# **תשובות לתרגיל 3** - נרקיס שלו קרמיזי 205832447 ואלעד ישראל 313448888

1. מחשב A שולח הודעה למחשב B, ובמסלול ביניהם יש 2 נתבים.

הMTU בין A ל R1, וכן בין R1 ל R2 הוא 1500B.

הMTU בין R2 לבין B הוא 660B.

מחשב A שולח למחשב B כמות של 1980 בתים של מידע (שכבת אפליקציה).

נתון: גודל תחילית UDP 10B, תחילית TCP של 20B ותחילית IP של 20B.

תארו כיצד ישלח המידע באמצעות UDP ובאמצעות TCP.

בתשובתכם יש לציין את ערכי השדות: IP-ID, Length, MF, DF, Offset.

---

**:UDP**

מA לR1-
שכבת האפליקציה מורידה לשכבת התעבורה 1980B.
שכבת התעבורה מוסיפה 10B של תחילית UDP ומעבירה לשכבת הרשת.
שכבת הרשת מוסיפה 20B של תחילית IP ובסה"כ היא מנסה להעביר לשכבת הערוץ 2010B.
הMTU הוא 1500B. שכבת הרשת לא יכולה להעביר לשכבת הערוץ יותר מהMTU ולכן אין לה ברירה אלא לעשות פרמנטציה כיוון שבUDP הערך הדיפולטיבי של השדה DF הוא 0.

במקום החבילה המקורית שהיא גדולה מידי למעבר שכבת הרשת מחלקת אותה ל2 פרגמנטים. כל פרגמנט כזה הוא חבילה בפני עצמו ולכן לכל פרגמנט כזה יהיה header IP משלו עם השדות ID,Length,MF(More Fragment),Offset.

מתוך ה1990B שקיבלנו משכבת התעבורה אנחנו לוקחים 1480=1500-20. למה ? כי אנחנו מוסיפים header IP שעולה לנו 20 בתים אז נשאר לנו 1480 בתים לdata.

נביט בשדות של הפרגמנטים.

פרגמנט 1:
ערך השדה Length הוא 1500 כיוון שיש 1480 בתים + 20 בתים של header IP.
ערך השדה ID הוא הID של החבילה המקורית כי מדובר בתת חבילה(פרגמנט) ששייכת לחבילה המקורית.
ערך השדה MF הוא 1 כי אחרי הפרגמנט הזה יהיו עוד פרגמנטים ששייכים לחבילה המקורית, שהרי לא הצלחנו להכניס את כל הdata של החבילה המקורית.
הOffset הוא 0 מכיוון שזהו הפרגמנט הראשון – אין לפניו בתים בחבילה המקורית.

נשארו לנו עוד 510B שצריך לשלוח (1990-1480).

פרגמנט 2:
ערך השדה Length הוא 530 כיוון שיש 510 בתים + 20 בתים של header IP.
ערך השדה ID הוא הID של החבילה המקורית מאותה סיבה כמו קודם.
ערך השדה MF הוא 0 כי זהו הפרגמנט האחרון של החבילה המקורית. אין אחריו עוד פרגמנטים.
הOffset הוא 185 מכיוון שבפרגמנט שהיה לפניו היו 1480 ואם נחלק את 1480 ב8 נקבל 185.

מR1 לR2- הפרגמנטים יעברו מכיוון שהMTU גדול/שווה לכמות הבתים בכל פרגמנט.

מR2 לB-

הMTU הוא 660B. שכבת הרשת לא יכולה להעביר לשכבת הערוץ יותר מהMTU ולכן אין לה ברירה אלא לעשות פרמנטציה לפרגמנט 1.

מתוך ה1480B של מידע שמרכיבים את הפרגמנט הראשון אנחנו לוקחים 20-660=640. למה ? כי לכל פרגמנט אנחנו מוסיפים header IP שעולה לנו 20 בתים אז נשאר לנו 640B לdata.

נביט בשדות של הפרגמנטים של פרגמנט 1.

פרגמנט 1.1:
ערך השדה Length הוא 660 כיוון שיש 640 בתים + 20 בתים של header IP.
ערך השדה ID הוא הID של החבילה המקורית כי מדובר בתת חבילה(פרגמנט) ששייכת לחבילה המקורית.
ערך השדה MF הוא 1 כי אחרי הפרגמנט הזה יהיו עוד פרגמנטים ששייכים לחבילה המקורית, שהרי לא הצלחנו להכניס את כל הdata של פרגמנט 1.
הOffset הוא 0 מכיוון שזהו הפרגמנט הראשון – אין לפניו בתים בחבילה המקורית.

נשארו לנו עוד 640-1480=840B שצריך לשלוח מפרגמנט 1.

פרגמנט 1.2:
ערך השדה Length הוא שוב 660 כיוון שיש 640 בתים + 20 בתים של header IP.
ערך השדה ID הוא הID של החבילה המקורית מאותה סיבה כמו קודם.
ערך השדה MF הוא 1 כי אחרי הפרגמנט הזה יהיו עוד פרגמנטים ששייכים לחבילה המקורית, שהרי לא הצלחנו להכניס את כל הdata של פרגמנט 1.
הOffset הוא 80 מכיוון שבפרגמנט שהיה לפניו היו 640 ואם נחלק את 640 ב8 נקבל 80.

נשארו לנו עוד 640-840=200B שצריך לשלוח מפרגמנט 1.

פרגמנט 1.3:
ערך השדה Length הוא 220 כיוון שיש 200 בתים + 20 בתים של header IP.
ערך השדה ID הוא הID של החבילה המקורית מאותה סיבה כמו קודם.
ערך השדה MF הוא 1 כי אחרי הפרגמנט הזה יהיו עוד פרגמנטים ששייכים לחבילה המקורית **מפרגמנט 2**, שהרי לא הצלחנו להכניס את כל הdata של החבילה המקורית לפרגמנט 1.
הOffset הוא 160 מכיוון שבפרגמנט שהיה לפניו היו 1280 ואם נחלק את 1280 ב8 נקבל 160.

לבסוף ישלח פרגמנט 2 בצורתו המקורית מכיוון שהוא קטן מהMTU.

**<u>TCP:</u>**

שכבת האפליקציה מורידה לשכבת התעבורה 1980B.

בTCP אנחנו מנסים להימנע מפרגמנטציה ע"י שימוש בסגמנטציה מראש.

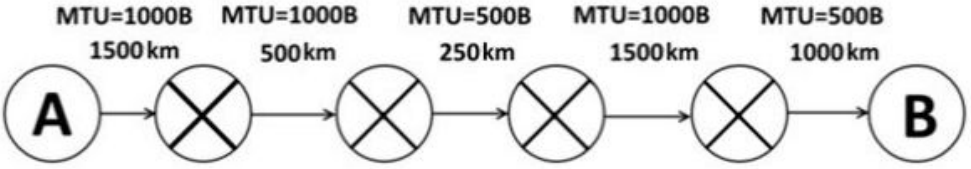
TCP יודע את ה MTU של הערוץ שלו. הוא גם יודע את הMTU של הערוץ שמחובר ליעד(כלומר מR2 לB) כי זה חלק מהמידע שהועבר לו בתהליך החיבור. TCP יודע שאין טעם לשלוח יותר מהמינימום של שני הערכים האלה אבל הוא עוד לא יודע את המינימום לאורך המסלול כי הוא לא יודע את הMTU של הראוטרים שבדרך(כלומר מR1 לR2). לכן, הוא מבצע את האלגוריתם Path MTU Discovery.

באמצעות אלגוריתם זה הוא מגלה את הMTU המינימלי לכל אורך המסלול- 660B, ומפצל את החבילה המקורית ל4 סגמנטים: 3 הראשונים בגודל 660B(מידע בגודל 620 + תחילית TCP בגודל 20 + תחילית IP בגודל 20), והאחרון 160B(מידע בגודל 120 + תחילית TCP בגודל 20 + תחילית IP בגודל 20).

מכיוון שגודל כל סגמנט הוא לכל היותר גודל הMTU המינימלי של המסלול מתקיים שכל הסגמנטים יעברו בהצלחה מA לB דרך הראוטרים.

נשים לב שTCP רוצה לדעת אם הMTU נהיה קטן יותר לאורך המסלול (מה שיגרום לפיצול של סגמנטים). לכן הוא יקבע את ערך הדגל DF להיות 1. בדרך זו הסגמנטים שלא הצליחו לעבור עקב הMTU הקטן מידי יחזירו שגיאה וכך הוא ידע להקטין את גודל הסגמנטים הבאים.

.2




במסלול בין A ל-B יש 4 נתבים, המרחקים של הערוצים וכמו כן ה MTU בערוצים מופיעים באיור לעיל ונניח ש-A שולח חבילה ל-B בגודל 2000 בתים בפרוטוקול UDP.
נתון:

- שידור FIFO וקצב שידור בכולם הוא 100KBps. מהירות ההתפשטות 250,000 ק"מ בשניה.
- במסלול בין A ל-B החבילה פוגשת ב4 חבילות (בגודל זהה 500 בתים) נוספות בכל נתב אי זוגי (כלומר הנתב הראשון והשלישי).
- ה MTU בערוץ הראשון, השני והרביעי (משמאל לימין) הוא 1000 בתים, ובערוץ השלישי והחמישי הוא 500 בתים.
- הניחו כי גודל תחילית IP הוא 20 בתים, גודל תחילית UDP הוא 10 בתים (מקורב לשם נוחות), וגודל תחילית TCP הוא 20 בתים.

א. חשבו כמה זמן עד ש-B מקבל את החבילה.
ב. כמה בתים מקבל B כאשר נשלח בצורה זו קובץ בגודל 1,000,000 בתים (שמחולק ל 500 חבילות בגודל 2000 בתים)?
ג. כמה בתים מקבל B כאשר נשלח בצורה זו קובץ בגודל 1,000,000 בתים בפרוטוקול TCP?

---

א. Am לR1-
שכבת האפליקציה מורידה לשכבת התעבורה 2000B.
שכבת התעבורה מוסיפה 10B של תחילית UDP ומעבירה לשכבת הרשת. כלומר נותר להעביר 2010B של מידע.
שכבת הרשת מוסיפה 20B של תחילית IP ובסה"כ היא מנסה להעביר לשכבת הערוץ 2030B.
הMTU הוא 1000B. שכבת הרשת לא יכולה להעביר לשכבת הערוץ יותר מהMTU ולכן אין לה ברירה אלא לעשות פרמנטציה כיוון שבUDP הערך הדיפולטיבי של השדה DF הוא 0.

במקום החבילה המקורית שהיא גדולה מידי למעבר שכבת הרשת מחלקת אותה ל3 פרגמנטים - שניים בגודל 1000 (980 בתים של מידע + 20 תחילית IP) ואחד בגודל 70 (50 בתים של מידע + 20 תחילית IP).

כאמור, בנתב R1 החבילה פוגשת ב4 חבילות מגודל 500B כל אחת, הנמצאות בתור לפניה.

מR1 לR2- הפרגמנטים יעברו מכיוון שהMTU גדול/שווה לכמות הבתים בכל פרגמנט.

מR2 לR3-

הMTU הוא 500B. שכבת הרשת לא יכולה להעביר לשכבת הערוץ יותר מהMTU ולכן אין לה ברירה אלא לעשות פרמנטציה לפרגמנט 1.

במקום הפרגמנט הראשון שהוא גדול מידי למעבר שכבת הרשת מחלקת אותו ל3 פרגמנטים - שניים בגודל 500 (480 בתים של מידע + 20 תחילית IP) ואחד בגודל 40 (20 בתים של מידע + 20 תחילית IP).

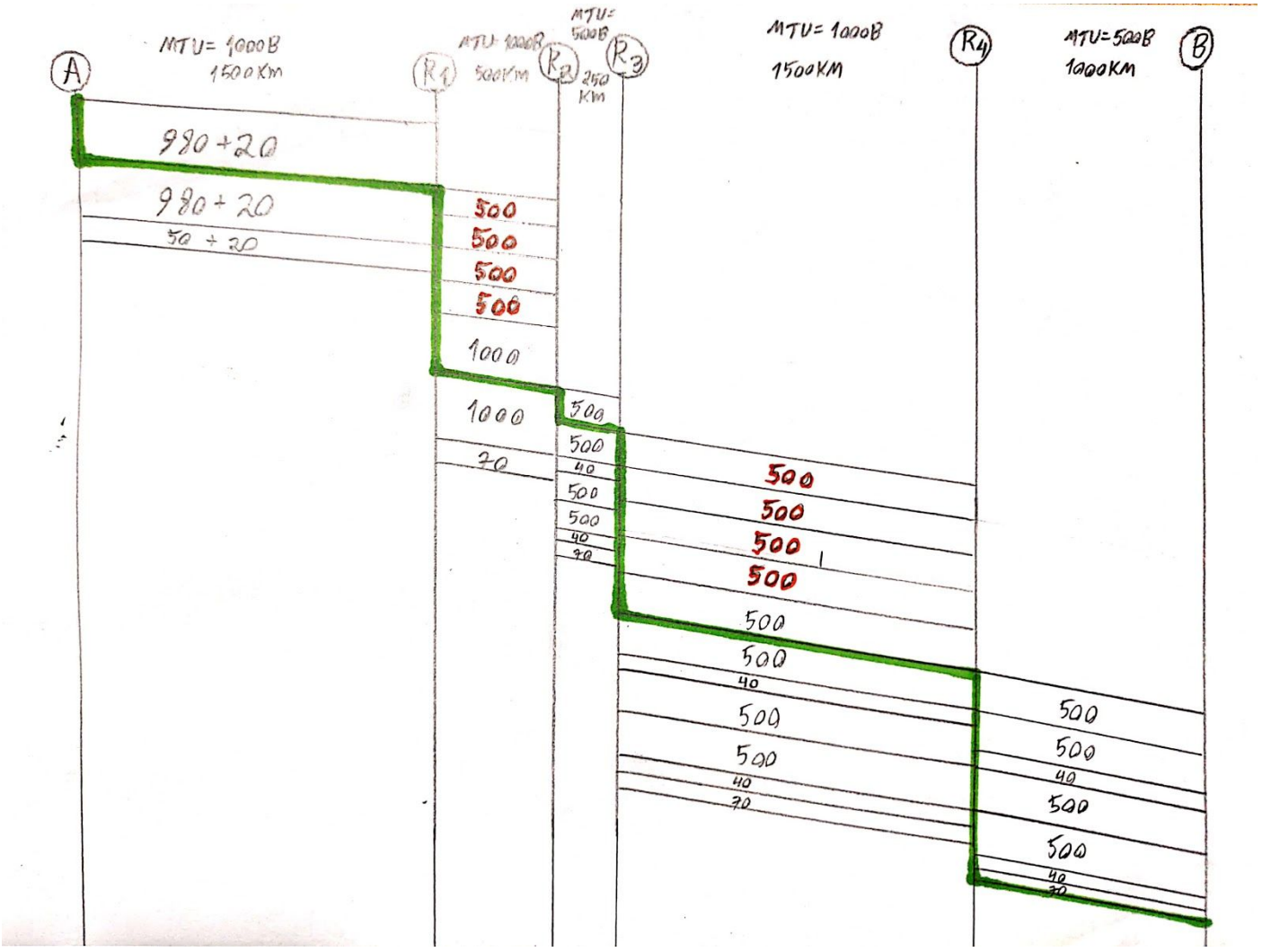
כנ"ל לגבי הפרגמנט השני.

הפרגמנט השלישי יעבור מכיוון שהMTU גדול/שווה לכמות הבתים בפרגמנט.

כאמור, בנתב R3 החבילה פוגשת ב4 חבילות מגודל 500B כל אחת, הנמצאות בתור לפניה.

מR3 לR4 ומR4 לB- הפרגמנטים יעברו מכיוון שהMTU גדול/שווה לכמות הבתים בכל פרגמנט.

חישוב זמנים:



ראשית, נחשב את השהיית ההתפשטות -

$$d_p = \frac{1500 + 500 + 250 + 1500 + 1000}{250000} = 0.019sec$$

שנית, נחשב את השהיית השידור -

$$d_t + d_q = \frac{1000 + 3000 + 500 + 2500 + 2150}{100 * 10^3} = 0.0915sec$$

1000 עבור מעבר החבילה הראשונה בערוץ הראשון, 3000 עבור מעבר 4 החבילות שבתור ואז החבילה הראשונה בערוץ השני, 500 עבור מעבר החבילה הראשונה בערוץ השלישי, 2500 עבור מעבר 4 החבילות שבתור ואז החבילה הראשונה בערוץ הרביעי ו2150 עבור מעבר כל החבילות בערוץ האחרון.

סה"כ לוקח 0.1105=0.019+0.0915 שניות עד שB מקבל את החבילה.

ב. ראינו קודם שכל חבילה בגודל 2000 בתים מתפצלת לפרגמנטים כך שלבסוף נשלחים בפועל 2150 בתים. לכן, אם יש 500 חבילות שכל אחת בגודל 2000 אזי בסופו של דבר ישלחו 2150*500=1075000 בתים.

ג. בTCP אנחנו מנסים להימנע מפרגמנטציה ע"י שימוש בסגמנטציה מראש.

TCP יודע את ה MTU של הערוץ שלו. הוא גם יודע את הMTU של הערוץ שמחובר ליעד(כלומר מR4 לB) כי זה חלק מהמידע שהועבר לו בתהליך החיבור. TCP יודע שאין טעם לשלוח יותר מהמינימום של שני הערכים האלה אבל הוא עוד לא יודע את המינימום לאורך המסלול כי הוא לא יודע את הMTU של הראוטרים שבדרך. לכן, הוא מבצע את האלגוריתם Path MTU Discovery.

באמצעות אלגוריתם זה הוא מגלה את הMTU המינימלי לכל אורך המסלול- 500B, ומפצל את החבילה המקורית ל 2173=1000000/460 סגמנטים בגודל 500 בתים(460B של מידע + 20B של תחילית TCP ועוד 20B של תחילית IP) ועוד סגמנט אחד בגודל 460=420+40. סה"כ מספר הבתים יהיה -

2173 * 500 + 460 = 1086960

3. לקוח רוצה לשלוח הרבה מידע לשרת. נתון:

- הבאפר אצל השרת הוא 360B.
- משתמשים במספרים סידוריים יחסיים המתחילים מ 0.
- ה MSS=360B
- אין delayed Acks.
- האפליקציה קוראת בקצב שליש כאשר היא מקבלת מידע (על כל 3 בתים שהיא מקבלת היא מספיקה לקרוא 1), וכאשר היא לא מקבלת מידע (כלומר, בזמן שלוקח לack להתפשט ולbyte הראשון של החבילה הבאה להגיע), היא מספיקה לקרוא 40 בתים.
- אין שימוש במנגנונים שנועדו להתמודד עם silly window syndrome.

א. הראו באמצעות דיאגרמת חבילות (ללא חישוב זמנים) את הקמת החיבור ואת החבילות הנשלחות עד שהלקוח יודע בוודאות שהשרת קיבל לפחות 560 בתים. עבור כל החבילות יש לציין את השדות:

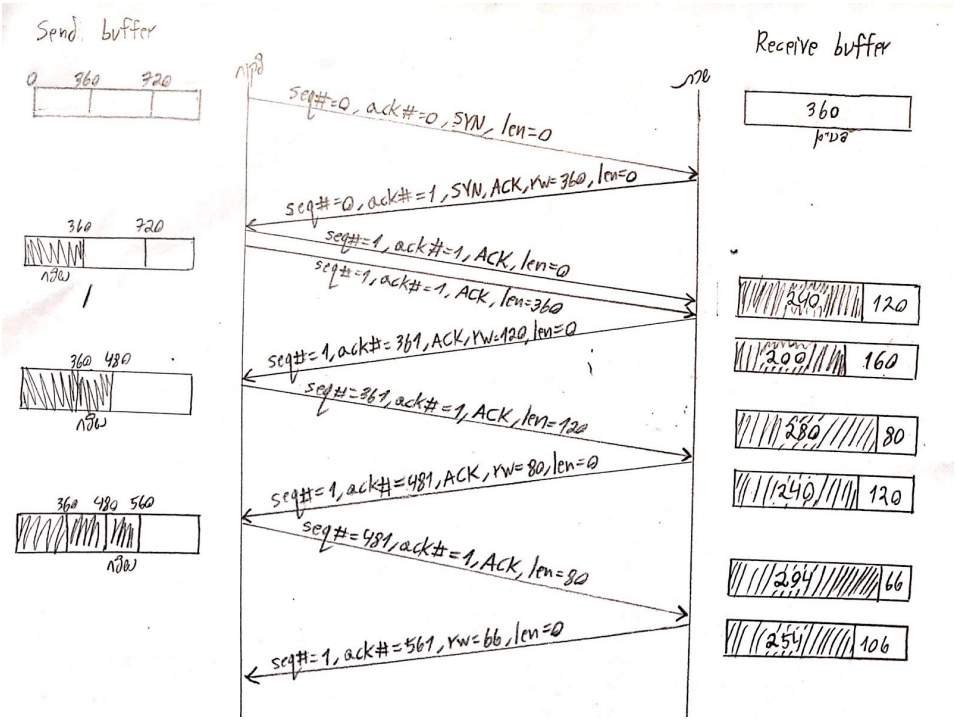
seq #, ack #, דגלי Syn ו Ack, ReceiveWindow, Data length

כמו-כן, יש לצייר את באפר השליחה של הלקוח ואת באפר הקבלה של השרת לאורך הדיאגרמה.

ב. הסבירו את בעיית ה silly window syndrome כפי שמתבטאת בחלק א', והסבירו כיצד המנגנון המתאים היה פותר אותה.

---

**א.**



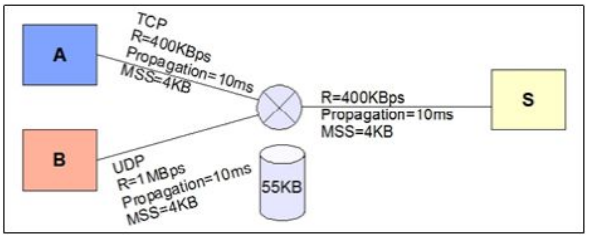

ב. ניתן לראות בסעיף א' שקורה המצב שנקרא silly window syndrome. במצב זה האפליקציה בצד המקבל (השרת) קוראת את המידע מהבאפר לאט יותר מהקצב שבו היא מקבלת את המידע ואז בכל ack החלון שנשלח קטן יותר ויותר. דבר זה בעייתי מכיוון שזה גורם לשליחת חבילות קטנות כאשר אם היינו מחכים מעט לפינוי של הבאפר היינו יכולים לשלוח חבילות גדולות יותר ולא לבזבז בתים על תחיליות.

המנגנון המתאים לפתרון הבעיה הוא המנגנון הבא -

אם המקבל רואה שגודל החלון קטן מ1MSS אזי הוא ישלח בreceiveWindow שהמקום שפנוי אצלו בבאפר הוא 0. בצורה זו, המקבל מונע מצב שבו השולח ישלח חבילות קטנות עד שיתפנה מספיק מקום בבאפר.

אם היינו משתמשים בסעיף א' במנגנון זה אזי בack הראשון מהשרת ללקוח היה נשלח 360 בreceiveWindow כי הבאפר ריק, אבל בack השני היה נשלח 0 כיוון שבבאפר יש מקום רק ל120 בתים ולא ל1MSS. ברגע שהיה מתפנה מקום ל360 בתים המקבל היה שולח לשולח ack עם 360 בreceiveWindow והשולח היה יכול לשלוח חבילה גדולה בגודל 360 בתים.

4. נתונה הרשת הבאה:



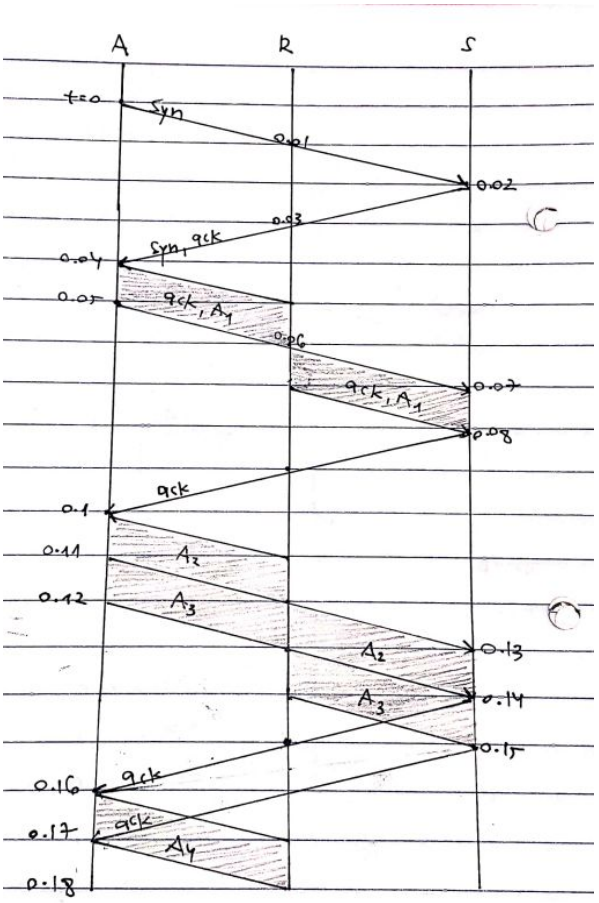

נתון:

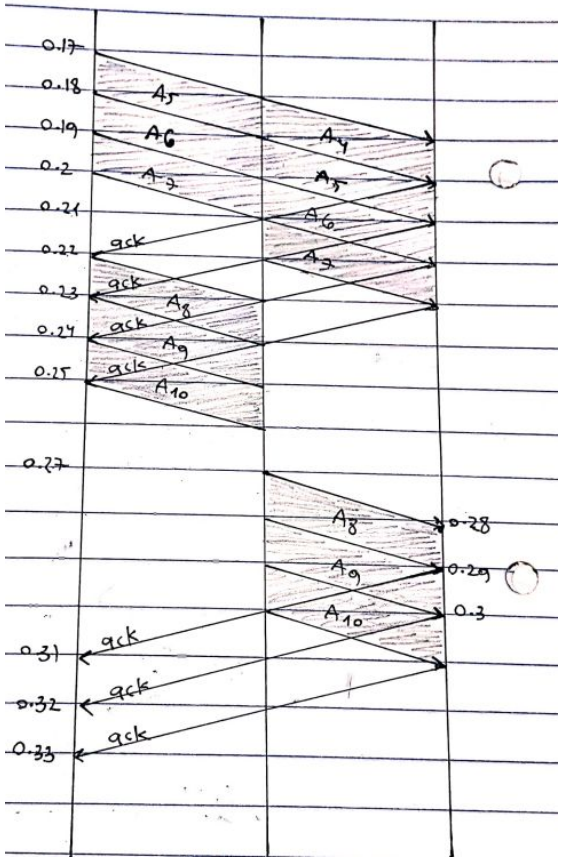
- לקוח A מעלה קובץ בגודל 40KB לשרת S מעל TCP.
- לקוח B מעלה קובץ בגודל 20KB לשרת S מעל UDP.
- בזמן 0 לקוח A פונה לשרת S לצורך העלאת הקובץ בגודל 40KB מעל TCP, וכעבור 0.2 שניות הלקוח B מתחיל בהעלאת הקובץ שלו, קובץ בגודל 20KB לשרת S מעל UDP.
- התור בנתב בגודל 55KB.
- הבאפר בשרת S בגודל 100KB.
- ה-MSS בגודל 4KB.
- אין delayed ACK.
- קצב שידור של לקוח A, שרת S והנתב הוא 400KBps, וקצב שידור של B הוא 1MBps.
- timeout=0.1seconds.
- השהיית ההתפשטות בכל הערוצים 10 מילישניות.
- התעלמו מ-Headers ומהשהיית השידור של הודעות בקרה (ACK).

הדגימו באמצעות תרשים חבילות וזמנים מזמן 0 ועד שהלקוח A יודע בוודאות שהשרת S קיבל את הקובץ בשלמותו. הציגו חישוב זמנים מפורט וחשבו במדויק את הזמן עד הרגע ש-A
יודע בוודאות שהקובץ התקבל אצל השרת בהצלחה.

---

תרשים:



ראשית, נחשב את השהיית השידור בערוצים השונים עבור חבילה בגודל MSS:
השהיית השידור בערוץ בין A לR ובערוץ בין R לS -

4KB/400KBps=0.01s

השהיית השידור בערוץ בין B לR -

4KB/1MBps=0.004s

מאחר וA מעלה את הקובץ לשרת מעל TCP הוא פותח בחיבור שכולל SYN, לאחר מכן SYN וACK ולאחר מכן ACK ביחד עם המידע הראשון. נתון לנו שיש להתעלם מהשהיות השידור של ACKים. לכן, נתחשב רק בהשהיות התפשטות של התחיליות המשתתפות בתהליך החיבור. יש 2 תחיליות כאלו שלא כוללות מידע. כל תחילית מתפשטת מA לR ולאחר מכן מR לS. נתון לנו שהשהיית ההתפשטות בכל הערוצים היא 10 מילישניות שזה 0.01 שניות. לכן, סה"כ שליחת המידע עצמו מתחילה רק בשניה ה0.04.

A מחלק את החבילה ל10 סגמנטים בגודל MSS כל אחד (סה"כ יחד A שולח 40KB, נתון לנו שיש להתעלם מתחיליות). נסמן את הסגמנטים הללו בAi כאשר i נע בין 1 ל10.

השליחה מתבצעת בslow start ולכן A מתחיל עם חלון בגודל 1MSS כך שהחלון גדל אקספוננציאלית בכל פעם.

בזמן 0.04 שניות A מתחיל לשדר את חבילה A1, השהיית השידור בערוץ בין A לR ובערוץ בין R לS היא 0.01 והשהיית ההתפשטות היא 0.01 ולכן בזמן 0.06 A1 מגיעה לR ובזמן 0.08 A1 מגיעה לS. בזמן 0.08 S מתחיל לשדר את הACK על A1, בזמן 0.09 הACK מגיע לR ובזמן 0.1 הACK מגיע לA.

החלון גדל ל2MSS ובאותו אופן משדרים את A2 ואת A3 ואת הACKים עליהם. החלון גדל ל4MSS ובאותו אופן משדרים את A4, את A5, את A6 ואת A7 ואת הACKים עליהם.

בזמן 0.2, A מסיים לשדר לR את A7 וחבילה זו מסיימת להתפשט לR בזמן 0.21. בנוסף, בדיוק באותו זמן (0.2) B מתחיל לשלוח מידע. מאחר וB מעלה את הקובץ לשרת מעל UDP הוא לא פותח בחיבור וישר מתחיל בשליחת חבילות. כמו כן, אין ב UDP חלונות שליחה ולכן הוא פשוט שולח את החבילות שלו אחת אחרי השנייה.

B מחלק את החבילה ל5 פרגמנטים בגודל MSS כל אחד (סה"כ יחד B שולח 20KB, נתון לנו שיש להתעלם מתחיליות). נסמן את הפרגמנטים הללו בBi כאשר i נע בין 1 ל5.

בזמן 0.2 שניות B מתחיל לשדר את חבילה B1, השהיית השידור בערוץ בין B לR היא 0.004 והשהיית ההתפשטות היא 0.01 ולכן בזמן 0.214 B1 מגיעה לR. אמרנו קודם שחבילה A7 מגיעה לR בזמן 0.21. כלומר, חבילה B1 מגיעה אחריה ומשודרת אחריה בזמן 0.22. לאחר מכן, משודרים מB ארבעת הפרגמנטים הנוספים אחד אחרי השני, לכל אחד לוקח 0.004 שניות להגיע לR (מספיק לקחת בחשבון התפשטות של חבילה אחת). סה"כ כל הפרגמנטים מגיעים לR עד זמן 0.23 שניות.

בזמן 0.22 מגיע ACK על A4 ולכן A מתחיל לשדר לR את A8 וחבילה זו מסיימת להתפשט לR בזמן 0.24. כלומר, חבילה A8 מגיעה לR אחרי כל החבילות מB. לכן, A8 ואז גם A9 וA10 יחכו בתור עד סיום השידור של כל החבילות מB.
כיוון שR מתחיל לשדר את B1 בזמן 0.22, עד 0.23 היא משודרת וזה בדיוק הזמן בו כל 4 החבילות הנוספות של B מגיעות ל R. לכן, משדרים אותן אחת אחרי השנייה ומקבלים שבזמן 0.27 השידור הסתיים וניתן לשדר את חבילות A8 עד A10.

בזמן 0.31 שניות 3 החבילות מסיימות להשתדר ולהתפשט לS. לכן, בזמן 0.31 S מתחיל לשדר את הACK על A10(הסגמנט האחרון), בזמן 0.32 הACK מגיע לR ובזמן 0.33 הACK מגיע לA.
בעצם כאשר הACK האחרון מגיע לA, A יודע בוודאות שהקובץ התקבל אצל השרת בהצלחה. לכן התשובה היא שלאחר 0.33 שניות A יודע בוודאות שהקובץ התקבל אצל השרת בהצלחה.

5. לקוח ושרת מחוברים ביניהם בעזרת ראוטר. לקוח פותח חיבור לשרת ושולח בקשה (קצרה) להורדת קובץ בגודל 16KB מהשרת (הניחו זמן שידור הבקשה זניח, ולכן ניתן להתעלם ממנו).

נתון:

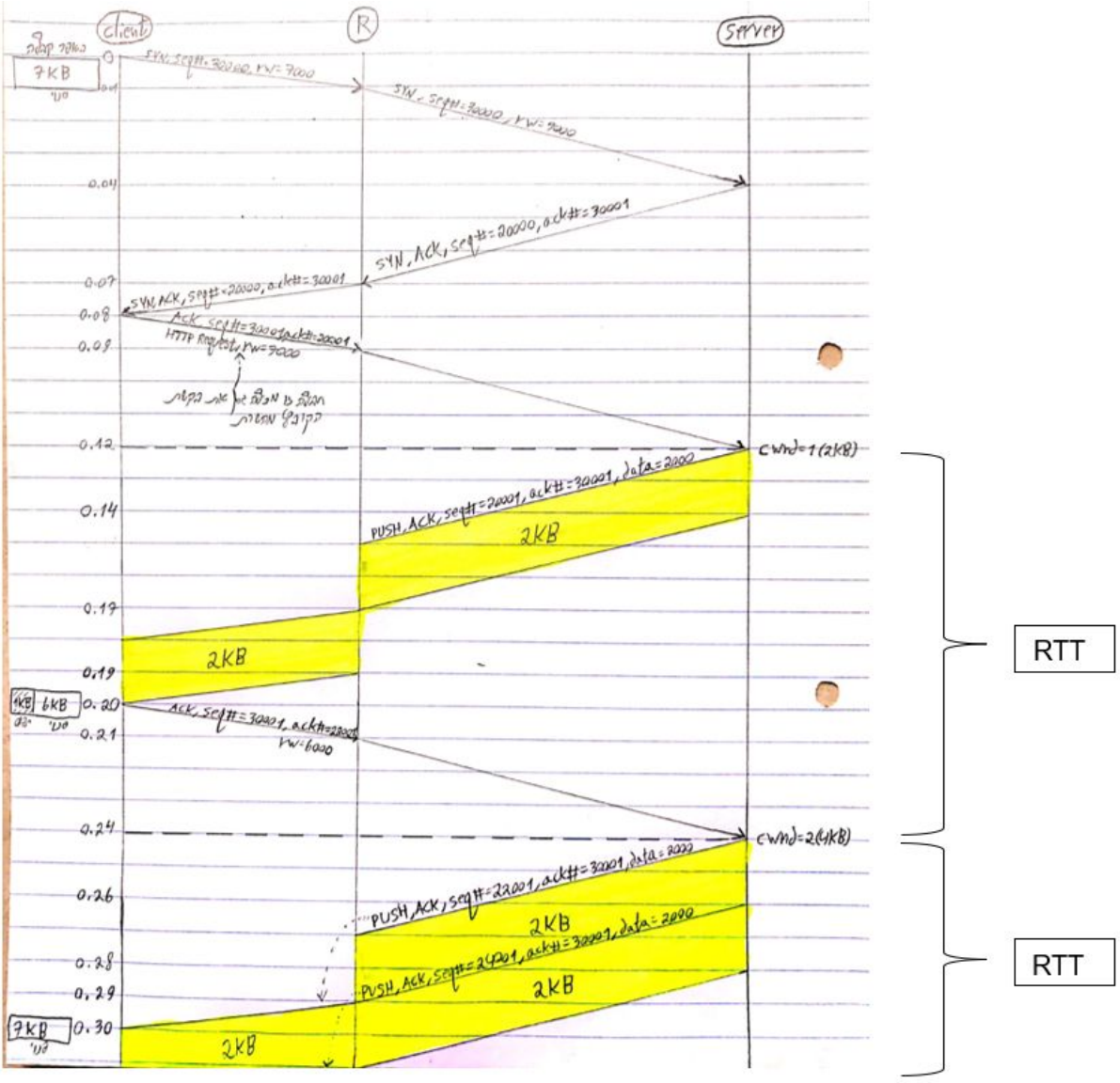
- קצב שידור בשני הערוצים (בשני כיוונים) הוא 1MBps
- מהירות התפשטות 250000 ק"מ לשניה.
- מרחק בין הלקוח לנתב הוא 250 ק"מ ובין נתב לשרת 750 ק"מ.
- ה-MSS הוא 2KB (התעלמו מ-headers), ועם סיום קבלת החבילה תמיד נשלח ACK.
- הודעות ה ACK קטנות וקצב שידור שלהן זניח (לכן ניתן להתעלם מזמן השידור שלהן).
- התור בנתב בגודל 4KB ואין תעבורה אחרת ברשת.
- הניחו שהאפליקציה בלקוח קוראת מה-buffer בקצב קבוע של 0.5MBps. גודל ה-receiveBuffer הוא 7KB.

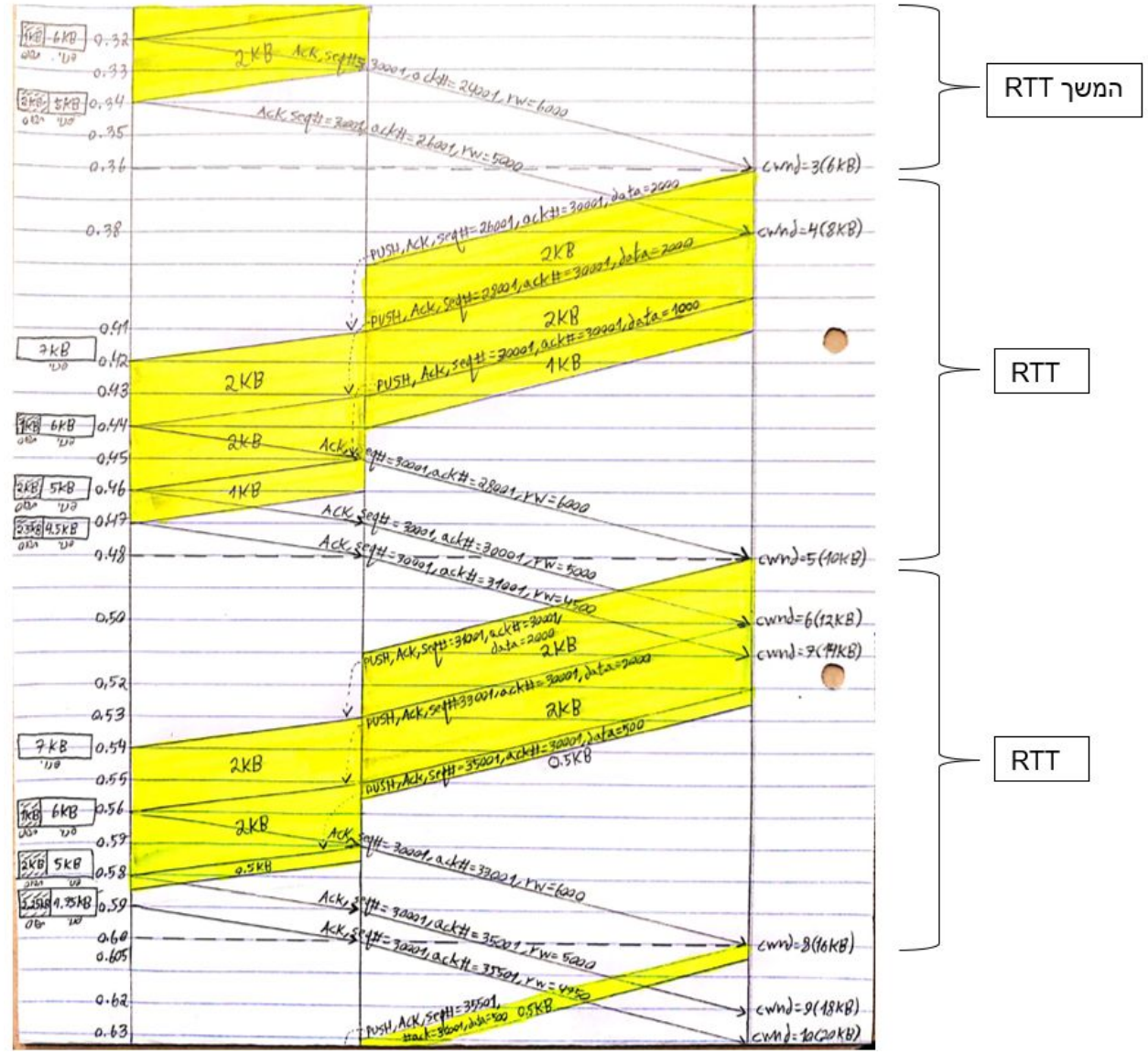
א. הראו דיאגרמת זמנים וחבילות בין השרת והלקוח כולל הקמת וסגירת החיבור (המספר הסידורי ההתחלתי בשרת הוא 20000 ובלקוח הוא 30000). על כל חבילה שנשלחת יש לציין את המספרים הסידוריים, גודל ה-receive Window, דגלי TCP, זמן וחלון בקרת עומס.

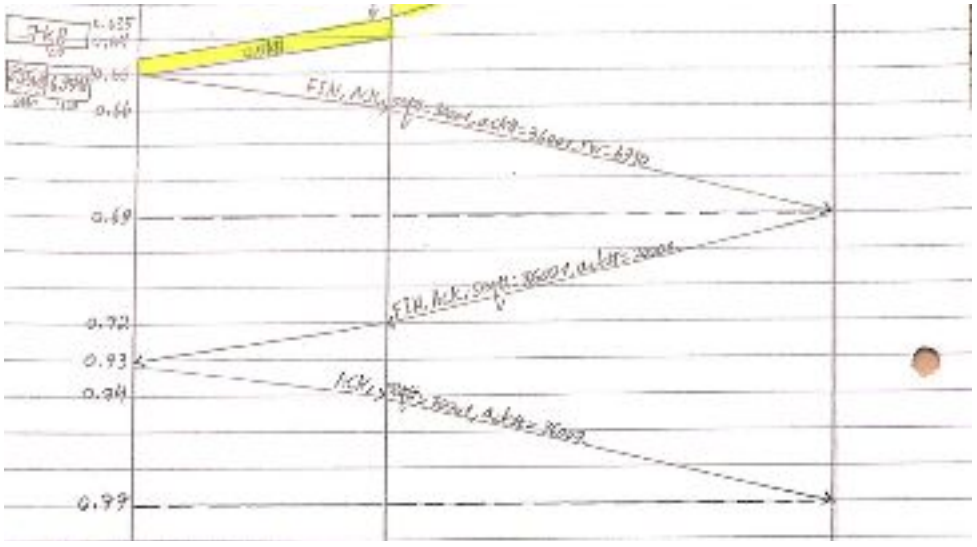
ב. חשבו כמה זמן יקח להעביר את הקובץ.

---

א. תרשים:





הערות לגבי הדיאגרמה:

- הזווית של הקווים בין הclient לR ובין R לserver ולהיפך אמורה להיות זהה מכיוון שקצב ההתפשטות זהה. בערוץ בין R לserver הזווית קצת יותר תלולה משיקולי נוחות בלבד.
- בצד ימין ניתן לראות את הcwnd שגדל ב1 בכל RTT, ומצד שמאל ניתן לראות את הreceive buffer של הclient ואת ציר הזמן בשניות.

הסבר -

ראשית, נחשב את השהיית השידור והשהיית ההתפשטות בערוצים השונים:
השהיית השידור עבור חבילה בגודל MSS בערוץ בין הClient לR ובערוץ בין R לServer (זהה ל2 הערוצים)-
$2KB/1MBps = (2*10^3)/10^6s = 2/1000s = 2ms$
השהיית ההתפשטות בערוץ בין הClient לR -
$250KM/250000KMps = 0.001s = 1ms$
השהיית ההתפשטות בערוץ בין R לServer -
$750KM/250000KMps = 0.003s = 3ms$

שנית, נשים לב שהאפליקציה קוראת מהבאפר בקצב קטן פי 2 מהקצב בו היא מקבלת (נתון שהאפליקציה בלקוח קוראת מהבאפר בקצב קבוע של 0.5MBps כאשר קצב השידור הוא 1MBps).

מזמן 0 שניות עד זמן 0.12 ניתן לראות את הקמת החיבור בין הלקוח לשרת שכוללת SYN, SYN+ACK וACK (שכולל בתוכו גם את הבקשה של הלקוח לשרת).

השליחה מתבצעת בslow start ולכן השרת מתחיל עם חלון בגודל 1MSS כך שהחלון גדל אקספוננציאלית בכל פעם.

בזמן 0.12 שניות השרת מתחיל לשדר את החבילה הראשונה שגודלה 2KB כך שבזמן 0.2 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 1KB. לכן, הבאפר שלו שהיה ריק עד כה (7KB מקום פנוי) כעת בתפוסה של 1KB ופניות של 6KB. לכן, בACK שנשלח בזמן 0.2 הלקוח מעדכן את השרת בrw בגודל 6KB.

בזמן 0.24 הACK על החבילה הראשונה מגיע לשרת והחלון גדל ל2MSS. השרת משדר את החבילה השנייה שגודלה 2KB כך שבזמן 0.32 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 1KB. לכן, הבאפר שלו שהספיק להתרוקן כבר (7KB מקום פנוי) כעת בתפוסה של 1KB ופניות של 6KB. לכן, בACK שנשלח בזמן 0.32 הלקוח מעדכן את השרת בrw בגודל 6KB.
בזמן 0.26 השרת משדר את החבילה השלישית שגודלה 2KB כך שבזמן 0.34 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 1KB. לכן, הבאפר שלו כעת בתפוסה של 2KB ופניות של 5KB. לכן, בACK שנשלח בזמן 0.34 הלקוח מעדכן את השרת בrw בגודל 5KB.

בזמן 0.36 הACK על החבילה השנייה מגיע לשרת והחלון גדל ל3MSS. השרת משדר את החבילה הרביעית שגודלה 2KB כך שבזמן 0.44 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 1KB. לכן, הבאפר שלו שהספיק להתרוקן כבר (7KB מקום פנוי) כעת בתפוסה של 1KB ופניות של 6KB. לכן, בACK שנשלח בזמן 0.44 הלקוח מעדכן את השרת בrw בגודל 6KB.
בזמן 0.38 הACK על החבילה השלישית מגיע לשרת והחלון גדל ל4MSS. השרת משדר את החבילה החמישית שגודלה 2KB כך שבזמן 0.46 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 1KB. לכן, הבאפר שלו כעת בתפוסה של 2KB ופניות של 5KB. לכן, בACK שנשלח בזמן 0.46 הלקוח מעדכן את השרת בrw בגודל 5KB.
בעיקרון מותר לשרת לשלוח עוד 2 חבילות לפי החלון אבל לפי הrw מותר לו לשלוח רק עוד 1KB (נשלחו כבר 4KB ומותר 5KB). לכן, בזמן 0.4 השרת משדר את החבילה השישית שגודלה 1KB כך שבזמן 0.47 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 0.5KB. לכן, הבאפר שלו כעת בתפוסה של 2.5KB ופניות של 4.5KB. לכן, בACK שנשלח בזמן 0.47 הלקוח מעדכן את השרת בrw בגודל 4.5KB.

בזמן 0.48 הACK על החבילה הרביעית מגיע לשרת והחלון גדל ל5MSS. השרת משדר את החבילה השביעית שגודלה 2KB כך שבזמן 0.56 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 1KB. לכן, הבאפר שלו שהספיק להתרוקן כבר (7KB מקום פנוי) כעת בתפוסה של 1KB ופניות של 6KB. לכן, בACK שנשלח בזמן 0.56 הלקוח מעדכן את השרת בrw בגודל 6KB.
בזמן 0.5 הACK על החבילה החמישית מגיע לשרת והחלון גדל ל6MSS. השרת משדר את החבילה השמינית שגודלה 2KB כך שבזמן 0.58 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 1KB. לכן, הבאפר שלו כעת בתפוסה של 2KB ופניות של 5KB. לכן, בACK שנשלח בזמן 0.58 הלקוח מעדכן את השרת בrw בגודל 5KB.
בזמן 0.51 הACK על החבילה השישית מגיע לשרת והחלון גדל ל7MSS.

בעיקרון מותר לשרת לשלוח עוד 5 חבילות לפי החלון אבל לפי הrw מותר לו לשלוח רק עוד 0.5KB (נשלחו כבר 4KB ומותר 4.5KB). לכן, בזמן 0.52 השרת משדר את החבילה התשיעית שגודלה 0.5KB כך שבזמן 0.585 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 0.25KB. לכן, הבאפר שלו כעת בתפוסה של 2.25KB ופניות של 4.75KB. לכן, בACK שנשלח בזמן 0.59 הלקוח מעדכן את השרת בrw בגודל 4.75KB.

בזמן 0.6 הACK על החבילה השביעית מגיע לשרת והחלון גדל לMSS8. השרת משדר את החבילה העשירית שגודלה 0.5KB (גודל הקובץ הוא 16KB ונשלחו עד כה 15.5KB) כך שבזמן 0.65 החבילה מגיעה ללקוח. הלקוח כבר מספיק לקרוא חצי מגודל החבילה שנשלחה אליו, כלומר, 0.25KB. לכן, הבאפר שלו שהספיק להתרוקן כבר (7KB מקום פנוי) כעת בתפוסה של 0.25KB ופניות של 6.75KB. לכן, בACK שנשלח בזמן 0.65 הלקוח מעדכן את השרת בrw בגודל 6.75KB.

ב. נתון שהודעות הACK קטנות וקצב השידור שלהן זניח. לכן, ניתן להתעלם מהשהיות השידור של שלושת התחיליות של הקמת החיבור ולקחת בחשבון רק את השהיות ההתפשטות שלהן מהלקוח לנתב (0.01) ומהנתב לשרת (0.03). כלומר,

0.12 = 0.04*3

סה"כ קיבלנו שהלקוח מתחיל לשלוח את הבקשה שלו בזמן 0.12. נתון שזמן שידור הבקשה זניח ולכן נתעלם ממנו.

נשים לב שהזמן שלוקח מתחילת שידור החבילה הראשונה ועד תחילת שידור החבילה האחרונה שקול ל4 פעמים RTT של חבילה בגודל 2KB.

RTT של חבילה בגודל 2KB מהשרת ללקוח לוקחת שידור והתפשטות של החבילה מהשרת לנתב ואז מהנתב ללקוח ועוד התפשטות של הACK מהלקוח לנתב ואז מהנתב לשרת -

0.12=0.02+0.03+0.02+0.01+0.03+0.01

סה"כ קיבלנו שכל החבילות חוץ מהאחרונה והACKים עליהן מגיעים עד זמן –

0.6=0.12*0.12+4

נותר לחשב את הRTT של החבילה האחרונה שהיא בגודל 0.5KB (כלולה כאן גם השהיית השידור של התחילית הראשונה של סגירת החיבור) –

0.09=0.005+0.03+0.005+0.01+0.03+0.01

**סה"כ קיבלנו שכל החבילות והACKים עליהן מגיעים עד זמן 0.69.**

ניתן להתעלם מהשהיות השידור של שתי התחיליות הנותרות של סגירת החיבור ולקחת בחשבון רק את השהיות ההתפשטות שלהן מהלקוח לנתב (0.01) ומהנתב לשרת (0.03). כלומר,

0.08 = 0.04*2

סה"כ קיבלנו שהחיבור נסגר עד זמן 0.77.

6. לארגון A ישנו ראוטר R1 אשר מחובר בערוץ ישיר לראוטר R2 של ארגון B.
קצב השידור של הערוץ הינו 100MBps.
בתוך ארגון A ישנם 10 לקוחות.
ובתוך ארגון B ישנו שרת שיכול לטפל בלקוחות בו זמנית.
בשאלה זו נתעלם מכל ההשהיות בתוך הרשתות המקומיות.
המרחק בין R1 לבין R2 הוא 2500m ומהירות ההתפשטות היא $2.5\cdot10^8$ מטר לשניה.
בשרת יש 10 קבצים שונים, כל אחד בגודל 10KB.
נניח MSS=1KB.
בשאלה זו ניתן להזניח את זמן השידור של התחיליות והודעות בקרה.
כנו כן, נניח שה timeout=1s.
כל הלקוחות מתחילים בפניה בו-זמנית אל השרת להורדה של קובץ אחד כל אחד על גבי TCP.

א. כמה זמן עובר עד שהשרת יודע שהמחשב האחרון קיבל את הקובץ?
ב. חשבו את גודל החוצצים המינימלים בנתבים R_1,R_2, על מנת שלא יהיה אובדן חבילות.

---

א. ראשית כל אחד מהלקוחות יוצר חיבור עם השרת. מכיוון שזמן השידור של תחיליות הוא זניח ע"פ נתוני השאלה מתקיים שזמן השידור של בקשות SYN וACK הוא זניח. לכן, נתייחס רק לזמן ההתפשטות של אלו.
נשים לב שיש לקחת בחשבון זמן התפשטות של חיבור עבור לקוח אחד בלבד מאחר ובקשות החיבור עבור כל הלקוחות קורות במקביל.
זמן ההתפשטות של בקשת SYN או ACK עבור לקוח בודד:

$d_p$=2500m/(2.5*$10^8$mps) = $10^{-5}$s = 0.01ms

בקשת חיבור מורכבת מבקשת SYN, בקשת SYN+ACK, וACK. סה"כ 3 * $10^{-5}$s.

כאשר החבילה השלישית משודרת מהשרת החבילה הראשונה כבר סיימה להתפשט והתקבל עליה ACK מהלקוח.
בצורה דומה כאשר נגיע לחבילה ה11(הסגמנט ה2 של הקליינט הראשון) - השרת יוכל כבר לשלוח אותה בלי להמתין מכיוון שהACK עבור החבילה ה1(הסגמנט ה1 של הקליינט הראשון) כבר התקבל מזמן. כנ"ל לשאר החבילות.
לכן נשים לב שנוצר פה מצב של full pipe- השרת יכול להמשיך לשדר כל הזמן מבלי להמתין לחבילות, מה שאומר שזמן השידור של כל 10 הקבצים היא

(10*10KB)/100MBps= 0.001s = 1ms

סה"כ הזמן שעובר עד שהשרת יודע שהמחשב האחרון קיבל את הקובץ הוא: זמן החיבור + זמן השידור של כל הקבצים + זמן ההתפשטות של החבילה האחרונה + ACK על החבילה האחרונה:

3*0.01ms + 1ms + 0.01ms + 0.01ms = 1.05ms

ב. כדי לענות על השאלה אנחנו צריכים לחשב מה המספר המקסימלי של חבילות שיכולות להיות בבאפר. על פי הנתונים רק R2 צריך באפר גדול. לכן, נמצא את המספר המקסימלי רק עבורו.
בהתחלה יש בבאפר 10KB שהם 10 סגמנטים - הסגמנט הראשון של כל לקוח.

ראינו כבר בסעיף הקודם שרק אחרי שידור החבילה השלישית מגיע הACK על החבילה הראשונה. כלומר, כאשר מגיע הACK הראשון יש בבאפר 7 חבילות.

השליחה מתבצעת בslow start ולכן מתחילים עם חלון בגודל 1MSS כך שהחלון גדל אקספוננציאלית בכל פעם. כלומר, על כל ACK שחוזר נוספות עוד 2 חבילות לחלון(ובעקבות כך- לבאפר) ומשודרת רק חבילה אחת עד הACK הבא.

כעת לשרת נשארו עוד 90 חבילות לשלוח (סה"כ צריך לשלוח 100 ועד כה שלח 10). לכן, בנוסף ל7 החבילות שכרגע בבאפר, במהלך 45 הACKים הבאים בכל פעם נוספות 2 חבילות (מגיעים ל90) ומשודרת רק חבילה אחת ולכן מתווספת חבילה אקסטרה - כלומר 45 חבילות. לכן, סה"כ מספר החבילות המקסימלי שימתינו בבאפר הוא 7+45=52. כל חבילה בגודל 1KB ולכן גודל הבאפר המינימלי צריך להיות 52KB.

7. אפליקציה מסויימת מתקשרת בעזרת ערוץ לוויני על גבי מרחק של 60,000Km כאשר באמצע אין נתבים. מהירות ההתפשטות היא 300,000Km/s. כלומר, השהיית ההתפשטות היא $d_p$=(60000km)/(300000km/s) = 0.2 sec
נניח שלמקבל יש באפר בגודל 200,000B.

א. נניח שקצב השידור הוא 10MBps והאפלקציה קוראת מידע מהבאפר בקצב קבוע של 8MBps כיצד ניתן לשפר את קצב העברת המידע?
ב. מהו החסם העליון על גודל הבאפר שיכול להיות מפורסם?

---

א. ראשית, נשים לב שכיוון שקצב השידור הוא 10MBps והאפליקציה קוראת מידע מהבאפר בקצב של 8MBps מתקיים שעל כל 10 בתים שנכנסים האפליקציה מספיקה כבר לקרוא 8.
נניח ושלחנו 200000 בתים - עד סיום ההתפשטות שלהם(עד שהאפליקציה תקבל את הבית האחרון) האפליקציה תספיק לטפל ב160000 בתים ולכן ברגע זה יש לה 160000 בתים פנויים בבאפר. ורק אז היא תשלח ACK ללויין, בו יצויין שחלון הקבלה שלה הוא 160000B. הלוויין לא יכול לשלוח מידע נוסף עד שיקבל את הACK הזה וידע כמה בתים הוא יכול לשלוח מבלי שהחבילה תיזרק עקב מחסור מקום בבאפר אצל האפליקציה. כלומר בזמן הזה שבין שליחת הACK ע"י האפליקציה לבין קבלת הבית הראשון של החבילה הבאה מהלויין - האפליקציה לא עושה כלום(למעט עיבוד 40000 הבתים הקודמים שהיא מסיימת במהרה). נרצה לנצל את הזמן הזה, כלומר שהלויין ישלח יותר מידע מגודל החלון כך שנתחשב במידע שמטופל תוך כדי ההתפשטות של מידע מהלויין לאפליקציה והACK על המידע. דבר זה נקרא "בקרת זרימה אגרסיבית"- האפליקציה תפרסם חלון זרימה(rw) הגדול מגודל החלון האמיתי(החלק הפנוי בבאפר).

ב. הנוסחא היא:

W/(1-O/R)

כאשר W זה גודל הבאפר האמיתי, O זה קצב הקריאה של האפליקציה, וR זה קצב הכניסה של המידע לבאפר.

נמחיש את השיפור בקצב העברת המידע:
לפני השיפור אם השולח ישלח מידע בגודל 200000B (גודל הבאפר) וימתין להעברת המידע, גם אם נתחשב רק בהשהיית ההתפשטות נקבל שקצב העברת המידע הוא

200000B/RTT=200000B/(0.2s+0.2s)=500KBps

לעומת זאת לאחר השיפור, האפליקציה תפרסם חלון בגודל:

200000B/(1-(8MBps/10MBps))=$10^6$B

ואז נקבל שקצב העברת המידע הוא:

$10^6$B/RTT=$10^6$B/(0.2s+0.2s)=2.5MBps

כלומר קצב העברת המידע גדל פי 5 מהקצב המקורי.