראשון ב-stream שלא הגיע לעדנו בניסיון הראשון הוא הבית שמספרו 52166. מספר זה מייצג את ההפוכה שבhäbilah הראשונה לא התקבלה כראוי, ולכן נשלחה מחדש. מספר הסדרה של החבילות מייצג את מיקום הבית הראשון שלה בזרם הנתונים הכלול. כאשר מתרחש שידור חוזר, מספר הסדרה נשאר זהה, ומראה את המיקום המדוייק שבו החבילות המקוריות לא הגיעו לעדנה בניסיון הראשון.

tcp.analysis.retransmission						
	Info	Length	Protocol	Destination	Source	Time
52166 → 5201 [ACK] Seq=88366 Ack=1 Win=42496 Len=1448 Tsv=4277309186 Tscr=3184805973 [TCP Fast Retransmission] 1514	TCP	10..69..0..2	10..69..0..100	13..12188963 208		
52166 → 5201 [ACK] Seq=91262 Ack=1 Win=42496 Len=1448 Tsv=4277309211 Tscr=3184805998 [TCP Retransmission] 1514	TCP	10..69..0..2	10..69..0..100	13..12188963 209		
52166 → 5201 [ACK] Seq=94158 Ack=1 Win=42496 Len=1448 Tsv=4277309235 Tscr=3184806022 [TCP Retransmission] 1514	TCP	10..69..0..2	10..69..0..100	13..171613225 212		
52166 → 5201 [ACK] Seq=97054 Ack=1 Win=42496 Len=1448 Tsv=4277309284 Tscr=3184806070 [TCP Retransmission] 1514	TCP	10..69..0..2	10..69..0..100	13..220146711 215		

מרגע שהשליחה של חבילת X, האחרונה שהגיעה לפנייה מפילותות, ועד הדע ack נשלחו (חץ האדום), נשלחו חביבות. חלקן הגיעו וחלקן נפלן. אנחנו נכנסה לבירר יותר לעומק מה אירע.



יג. מהו הערך של ח? הנחיה: בדקו את החבילות האחרונות שנשלחה לפני ה-SEQ/ACK Analysis. מהערך זה תגזרו את מספר החביבות. הערה: תזכירו ששכבת ה-headers לא מתחשבת בגודל ה-datasize בslashes שמתוחתיו.

לפי הנחיות השאלה, מצאתי את החבילות האחרונות שנשלחה לפני שהתרחש ה-Fast Retransmit.

tcp.analysis.retransmission						
	Info	Length	Protocol	Destination	Source	Time
Seq=1 Ack=84022 Win=108655 Len=0 Tsv=3184805838 Tscr=4277309235 [ACK] 52166 → 5201 68	TCP	10..69..0..2	10..69..0..100	13..12188963 195		
Seq=157870 Ack=1 Win=42496 Len=1448 Tsv=4277309640 Tscr=3184805828 [ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..008575025 107		
Seq=1 Ack=85470 Win=108655 Len=0 Tsv=3184805852 Tscr=4277308240 [ACK] 52166 → 5201 66	TCP	10..69..0..100	10..69..0..2	13..008575025 107		
Seq=159318 Ack=1 Win=42496 Len=1448 Tsv=4277309651 Tscr=3184805852 [ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..008576892 108		
Seq=160766 Ack=1 Win=42496 Len=1448 Tsv=4277309652 Tscr=3184805856 [ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..008576892 109		
Seq=160766 Ack=1 Win=42496 Len=1448 Tsv=4277309653 Tscr=3184805856 [ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..021817961 200		
Seq=162124 Ack=1 Win=42496 Len=1448 Tsv=4277309151 Tscr=3184805922 [ACK] 5201 → 52166 1514	TCP	10..69..0..100	10..69..0..2	13..073128445 201		
Seq=162124 Ack=1 Win=42496 Len=1448 Tsv=4277309152 Tscr=3184805922 [ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..073135946 202		
Seq=162124 Ack=1 Win=42496 Len=1448 Tsv=4277309153 Tscr=3184805922 [ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..073135946 203		
Seq=165151 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805949 [ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..097482780 205		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..097484887 206		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..121876832 207		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..121876832 208		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..147372585 209		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..147388032 210		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..100	10..69..0..2	13..171609497 211		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..195571563 212		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..195571563 213		
Seq=165558 Ack=1 Win=42496 Len=1448 Tsv=4277309161 Tscr=3184805950 [PSH ACK] 5201 → 52166 1514	TCP	10..69..0..2	10..69..0..100	13..220176681 214		

[TCP Segment Len: 1448]

כדי לחשב את מספר החבילות, נחלק את הערך של bytes in flight בגודל החבילה.

$$n = \frac{\text{bytes in flight}}{\text{len}} = \frac{76744}{1448} = 53$$

החבילה הראשונה שהלכה לאיבוד נשלחה ב-fast retransmit. לאחר מכן התקבלו חבילות dup ack, עד שלאחר זמן מה התקבל ACK על ה-fast retransmit של ack.

יד. מה המספר הסידורי של ack על ?fast retransmit?

.**88366** המספר הסידורי של ack על fast retransmit הוא.

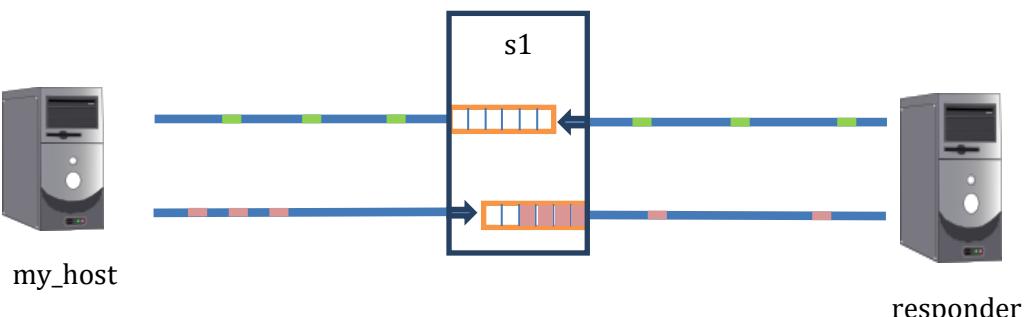
5201 → 52166 [ACK] Seq=1 Ack=88366 Win=198656 Len=0 Tsv=3184805973 Tscr=4277388264 SLE=9560 [TCP Dup ACK 201#2] 94 TCP 10.69.0.100 10.69.0.2 13.121876628 207
5201 → 52166 [ACK] Seq=1 Ack=88366 Win=198656 Len=0 Tsv=3184805973 Tscr=4277388264 SLE=9560 [TCP Fast Retransmission] 1514 TCP 10.69.0.100 10.69.0.2 13.121889663 208
5201 → 52166 [ACK] Seq=1 Ack=88366 Win=198656 Len=0 Tsv=3184805973 Tscr=4277388264 SLE=9560 [TCP Dup ACK 201#3] 94 TCP 10.69.0.100 10.69.0.2 13.121889663 209
5201 → 52166 [ACK] Seq=1 Ack=88366 Win=198656 Len=0 Tsv=3184805973 Tscr=4277388264 SLE=1015 [TCP Retransmission] 1514 TCP 10.69.0.100 10.69.0.2 13.121889663 210
5201 → 52166 [ACK] Seq=1 Ack=88366 Win=198656 Len=0 Tsv=3184805973 Tscr=4277388264 SLE=1015 [TCP Dup ACK 201#4] 94 TCP 10.69.0.100 10.69.0.2 13.171605697 211

טו. כמה ack dup התקבלו על החבילות הראשונות שהלכה לאיבוד? אין צורך לספר ידנית, dup ack ממוספרים.

ניתן לראות שהתקבלו **36 Dup ACKs** בעבר החבילות הראשונות שהלכה לאיבוד.

No	Time	Source	Destination	Protocol	Length	Info
1	10.69.0.2 13.79462938 455	10.69.0.100	10.69.0.2 13.728792915 357	TCP	0	
2	10.69.0.2 13.72884978 259	10.69.0.100	10.69.0.2 13.72884978 259	TCP	0	
3	10.69.0.2 13.72884978 261	10.69.0.100	10.69.0.2 13.72884978 261	TCP	0	
4	10.69.0.2 13.72884978 483	10.69.0.100	10.69.0.2 13.72884978 483	TCP	0	
5	10.69.0.2 13.82573576 455	10.69.0.100	10.69.0.2 13.82573576 455	TCP	0	
6	10.69.0.2 13.840972939 267	10.69.0.100	10.69.0.2 13.840972939 267	TCP	0	
7	10.69.0.2 13.873805707 269	10.69.0.100	10.69.0.2 13.873805707 269	TCP	0	
8	10.69.0.2 13.922373039 273	10.69.0.100	10.69.0.2 13.922373039 273	TCP	0	
9	10.69.0.2 13.946881252 275	10.69.0.100	10.69.0.2 13.946881252 275	TCP	0	
10	10.69.0.2 14.40572232 315	10.69.0.100	10.69.0.2 14.40572232 315	TCP	0	

ונסה להבין מאיין נובעת תשובה זו שקיבלתם. הבינו באior הבא. בשאלת ט' חישבתם כמה חבילות נמצאות בזמן מסוים על הקיום בשני היכונים.icut עלינו להתייחס לחבילות בכיוון השילוח בלבד (החבילות הכתומות), וכן להתחשב בחבילות שהמтиינו בתור שבנטב.



שיםו לב כי הגדרנו את גודל התור ל 25 חבילות

טז. מהי תפוסת התור בנתב ברגע איבוד החבילות הראשונות? ניתן להניח כי תפוסה זו נשמרת לאחר האיבוד, מכיוון שקצב השידור אינו יורד.

תפוזת התור בנתב ברגע איבוד החבילה הראשונה הייתה **25 חבילות**.
 כאשר התור מתמלא, כל חבילה נוספת שמנסה להיכנס לתור נדחית, מה שהגורם לאיבוד
 חבילות. במצב זה, התור נמצא בתפוצה מלאה, ואיבוד החבילות מתרחש כתוצאה מהעומס
 המלא בתור. ההנחה שהקצב לא יורד עזרה לי להבין שהטור נשאר בתפוצה מלאה גם
 לאחר איבוד החבילה הראשונה, מכיוון שהעומס בתור נמשך.

יב. סיכמו את מספר החבילות בטור יחד עם מספר החבילות הממוצע בכיוון השילחה. האם חישוב זה מספק תשובה מקרובה למספר $\text{ack} - \text{dup}$ שהתקבלו בפועל?

- מספר החבילות בתור: 25
מספר החבילות הממוצע בכיוון השלילה: 10.32

$$25 + 10.32 = 35.32$$

чисוב זה מספק תשובה מוקrbת למספר Dup ACKs שהתקבלו בפועל, אשר היה 36 (חישבנו בסעיף טו). ההתאמה בין המספרים מעידה על כך שהחישוב מספק הערכה טובה של המצב בפועל בראשת.

יח. לפי התשובות של יג ייחד עם שאלתנו כמה חבילות נפלו?

בבנימוף ט' מוצאנו שמספר החבילות שווה ל 36. בבנימוף י' מוצאנו שמספר DUP ACKS הוא 53.

על מנת לחשב את מספר החבילות שנפלו אחרי החבילה הראשונה, נחסיר מספר ה **bytes in flight**.
ACKs שהתקבלו ממספר החבילות הכולל ב.

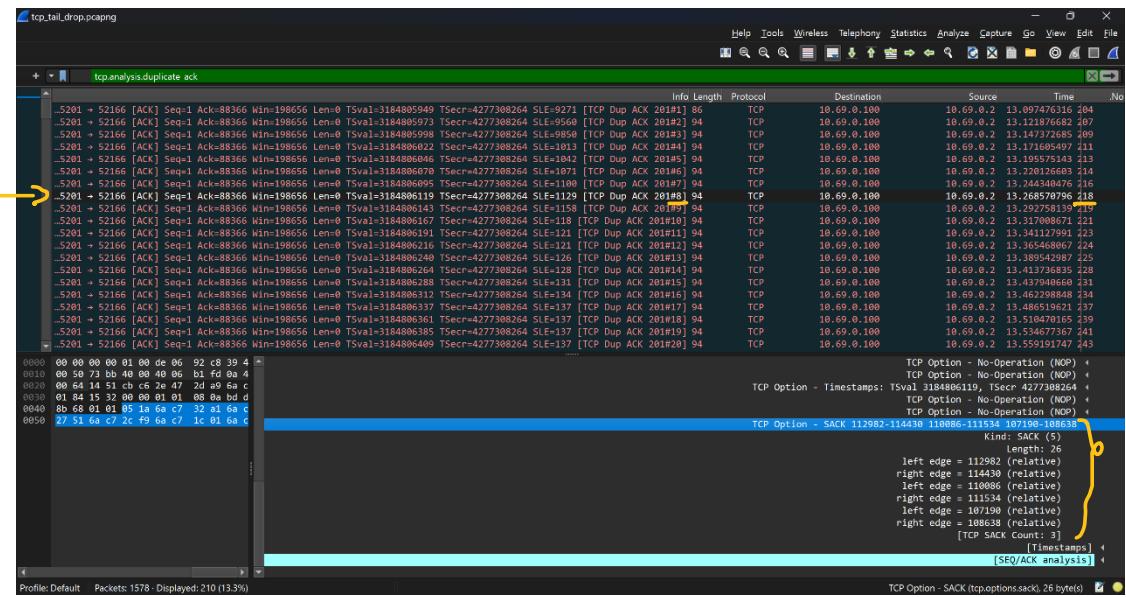
כלומר, 18 חבילות נפלו בפועל במהלך השידור. (כולל החבילה הראשונה).

.Selectiv Ack נבחן כיצד מוממש מנגנון

מיצאו את הודעת-h-Ack Dup השמינית, סמנו אותה ורשמו את מספרה. הרחיבו את שדה-options והעתיקו את שדה-h-sack (השתמשו-b>Description). על כמה רצים מציביע-h-sack? כמה בתים חסרים בין כל 2 רצים?

218 מס' הודעתה Dup Ack השמינית:

TCP Option - SACK 112982-114430 110086-111534 107190-108638



ה SACK מצביע על 3 רצפים שונים:

- רצף ראשון: 107190 עד 108638
- רצף שני: 110086 עד 111534
- רצף שלישי: 112982 עד 114430

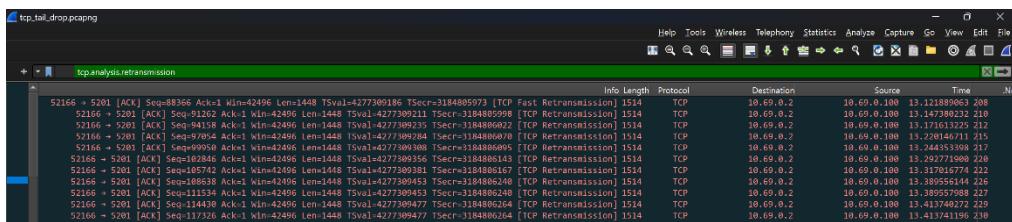
מספר הבתים החסרים בין הרצף הראשון והשני:
 $110086 - 108638 = 1448$

מספר הבתים החסרים בין הרצף השני והשלישי:
 $112982 - 111534 = 1448$

בין כל שני רצפים יש **1448** בתים חסרים.

ב. מה מספר הבעיות בהם נשלחו מחדש הרצפים החסרים מהשאלה הקודמת? שימו לב שהרבה פעמים הבעיות מוסמנות ע"י `out of order` Wireshark - `retransmit`. מעשית אין בעיות שנשלחות `out of order` במהלך הניסוי. כל מה שמוסמן כזה הוא `retransmit` שלא פורש כהלה.

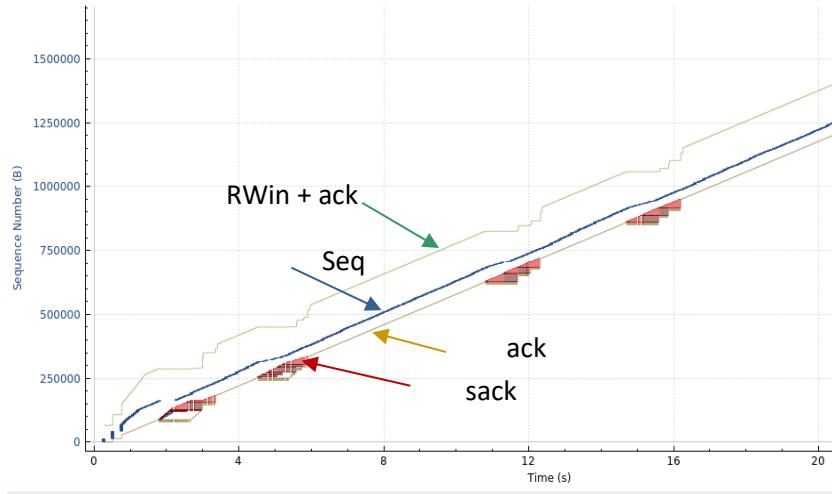
מספר הבעיות בהם נשלחו מחדש הרצפים החסרים מהשאלה הקודמת הוא 3.



הרץ' הראשון (108638 עד 107190): נשלח מחדש בעיה מס' 226
 הרץ' השני (110086 עד 111534): נשלח מחדש בעיה מס' 227
 הרץ' השלישי (112982 עד 114430): נשלח מחדש בעיה מס' 222

ניתוח גרפי של התעבורה

- סמןנו חבילה של תעבורה מהמחשב שלכם ל-responder.
- פיתחו את הגרף Statistics->TCP Stream Graph->Time Sequence Graph(tcptrace).
- זהו גרפ קצט מסווב, וננסה להבין ממה הוא מורכב.



בגרף זה כמה קווים: התחתון יירקן מייצג את ה-ack, הכהה המודגם מצין את ה-seq. הכתמים האדומים מסמנים את רצפי ה-sack שהתקבלו.

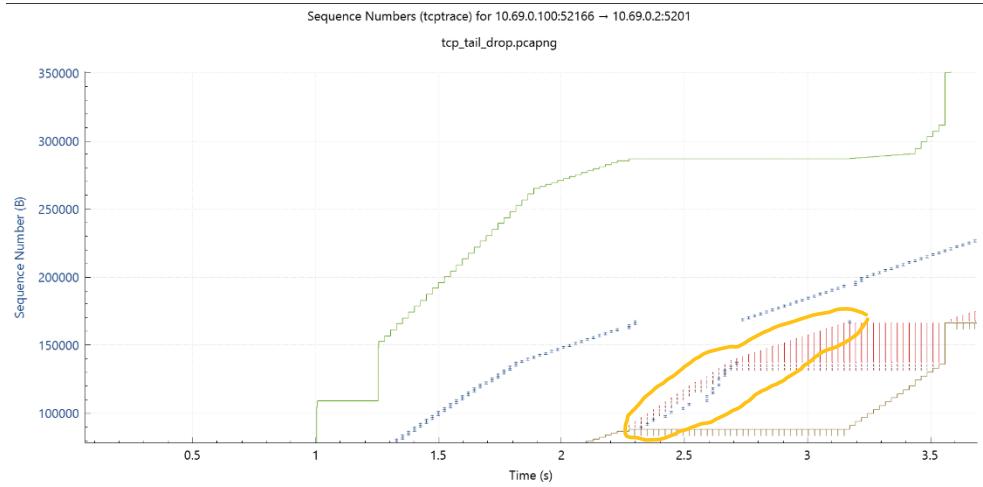
הקו הירוק העליון הוא החישוב של RWin+ack. ככלمر הוא הגבול העליון שה-seq (כלומר הכהול המודגם) יכול להגיע אליו.

השיפוע של הקו התחתון מייצג למעשה את הקצב האפקטיבי של קשר ה-TCP, מכיוון שהוא מראה באיזה קצב המידע הגיע לידי.

השיפוע של הקו המודגם מראה את הקצב באותו רגע של שילוח המידע.

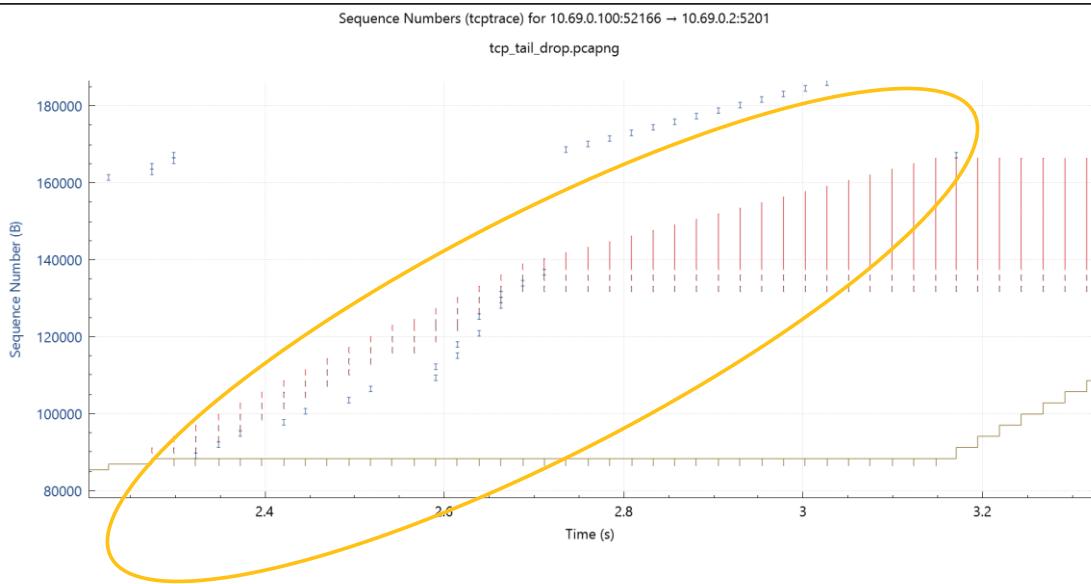
- בצעו זום באיזור הזמן הראשון בו ארעו איבודים, נניח בין שנייה 1.5 לשניה 3.5. שימו לב שמה שנראה קודם לקו כחול רצוף, הוחלף לנקודות. כל נקודה כזו מייצגת הودעת TCP עם SEQ שערכו מסווגן.

כא. העתיקו והדביקו את הגרף. היכן מופיעות חבילות retransmit? סמןו אותן.



ניתן לזהות את חבילות `retransmit` באזור המסומן בכתום. הנקודות הכהולות בתוך אזור זה מייצגות חבילות ששודרו מחדש (`retransmissions`). הנקודות הללו מופיעות רק כאשר יש אובדן חבילות, ולכן ההנחה שהן `retransmissions` היא הגיונית וਮתבוססת על כך ששידורים חוזרים מתרחשים כתוצאה מאובדן חבילות. הגרף מאפשר לנו לזהות את מספרי החבילות עבור כל נקודה על ידי הצבעה על הנקודות, מה שמאפשר לבדוק את חבילות `retransmit`. הנקודות הכהולות מופיעות בדיקת באזורי שביהם היי איבודי חבילות, וכך ניתן להסיק שהן מייצגות חבילות ששודרו מחדש כתוצאה מכך.

כב. סיפו את הנקודות. כמה `retransmits` נשלחו? האם זה תואם את תשובהיכם מיח?



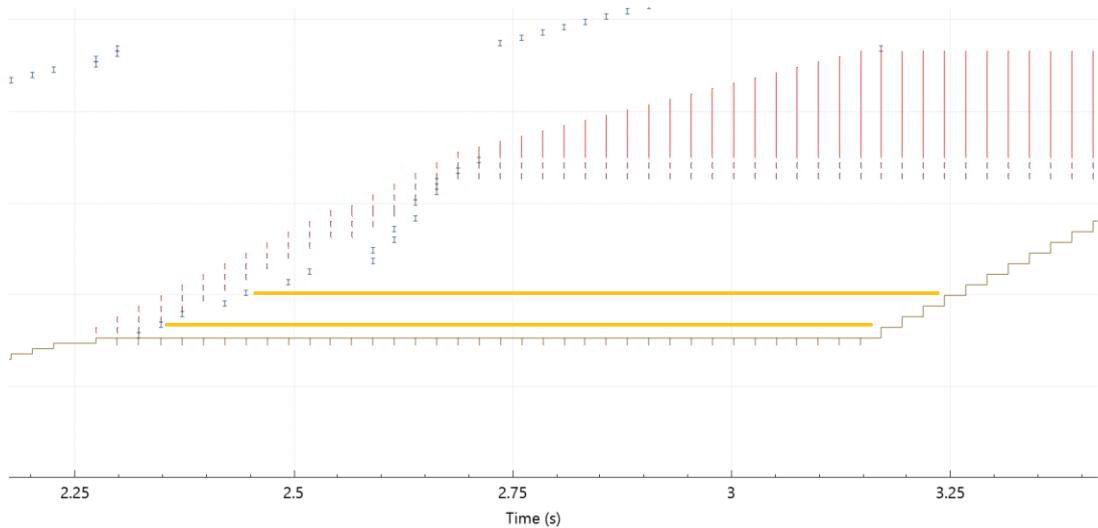
ביצעתו זום לגרף באזור המסומן בכתום ולאחר מכן ספרתי את הנקודות הכהולות באמצעות העכבר. לפי הספירה, זה יהיה 18 חבילות `retransmit`.

בהתואנה לנטואה שחוسبة בסעיף יח, שם מצאתי כי 18 חבילות נפלו, ניתן לראותות שהנתואנה תואמת את מה שחווסף קודם לכן. כמובן, מספר החבילות ששודרו מחדש אכן תואם את מספר החבילות שאבדו כפי שנמצא בסעיף יח, מה שמצויב על עקביות נתונים.

כג. מה מייצג החלק בקוו בו השיפוע בקוו התחתיו של ה-ACK הוא 0?

שיפוע 0 בקוו התחתיו של ה-ACK בgraf מציין על תקופה שבה נשלחות הודעות שהמקבל לא מקבל את החבילה הבאה בסדר הנכון וממשיכר לאשר את החבילה האחורונה שהתקבלה. מכיוון שציר ה-Y מייצג את מספרי הסequence, שיפוע שטוח מעיד על כך שהשלוח מקבל שוב ושוב את ACKים, מה שמצוין על חבילה שאבדה או לא התקבלה כראוי.

כד. לפי הגרף, כמה זמן לקח משליחת retransmit ועד שהגיע ה-ack אליו?



הזמן שבער בין שליחת ה retransmit ועד שהתקבל ה ACK עליו הוא בין 0.75-0.85 (בערך) שניות.

$$3.20 - 2.35 = 0.85$$

$$3.25 - 2.45 = 0.8$$

$$3.40 - 2.65 = 0.75$$