

תירגול מס' 4 – שכבת התעבורה



נושאים שמופיעים בקורת הרשת:

- שרתי שכבת ההובלה.
- ריבוב/פילוג.
- תעבורה חסרת קשר:
UDP.
- עקרונות תעבורת מידע
אמינה ברשת.
- תעבורה מונחת קשר:
TCP
 - מבנה המקטע.
 - תעבורת מידע אמינה.
 - בקרת זרימה.
 - ניהול הקשר
- עקרונות של בקרת עומסים.
- בקרת עומסים TCP.

תלכורת

□ נושאי מפתח:

- שירותי תעבורה שונים.
- Multiplexing (ריבוב) / Demultiplexing (פילוג)
- תעבורה UDP:connectionless
- עקרונות תעבורת מידע אמינה ברשת.

□ שירותי תעבורה ופרוטוקולים

- שירותי ופרוטוקולי התעבורה מספקים **חיבור לוגי** בין שני מחשבים ברשת. שני יישומים מחוברים כביכול דרך ערוץ ישיר.
- **פרוטוקולי התעבורה** רצים בחשבי הקצה: **הצד השולח** מחלק את המסרים לסגמנטים ומעביר אותם לרמת הרשת. **הצד המקבל** מרכיב מחדש את הסגמנטים למסרים ומעביר אותם לרמת היישום.
- קיימים מספר פרוטוקולי תעבורה, העיקריים באינטרנט הם **TCP** ו-**UDP**.

תלכורת

□ פרוטוקולי תעבורה באינטרנט:

- שירות תעבורה אמין **TCP** - שירות שבו הודעות מועברות ומתקבלות בסדר הנכון מבלי ללכת לאיבוד.

■ מאפיינים:

- בקרת עומס
- בקרת זרימה
- הקמת קשר במנגנון Handshake.

□ שירות תעבורה לא אמין **UDP** : **Best effort** - ייתכן איבוד

הודעות, הגעתן שלא על פי הסדר והכפלתן. בפרוטוקול זה יש הרחבה מינימלית של רמת ה IP.

□ שירותים שאינם קיימים באינטרנט:

- הבטחת רוחב פס מינימלי
- הבטחת השהייה מינימלית.
- כדי לתמוך ביישומי זמן אמת, כמו טלפוניה, צריך לתמוך בשירותים האלו.

איך מקצצים הפטת אמנות דרך ארץ/ רשת שאינם מקציצים אמנות?

□ Rdt1.0: פרוטוקול פשוט שמניח שהערוץ שמשתמשים בו אמין.

□ Rdt2.0: ערוץ תקשורת לא אמין.

■ הנחות:

□ אין אובדן חבילות. (ייתכן שיגיעו משובשות)

□ החבילות מגיעות בסדר הנכון.

□ כל הטעויות מתגלות על ידי מנגנון ה-checksum.

■ קיים היזון חוזר מהצד המקבל לצד השולח:

□ **ACK** (acknowledgment): אישור על קבלת חבילה תקינה.

□ **NAK** (negative acknowledgment): אישור על קבלת הודעה משובשת.

□ חבילה שהגיעה משובשת תשלח מחדש (retransmission).

■ תוספות שלא היו קיימות ב-Rdt1.0:

□ גילוי טעויות.

□ היזון חוזר מהצד המקבל לשולח (ACK, NAK)

□ Rdt2.1 פתרון שיבוש הודעות ACK ו-NAK:

- הנחות:

- אין אובדן מידע.

- אין שינוי בסדר החבילות.

- תוספות לפרוטוקול: מונה של ביט אחד.

□ rdt2.2: טיפול בשכפולי הודעות:

- שיפור לעומת rdt2.1: מונע שכפולי חבילות ו-ACK.

- אם יש שכפול של הודעת ACK אז השולח ישלח מחדש את החבילה הנוכחית ולא יהיה איבוד מידע.

- אין טיפול באובדן חבילות, ייתכן מצב בו המקבל והשולח יחכו זמן אינסופי.

□ rdt3.0: פרוטוקול הביט המתחלף:

- שיפור לעומת rdt2.1: ההתמודדות עם אובדן חבילות.

- **Timer**: השולח מחכה לACK פרק זמן "סביר" אשר קטן מהעיכוב המקסימלי (RTT). במידה ולא התקבל ACK בזמן הזה תתבצע שליחה מחודשת.

פרוטוקולי Pipeline

- בפרוטוקולים מסוג stop and wait נוצרה בעיה של ניצולת נמוכה, ניתן לשלוח הודעה אחת בכל פעם בלבד.
- פרוטוקול (GBN) Go-Back-N - בפרוטוקול GBN לשולח מותר לשלוח יותר מחבילה אחת בלי לחכות לאישור. קיימת הגבלה למספר הודעות מקסימלי N ב-pipeline. הטווח של מספרים סידוריים אפשריים לחבילות שנשלחו אבל עדיין לא התקבל עליהן אישור מוגדר כ"חלון" בגודל N . תוך כדי פעולת הפרוטוקול החלון זז קדימה.
- פרוטוקול (SR) Selective Repeat - הפרוטוקול מונע שליחה מחדש מיותרת בגלל שהשולח שולח מחדש רק את החבילות שאבדו או הגיעו למקבל משובשות. לשם כך המקבל צריך לשלוח ACK לכל חבילה שהגיעה תקינה בנפרד. נדרש כאן חלון בגודל N כדי להגביל את מספר החבילות הבלתי מאושרות ב-pipeline.

תצבורה חסרת קשר (connectionless UDP):

- פרוטוקול תעבורה פשוט.
- חבילות UDP יכולות ללכת לאיבוד או להגיע של על פי הסדר.
- Connectionless: ללא לחיצת יד, כל חבילת UDP מטופלת בנפרד. חסכון ביצירת החיבור.
- מדוע צריך את פרוטוקול UDP?
- חסכון התקורה ביצירת החיבור.
- פרוטוקול פשוט – חשוב בשביל מכשירים פרימיטיביים, כמו מכשיר אזעקה.
- בעל header קטן – הודעות קצרות יותר.
- אין בקרת דחיסה ועומס – גורם להעמסת הרשת ואפשרות לניצולת מקסימלית.
- משמש ליישומי מולטימדיה, בהם אפשרי לאבד חבילות ובהם חשוב קצב מהיר של העברה, יש עדיפות למהירות על פני אמינות.

תקשורת מכוונת קשר (Connection Oriented) :TCP

- קשר נקודה לנקודה (point to point). שולח אחד ומקבל אחד. אין אפשרות ל-multicasting: העברת מידע משולח אחד להרבה מקבלים בפעולת שליחה אחת אינה אפשרית.
- הפרוטוקול רץ רק במחשבי הקצה ולא בנתבים בדרך.
- אמינות: הגעת הודעות בסדר הנכון.
- תומך ב-session דו-כיווני, שני הצדדים שולחים ומקבלים מידע (full duplex data) קיים MSS (maximum segment size) שהוא גודל מקסימלי של סגמנט שניתן להעביר.
- מוכוונת קשר (connection oriented): קיימת לחיצת יד (handshake) שמאתחלת את מצבי השולח והמקבל לפני העברת מידע ביניהם.
- בקרת זרימה (flow control): השולח לא ישלח יותר מידע ממה שהנמען יכול לקבל.
- קיימים חוצצים בשני הצדדים.

TCP

□ פרוטוקול TCP- מאורעות בצד השולח:

- כאשר מתקבל מידע מרמת היישום:

נוצר סגמנט עם מס' סידורי שהוא מספרו הסידורי של ה-byte הראשון באותו סגמנט. אתחול ה-timer אם אינו מופעל כבר. (ה-timer הוא עבור ההודעה הישנה ביותר שלא התקבל עליה ACK).

- Timeout: שליחה מחדש של סגמנט שגרם ל- timeout ואתחול ה-timer.

- קבלת ACK: אם מתקבל ACK על חבילות שעדיין לא הגיע להן ACK, אז מתבצע עדכון של הסגמנטים אשר עליהם צריך לעשות ACK.

□ פרוטוקול TCP- מאורעות בצד המקבל:

- מס' סידורי: סגמנט עם מס' סידורי שהוא מספרו הסידורי של ה-byte הראשון באותו סגמנט.

- ACK: מס' סידורי של ה-byte הבא אותו מצפים לקבל מהצד השולח.

- הצד המקבל מתמודד עם סגמנטים שהגיעו לא לפי הסדר בכך שהוא מתעלם מחבילות ישנות, ושם בחוצצים או מתעלם מחבילות עתידיות.

הקרת לרימה ב-TCP:

- המקבל מקצה מקום למידע שהוא אמור לקבל מהשולח – RcvBuffer ומסמן לשולח כמה מקום נותר לו פנוי לאורך זמן החיבור – RcvWindow.
- השולח מגביל את נפח המנות שהוא יכול לשלוח ללא קבלת ACK, נפח זה הוא כנפח ה-RcvWindow. זאת כדי למנוע הצפה של החוצצים של המקבל.
- ה-ACKים שמתקבלים, לא תמיד מגיעים לפי סדר השליחה המקורי.
- סיבות לכך שהחוצץ מלא:
 1. חבילה ישנה לא התקבלה (לא התקבל ACK עליה).
 2. האפליקציה לא קראה עדיין מידע מהחוצצים.

עקרונות בקרת העומס:

- בעומס יתר תורי ההמתנה בנתבים נעשים ארוכים וכך נוצרות השהיות ארוכות. כמו כן, מנות הולכות לאיבוד.
- ההבדל בין בקרת עומס לזרימה הוא שבעומס מתמקדים בחוצצים של הנתבים ובזרימה בחוצצים של מחשבי הקצה.
- אין שליחת אישורים מהנתבים, תפקידם היחיד הוא העברת החבילה ליעד הבא.

TCP מתחיל כאף

- במידה וישנן נתבים ו/או חיבורים איטיים ברשת בין השולח למקבל עלולות להיווצר בעיות כאשר חבילות יאבדו או התעכבו כתוצאה מעומס זמני.
- בכדי לווסת את קצב כתיבת המידע לרשת ע"י השולח כך שכל המידע יגיע למקבל משתמשים, בין השאר באלגוריתם "Slow start".
- אלגוריתם "Slow start" מוסיף עוד חלון ל-TCP של השולח, Congestion Window, הנקרא גם cwnd. כאשר נוצר קשר TCP חדש עם מחשב ברשת אחרת, ה-cwnd מאותחל לגודל של סגמנט אחד.
- בכל פעם שמתקבל ack, ה-cwnd גדל בסגמנט אחד.
- השולח יכול לשדר עד שהוא מגיע למינימום בין ב-cwnd ל-Advertised window.
- ה-cwnd משמש כבקרת זרימה של השולח, בעוד ה-Advertised window (גודל החלון המרבי איתו התחנה המקבלת מוכנה לעבוד) משמש כבקרת זרימה של המקבל.
- השולח מתחיל ע"י שידור של סגמנט אחד והמתנה ל-ack. כאשר ה-ack מתקבל גדל ה-cwnd לגודל של שני סגמנטים ואז השולח משדר שני סגמנטים.
- כאשר מתקבלים ack על הסגמנטים גדל ה-cwnd להיות בגודל 4 סגמנטים וכך הלאה. **התנהגות זאת מתארת גידול אקספוננציאלי של גודל חלון ה-cwnd.**

תאונה - Timeout - 3dupACKs

רעיון

• הרעיון מאחורי האלגוריתם הוא ההבדלה בין שני המאורעות שנגרמות עקב עומס ברשת.

• האלגוריתם מבדיל בין חומרתן של שתי המאורעות. שלוש ACK זהים רצופים מעידים על עומס גדול אך חמורים פחות מ-timeout. לכן, כאשר מקבלים שלוש ACK מפחיתים לחצי את ה-CongWin וכאשר מקבלים timeout מפחיתים למינימום האפשרי.

• אחרי שמתרחש timeout וגודל ה-CongWin קטן למינימום, ה-CongWin מתחיל לגדול אקספוננציאלית עד שהוא מגיע לחצי גודל ה-CongWin שבו התרחש ה-timeout (נקודת הסף) ואז הוא גדל לינארית.

□ אחרי 3 ACK זהים:

■ CongWin קטן בחצי
(multiplicative decrease)

■ החלון לך גדל באופן ליניארי
(additive increase)

□ אחרי Timeout:

■ התחל מחדש חיבור.

■ CongWin נקבע ל-1.MSS.

■ גידול מעריכי עד לערך סף, ואח"כ גידול ליניארי.

תרגילים לדוגמא



תרגיל 1 (תוספת לתירגול 3)

ברשת slotted aloha ישנן בדיוק שתי תחנות שרוצות לשדר באותו חריץ (slot). הרשת משתמשת ב exponential backoff – ז"א, תחנה המזהה התנגשות בוחרת באקראי אחד מתוך שני slot-ים הבאים לשדר, ואם עדיין מזהה התנגשות, בוחרת באקראי אחד מתוך ארבעת ה slot-ים הבאים לשדר, וכך הלאה, עד לשידור מוצלח. מהי ההסתברות לשתי התנגשויות בדיוק, לפני שידור מוצלח (בהנחה שעד סיום השידור לא תהיה עוד תחנה שתרצה לשדר)?

פתרון תרגיל 1

לאחר שארעה ההתנגשות הראשונה כל אחת מהתחנות בוחרת בין 2 slots:

$K = \{0, 1\}$ ולכן ההסתברות להתנגשות נוספת היא:

$$\square \text{ או ששניהם בחרו } 0: p(0) \cdot p(0) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

$$\square \text{ או ששניהם בחרו } 1: p(1) \cdot p(1) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

$$\square \text{ כלומר ההסתברות להתנגשות תהיה: } \frac{1}{4} + \frac{1}{4} = \frac{2}{4} = \frac{1}{2}$$

\square לאחר מכן אנו רוצים להבטיח שלא תהיה התנגשות (כי בשאלה נדרש לבדוק את ההסתברות ל-2 התנגשויות בדיוק).

פתרון תרגיל 1

□ ולכן, לאחר ההתנגשות השנייה כל תחנה צריכה לבחור בין 4 slots:
 $K = \{0, 1, 2, 3\}$ ולכן ההסתברות להתנגשות נוספת היא חיבור ההסתברויות
של המקרים שבהם שתי התחנות בחרו באותו slot:

$$p(0) \cdot p(0) + p(1) \cdot p(1) + p(2) \cdot p(2) + p(3) \cdot p(3) = \frac{1}{4} \cdot \frac{1}{4} + \frac{1}{4} \cdot \frac{1}{4} + \frac{1}{4} \cdot \frac{1}{4} + \frac{1}{4} \cdot \frac{1}{4} = 4 \cdot \frac{1}{16} = \frac{1}{4}$$

□ מכיוון שאנו רוצים להבטיח שלא תהיה כאן התנגשות ניקח את
ההסתברות למקרה המשלים:

$$1 - \frac{1}{4} = \frac{3}{4}$$

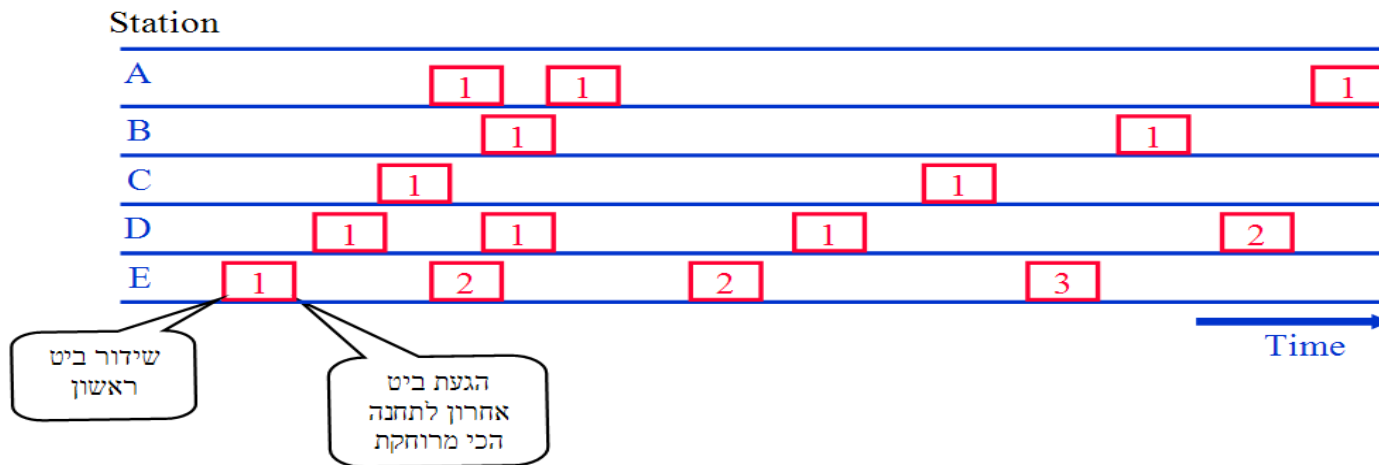
□ ולכן בסה"כ ההסתברות ל-2 התנגשויות בדיוק היא:

$$\frac{1}{2} \cdot \frac{3}{4} = \frac{3}{8} = 0.375$$

תרגיל 2 (תוספת אתיראל 3)

□ לפניך איור, הלקוח מההרצאה, המתאר חבילות הנשלחות ע"י חמש תחנות פעילות בפרוטוקול Aloha.

(1) סמן ב-X על גבי האיור את החבילות בהן תתרחש התנגשות. מה עושים במקרה של התנגשות?

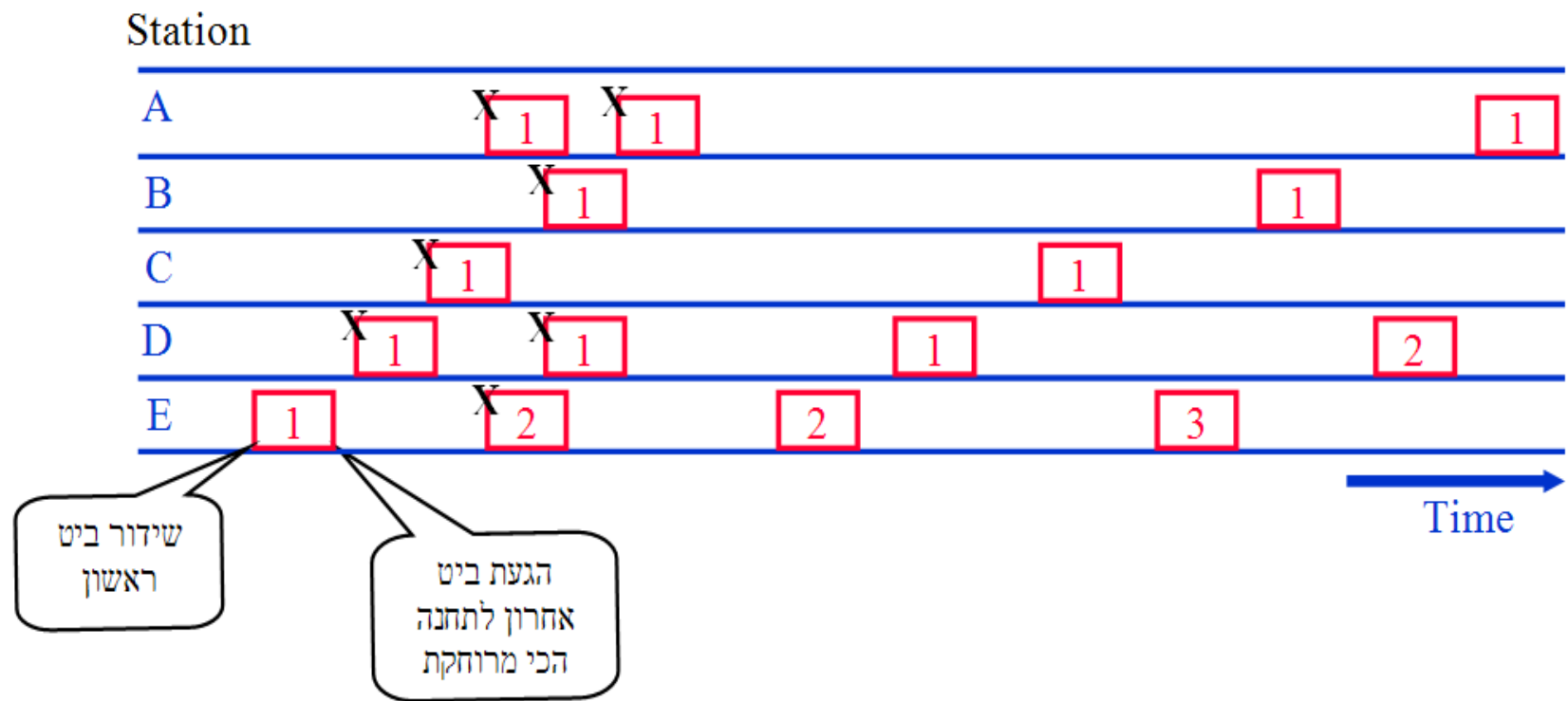


(2) בפרוטוקול Slotted Aloha אין התנגשויות משום שמחלקים את ציר הזמן לחרצים כך שלכל תחנה מותר לשדר רק בתחילת חריץ זמן. **בחר:** נכון / לא נכון **ונמק.**

(3) פרוטוקול CSMA/CD מהווה שיפור לעומת Aloha ו-Slotted Aloha משום שהתחנה מאזינה לערוץ לפני שידור וגם בזמן שהיא משדרת על מנת לאתר התנגשויות. **בחר:** נכון / לא נכון **ונמק.**

פתרון תרשיף 2

□ סעיף 1:



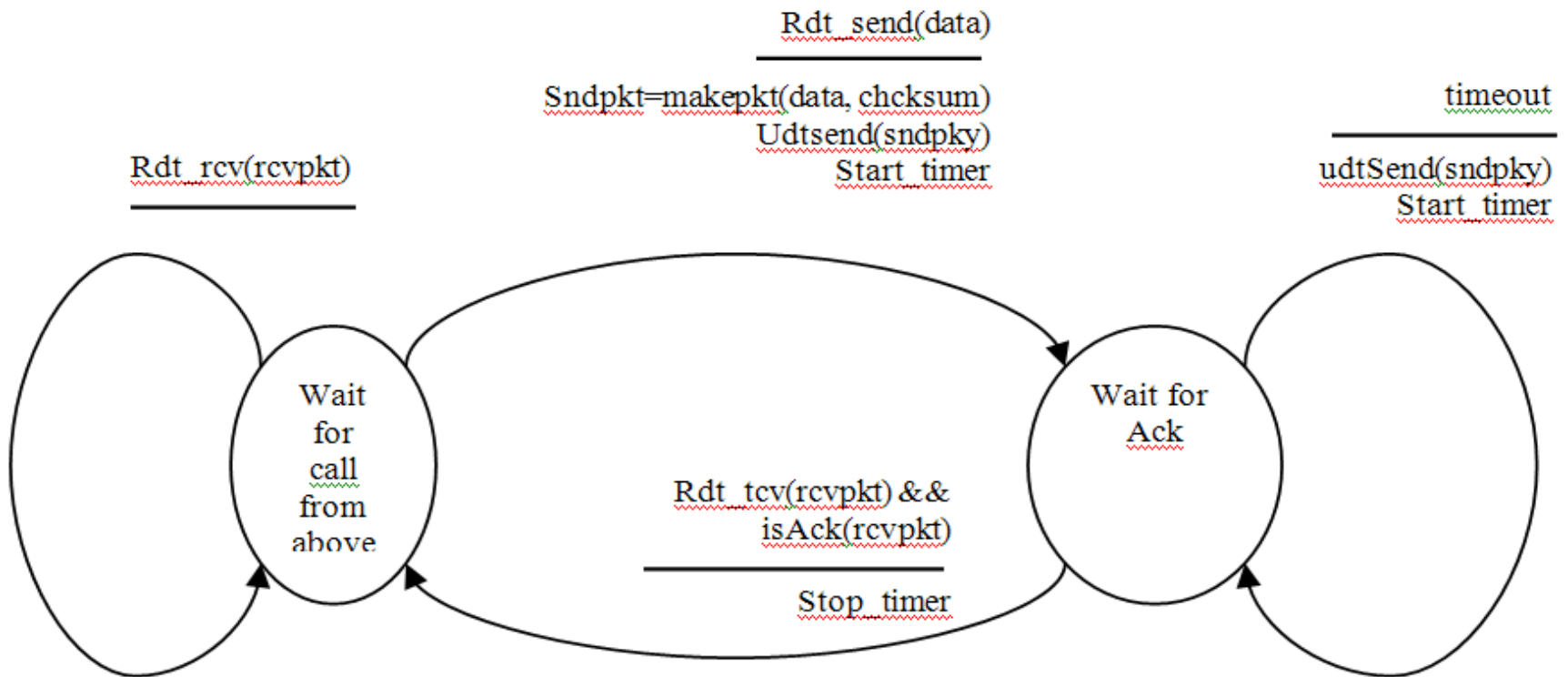
פתרון תרשיף 2

□ סעיף 2': לא נכון. ב-Slotted Aloha עדיין יתכנו התנגשויות, הרעיון הוא פשוט לחלק לחריצים על מנת להגדיר זמני שידור (ולכן גם התנגשויות) או פרק זמן בו חבילות בערוץ וכך להגדיל יעילות.

□ סעיף 3': נכון. ב-CSMA/CD התחנה מאזינה לערוץ לפני שידור וגם בזמן שידור וכך מאותרות התנגשויות בעת שהן קורות ואין בזבוז זמן. לכן גם יש שיפור ביעילות.

תרגיל 3

□ להלן דיאגרמת מצבים לשולח בפרוטוקול פשוט לשידור אמין:



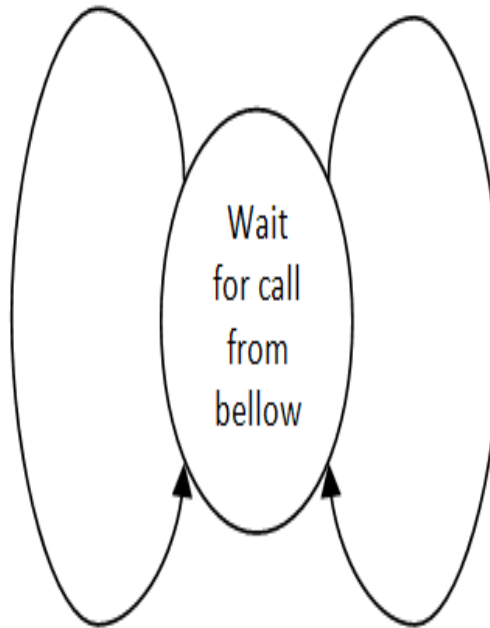
תרגיל 3 (המשק)

- א- הצג דיאגרמת מצבים למקבל.
- ב- באילו הנחות, אם בכלל, מבטיח הפרוטוקול אמינות? (הנחות לדוגמא: שכפול חבילות, הגעת חבילות לפי סדר, השהיה מירבית בערוץ, שיבוש חבילות, אבדן חבילות...)
- ג- הצג תרשים מאורעות שמדגים שהפרוטוקול לא מבטיח אמינות בשימוש מעל פרוטוקול IP.

פתרון תרסיף 3

□ סעיף א':

```
rdt_rcv(rcvpkt)
  && notcorrupt(rcvpkt, chksum)
  rcvpkt = extract(data)
  deliver_data(data)
  sndpkt = makepkt(ACK, chksum)
  udt_send(sndpkt)
```



```
rdt_rcv(rcvpkt)
  && corrupt(rcvpkt, chksum)
```


פתרון תרשיף 3

□ סעיף ב': הפרוטוקול מבטיח אמינות בהנחות הבאות:

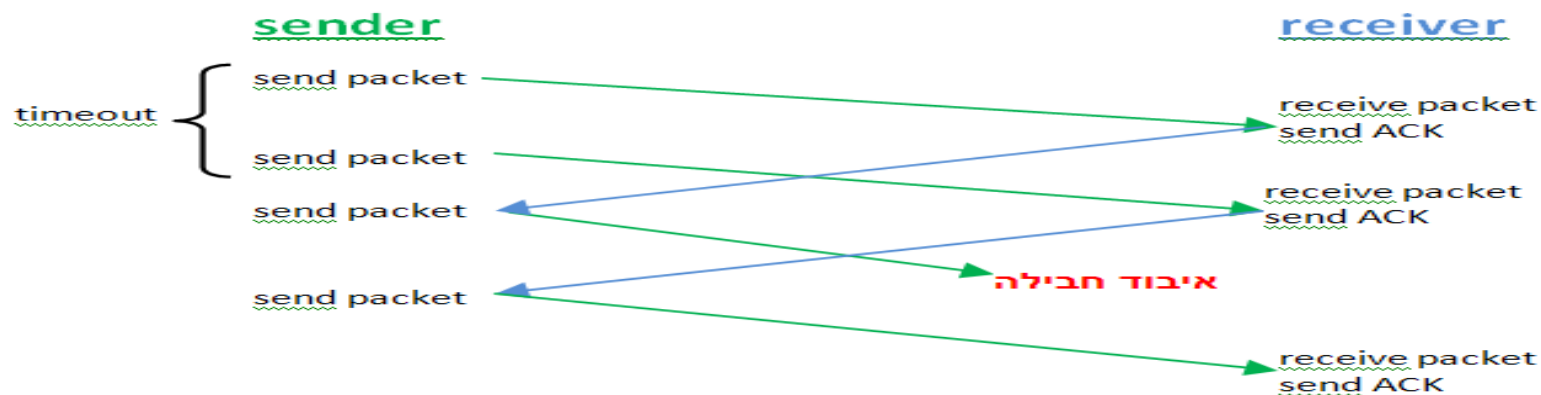
1. אין שיכפול חבילות. הפרוטוקול לא מאפשר לבדוק האם החבילות שהתקבלו הן שיכפול אחת של השנייה. ולכן אם יהיה שיכפול, המקבל ישלח ACK על החבילה עצמה, ו-ACK נוסף על השיכפול. לאחר שהשולח יקבל את ה-ACK הראשון הוא ישלח חבילה הבאה, ואז יגיע ACK על השיכפול של החבילה הקודמת, והשולח יבין שהמקבל קיבל את החבילה החדשה, למרות שהחבילה הייתה יכולה להיאבד בדרך.
2. יש השהיה מירבית בערוץ, שלפיה יקבע ה-timeout של השולח. אחרת אם לא תהיה ההשהיה מירבית יוכל להיווצר שיכפול חבילות – ה-timeout יסתיים והחבילה תשלח שוב, למרות שהחבילה המקורית הגיע, רק שלקח לה יותר זמן להגיע. ולכן נקבל ACK מיותר וכך יוכל להיווצר גם איבוד חבילה (כמו בשיכפול חבילות).
3. כל השגיאות מתגלות – אחרת המקבל ישלח לאפליקציה מידע שגוי; או שהשולח יקבל ACK משובש שלא יזהה אותו כ-ACK, אחרי timeout ישלח את החבילה שוב, וייווצר שיכפול.
4. אין אובדן של הודעות ACK – אחרת השולח יחכה ל-timeout, וישלח את החבילה שוב. ואז המקבל יקבל שיכפול של החבילה שכבר קיבל, ולא יוכל לזהות זאת.

פתרון תרסיף 3

סעיף ג':

IP שייך לשכבת ה-Network ואינו מבטיח אמינות, מבטיח רק שה-header יגיע תקין ע"י checksum. כלומר יכול לקרות שיכפול הודעות, שגיאות בחבילות, איבוד חבילות ולא ניתן לדעת את גודל ההשהיה המקסימלית.

תרשים מאורעות שמדגים שהפרוטוקול לא מבטיח אמינות בשימוש מעל פרוטוקול IP:



בתרשים אפשר לראות שאחרי timeout ה-sender שלח שוב את החבילה הראשונה, אבל ה-receiver קיבל פעמיים את החבילה הראשונה (בשליחה הראשונה התעכבה), ושלח שני ACKs, אחרי ה-ACK הראשון ה-sender שולח את החבילה השנייה ואחרי ה-ACK השני (על שיכפול החבילה הראשונה) שולח את החבילה השלישית, למרות שהחבילה השנייה נאבדה.

תרבי 4

- נתייחס למאורע זיהוי אובדן חבילה בפרוטוקול TCP, ונסמן ב-ח את מספר הסגמנט הכי מוקדם שעדיין לא התקבל עליו אישור (Ack). הרבה מימושי TCP שולחים מחדש רק את סגמנט ח, אף כי הסטנדרט מרשה גם לשלוח מחדש גם סגמנטים שנשלחו אחרי ח (וגם עליהם לא התקבל Ack); סגמנטים אלו נשלחים כאשר מתקבל ה-Ack על סגמנט ח. התייחס למצבים הבאים, כאשר תמיד המידע זורם רק מהשולח לנמען (שרק שולח אישורים); בכל מצב, תן דוגמאות למקרים בהם עדיפה כל אחת משתי השיטות – בחר דוגמאות פשוטות, ברורות וטיפוסיות ככל האפשר; אם אין דוגמא בה שיטה מסוימת עדיפה, הסבר מדוע.
- (a) השולח והנמען מחוברים דרך ערוץ אינפרא-אדום ייעודי (ללא שותפים).
- I. תן דוגמא למקרה בו עדיף לשלוח מחדש את כל הסגמנטים ולא רק את ח, או נמק בקצרה מדוע אין לעולם מקרה כזה.
- II. תן דוגמא למקרה בו עדיף לשלוח מחדש רק את סגמנט ח, או נמק בקצרה מדוע אין לעולם מקרה כזה.
- III. מה המימוש העדיף לדעתך למצב זה (נמק בקצרה)?

תוספת לתרגיל 4

□ נאזין לקליפ

תרגיל 4 (המשק)

(b) תחנת הקרקע (ground control) משדרת נתוני חשופים בזמן אמת למעבורת חלל (major Tom), הנמצאת בגובה 35,000 ק"מ בקצב 100Mbps. הנח שהפרעה אופיינית בשידור כזה נמשכת כעשירית שניה.

I. תן דוגמא למקרה בו עדיף לשלוח מחדש את כל הסגמנטים ולא רק את ח, או נמק בקצרה מדוע אין לעולם מקרה כזה.

II. תן דוגמא למקרה בו עדיף לשלוח מחדש רק את סגמנט ח, או נמק בקצרה מדוע אין לעולם מקרה כזה.

III. מה המימוש העדיף לדעתך למצב זה (נמק בקצרה)?

(c) השולח והנמען מחוברים דרך האינטרנט.

I. תן דוגמא למקרה בו עדיף לשלוח מחדש את כל הסגמנטים ולא רק את ח, או נמק בקצרה מדוע אין לעולם מקרה כזה.

II. תן דוגמא למקרה בו עדיף לשלוח מחדש רק את סגמנט ח, או נמק בקצרה מדוע אין לעולם מקרה כזה.

III. מה המימוש העדיף לדעתך למצב זה (נמק בקצרה)?

פתרון תרגיל 4

□ סעיף a:

- I. כשהשולח נמצא במרחק מקסימלי, שמאפשרת החומרה, מהמקבל, יכולים לקרות הרבה איבודי סגמנטים. לכן אחרי איבוד סגמנט n עדיף ישר אחרי שליחה מחדש של הסגמט וקבלת ה-ACK עליו, לשלוח את הסגמנטים הבאים מחדש, כי בהסתברות גבוהה עוד סגמנטים נאבדו.
- II. במרחק סביר (לפי החומרה) תהיה תקשורת טובה בערוץ תקשורת זה בין השולח למקבל, אבל בכל זאת יכולים מדי פעם להיאבד סגמנטים. מכיוון שאיבוד סגמנטים במקרה כזה לא יהיה תדיר אפשר לשלוח מחדש רק את הסגמנט n .
- III. עדיף לשלוח רק את סגמנט n מכיוון שברוב המקרים המרחק בין השולח למקבל יהיה סביר, והסיכוי שנוכל למקם את השולח והמקבל במרחק המקסימלי שמאפשרת החומרה מבלי שהקשר יתנתק לגמרי הוא קטן.

פתרון תרסיף 4

□ סעיף ב:

I. על השידור למעבורת החלל יכולים להשפיע שידורים ממקורות אחרים ותופעות טבע שונות, לכן זהו ערוץ תקשורת לא אמין, גם RTT בערוץ תקשורת זה ארוך ולכן עדיף לשלוח מחדש את הסגמנטים שנשלחו מיד אחרי כי כנראה גם הם נאבדו בהפרעה.

II. אם לשולח (ground control) יש מעבד חלש שאחראי גם על שליחת המידע, אז כדי לא להעמיס עליו עדיף לשלוח רק את הסגמנט ה-n שבוודאות לא הגיע.

III. עדיף לשלוח מחדש את כל הסגמנטים החל מ-n כי ברוב המקרים לשולח (ground control) יהיה מספיק כוח עיבוד כדי להקצות משאבים ולשלוח את הסגמנטים שוב; וגם בגלל ההפרעות הארוכות יחסית יש הסתברות גבוהה שיותר מסגמנט אחד נאבד/נפגם; ובגלל ה-RTT הגבוה, עדיף לא לחכות להודעה מהמקבל על כל סגמנט שנפגם, אלא לשלוח את כל הסגמנטים שנשלחו אחרי n מחדש (לאחר שהתקבל ACK על סגמנט n).

פתרון תרסיף 4

□ סעיף C:

- I. אם תשתית האינטרנט בין השולח למקבל מאפשרת תקשורת רק עם RTT גבוה, אז עדיף כמה שפחות לחכות לתגובות של המקבל, ולשלוח מחדש את כל הסגמנטים
- II. אם תשתית האינטרנט טובה, אז חוץ מה-RTT הנמוך, ההסתברות לשיבוש/איבוד סגמנטים תהיה קטנה, ולכן אם יקרה שהסגמנט נאבד, עדיף לשלוח רק אותו.
- III. ברוב המקרים היום על תשתית האינטרנט יש RTT נמוך, והסגמנטים משתבשים/נאבדים לעתים רחוקות, לכן אם נאבד סגמנט, עדיף לשלוח רק אותו.

תרגיל 5

ענה לכל סעיף בנכון/לא נכון? והסבר את תשובתך.

1. גודל חלון ה- Advertised Window ב- TCP אינו משתנה במשך ה- connection.
2. צומת A שולח לצומת B קובץ גדול מעל TCP. בזמן נתון תוך כדי השליחה מספר הבתים שלא נעשה עליהם ACK לא עולה על גודל ה- Receive buffer.
3. צומת A שולח לצומת B קובץ גדול מעל TCP. אם ה- sequence number של סגמנט הוא m , אזי ה- sequence number של הסגמנט הבא הוא $m+1$.
4. צומת A שולח לצומת B שני סגמנטים של TCP, כשבראשון ה- sequence number הוא 90 ובשני הוא 110. נניח שהראשון אבד. מה יהיה ה- ACK sequence number ש- B ישלח?
5. הצומת השולח מחשב את הזמן שעובר משליחת חבילה, ורק כאשר הזמן עובר סף מסויים הוא יודע שחבילה לא הגיע ליעד.

פתרון תרגיל 5

□ גודל חלון ה- Advertised Window ב- TCP אינו משתנה במשך ה- connection.

תשובה: לא נכון - החלון משתנה לפי העומס.

□ צומת A שולח לצומת B קובץ גדול מעל TCP. בזמן נתון תוך כדי השליחה מספר הבתים שלא נעשה עליהם ACK לא עולה על גודל ה- Receive buffer.

תשובה: נכון - מספר הבתים שלא נעשה עליהם ACK תמיד קטן או שווה לגודל ה- buffer אצל המקבל.

□ צומת A שולח לצומת B קובץ גדול מעל TCP. אם ה- sequence number של סגמנט הוא m , אזי ה- sequence number של הסגמנט הבא הוא בהכרח $m+1$.

תשובה: לא נכון - המספר הסידורי הינו m ומספר הבתים בסגמנט $(m+MSS)$ או $(m+n)$ ולא בהכרח $m+1$

פתרון תרגיל 5

□ צומת A שולח לצומת B שני סגמנטים של TCP, כשבראשון ה- sequence number הוא 90 ובשני הוא 110. נניח שהראשון אבד. מה יהיה ה- ACK sequence number ש-B ישלח?

תשובה: **90**

□ הצומת השולח מחשב את הזמן שעובר משליחת חבילה, ורק כאשר הזמן עובר סף מסויים הוא יודע שחבילה לא הגיע ליעד.

תשובה: לא נכון - גם קבלת duplicated ACK מציין איבוד חבילה

תרגיל 6

לכל אחת מהטענות הבאות, רשום **נכון** / **לא נכון** והסבר בקצרה.

- (1) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv4, השרתים מבצעים checksum בשכבת הרשת. **נכון** / **לא נכון**
- (2) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv4, השרתים מבצעים checksum בשכבת ה-transport. **נכון** / **לא נכון**
- (3) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv4, הנתבים בהם עוברת הודעה מבצעים checksum בשכבת הרשת. **נכון** / **לא נכון**
- (4) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv4, הנתבים בהם עוברת הודעה מבצעים checksum בשכבת ה-transport. **נכון** / **לא נכון**

תרכיף 6 (המשק)

(5) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv6, השרתים מבצעים checksum בשכבת הרשת. **נכון / לא נכון**

(6) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv6, הנתבים בהם עוברת הודעה מבצעים checksum בשכבת הרשת. **נכון / לא נכון**

(7) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv6, הנתבים בהם עוברת הודעה מבצעים checksum בשכבת ה-transport. **נכון / לא נכון**

(8) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv6, השרתים מבצעים checksum בשכבת ה-transport. **נכון / לא נכון**

פתרון תרגיל 6

(1) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv4, השרתים מבצעים checksum בשכבת הרשת.

נכון

(2) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv4, השרתים מבצעים checksum בשכבת ה-transport.

נכון

(3) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv4, הנתבים בהם עוברת הודעה מבצעים checksum בשכבת הרשת.

נכון

הסבר - Checksum: מציאת שגיאות פשוטות בלבד ללא מנגנון לתיקון שגיאות. השולח: מחלק את הסגמנט לקטעים של 16 ביט. מחבר אותם ומבצע השלמה ל 1-, ואת התוצאה שם בשדה ה-checksum. המקבל: מחשב את ה-checksum של הסגמנט שקיבל ובודק אם המחושב שווה לזה שהתקבל. אם הוא שונה התגלתה טעות, ואם זהה לא התגלתה טעות אבל ייתכן שהייתה טעות.

פתרון תרגיל 6 (המשק)

- בפרוטוקול ה-IP אחד השדות הוא שדה סכום הביקורת (checksum) שבו שדה בדיקה לווידוא תקינות הכותרת (header) בלבד.
- לכן כל מי מתעסק בשליחת/קבלת מידע ובניתוב "שכבת הרשת" ו-"שכבת ההובלה" (שרתים, מחשבי קצה ונתבים) חייב לדעת שהמידע שהוא מקבל אמין ולכן הבדיקה.
- (4) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv4, הנתבים בהם עוברת הודעה מבצעים checksum בשכבת ה-transport.
- לא נכון
- הסבר – בנתבים קיימות רק 3 השכבות הראשונות: הפיסית, קישור הנתונים והרשת, ושכבת התעבורה אינה קיימת כלל, ומכאן שברור שהתשובה היא לא נכון.
- (5) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv6, השרתים מבצעים checksum בשכבת הרשת.
- לא נכון
- (6) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv6, הנתבים בהם עוברת הודעה מבצעים checksum בשכבת הרשת. נכון / לא נכון

פתרון תרצי"א 6 (המשק)

(7) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv6, הנתבים בהם עוברת הודעה מבצעים checksum בשכבת ה-transport. **נכון / לא נכון**
לא נכון

הסבר - IPv6 עבר שינויים מ-IPv4: "Checksum" הסרת תכונה זו לחלוטין כדי שנוכל להוריד את זמן התהליך בכל hop ("ניתור" נקודת חיבור בין מחשבים ברשת). לכן אין מבצעים checksum בשכבת הרשת בנתבים ובשרתים. כמו שכבר אמרנו שכבת ה-transport לא קיימת בנתבים.

(8) במהלך שימוש ב-TCP/IP כאשר עובדים ב-IPv6, השרתים מבצעים checksum בשכבת ה-transport. **נכון / לא נכון**
נכון

הסבר - פרוטוקול IPv6 מבצע checksum בשכבת ה-transport עבור המידע.

תרגיל 7

בפרוטוקול Go Back N ישנו מנגנון שבו שולחים כמה חבילות בבת אחת (Pipelining). בהנחה שגודל החלון הוא n , קצב השידור הוא 100Mbps, גודל סגמנט מקסימלי הוא 2.5KB וה-RTT הוא 10ms:

- (1) ללא Pipelining (גודל חלון $n=1$) מהו אחוז הזמן שבו המחשב השולח עסוק בלשלוח את החבילות (Utilization)?
- (2) חזור על א' עבור $n = 30$?
- (3) חזור על א' עבור $n = 60$?

פתרון תרגיל 7

(1) ללא Pipelining (גודל חלון $n=1$) מהו אחוז הזמן שבו המחשב השולח עסוק בלשלוח את החבילות (Utilization)?
נתון:

$$RTT = 10\text{ms}$$

$$n = 1$$

$$L = 2.5\text{KB}$$

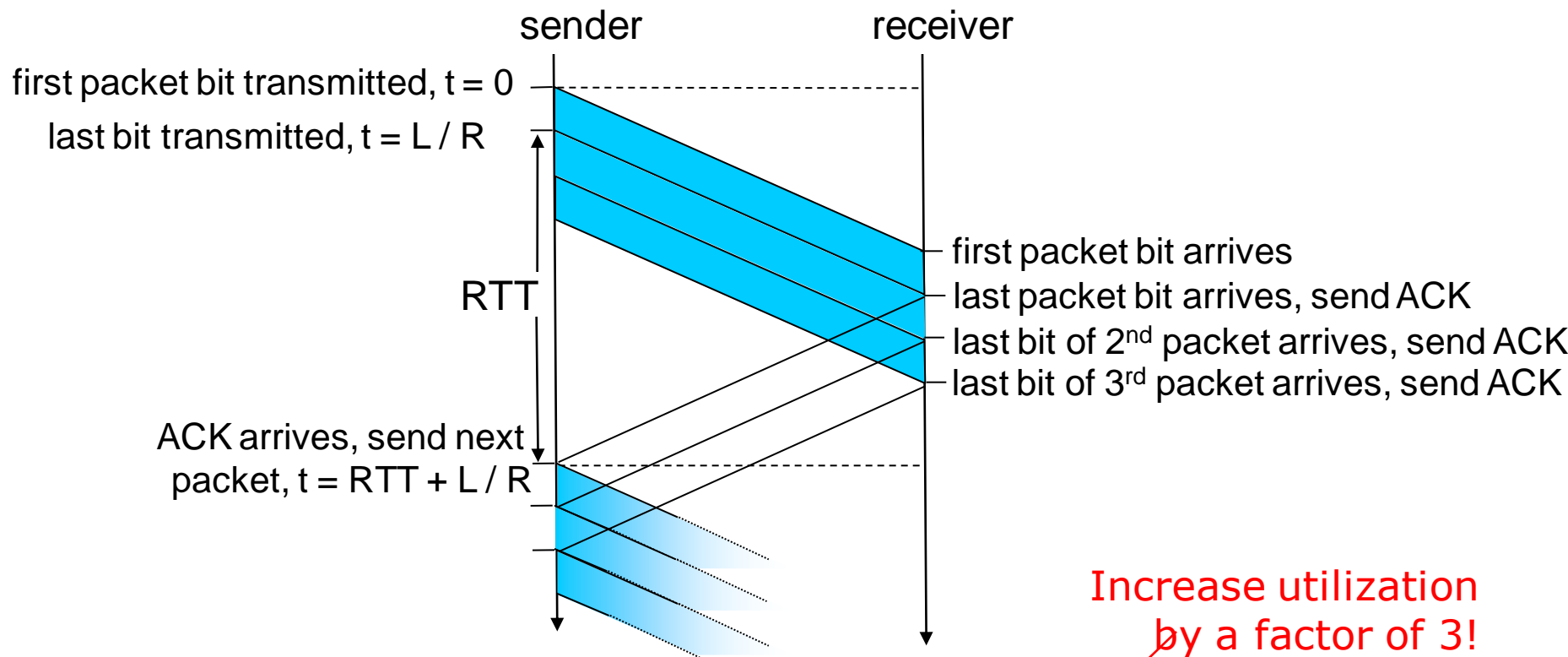
$$R = 100\text{Mbps}$$

$$(L/R) / (RTT + (L/R)) = (2.5 * 8 / 100) / (10 + (2.5 * 8 / 100))$$

$$\approx 1.96\% \text{ utilization.}$$

האדפאת הניצולת Pipelining

נשתמש בנתונים של החישוב הקודם כדי להראות כיצד pipeline מגדיל את הניצולת, במקרה זה פי 3 כאשר שולחים 3 הודעות במקביל:



$$U_{\text{sender}} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

פתרון תרגיל 7

(2) עבור $n = 30$.

$$L/R) / (RTT + (L/R)) = ((2.5 * 8 / 100) * 30) / (10 + (2.5 * 8 / 100))$$

$\sim = 58.8\%$ utilization

(3) עבור $n = 60$.

$$L/R) / (RTT + (L/R)) = ((2.5 * 8 / 100) * 60) / (10 + (2.5 * 8 / 100))$$

$\sim = 100\%$ utilization