

תרגול מס' 4 – שכבת התעבורה



רְוַיֵּאַמְּסֶעָנָזְרָן | גִּבְכָּתְּ הַכְּלָתָן:

- עקרונות של בקרת עומסים.
- בקרת עומסים TCP.
- שירותי שכבת ההפלה.
- ריבוב/פילוג.
- תעבורת חסרת קשר: UDP
- עקרונות תעבורת מידע אמינה ברשות.
- תעבורת מונחת קשר: TCP
 - מבנה המקטעה.
 - תעבורת מידע אמינה.
 - בקרת זרימה.
 - ניהול הקשר

תזכורת

❑ נושאי מפתח:

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

❑ שירותי תעבורה ופרוטוקולים

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

תזכורת

□ פרוטוקולי תעבורת אינטרנט:

- שירות תעבורת אמין TCP - שירות שבו הودעות מועברות ומתקבלות בסדר הנכון מבלי לכת לאיבוד.

■ מאפיינים:

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

□ שירות תעבורת לא אמין UDP - **Best effort**: יתכן איבוד הودעות, 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 בישומי זמן אמיתי, כמו טלפון, נדרש לתרmor בשירותים האלו.

ק'ק אגציימט הגדחת נאירומט זכק אכוּפֿוּ כתט ה'רמ אג'ימט נאירומט?

- Rdt1.0: פרוטוקול פשוט שמניח שהערך שימושים בו אמין.
- Rdt2.0: ערך תקשורת לא אמין.
 - הנחות:
 - אין אובדן חבילות. (יתכן שייגיעו משבשות)
 - החבילות מגיעות בסדר הנכון.
 - כל הטעויות מתגלות על ידי מנגןון ה-.checksum
 - קיימן היוזן חוזר מהצד מקבל לצד השולח:
 - אישור על קבלת חבילה תקינה (acknowledgment) **ACK**
 - אישור על קבלת הودעה משובשת (negative acknowledgment) **NAK**
 - חבילה שהגיעה משובשת תשלח מחדש (retransmission).
 - תוספות שלא היו קיימות ב-Rdt1.0:
 - גילוי טעויות.
 - היוזן חוזר מהצד מקבל לשולח (ACK, NAK)

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

- הנחות:
 - אין אובדן מידע.
 - אין שינוי בסדר החבילות.
 - תוספות לפרוטוקול: מונה של בית אחד.

□ rdt2.2: טיפול בשכפול הודעות:

- שיפור לעומת rdt2.1: מונע שכפול חבילות ACK.
- אם יש שכפול של הودעת ACK אז השולח ישלח מחדש את החבילה הנוכחית ולא יהיה איבוד מידע.
- אין טיפול באובדן חבילות, יתכן מצב בו מקבל והשלח יחכו זמן אינסופי.

□ rdt3.0: פרוטוקול הבית המתחלף:

- שיפור לעומת rdt2.1: התמודדות עם אובדן חבילות.
- **Timer**: השולח מחקה ל-ACK פרק זמן "סביר" אשר קטן מהעיכוב המקסימלי (RTT). במידה ולא התקבל ACK בזמן זהה תבצע שליחה מחדש.

.PIPELINE

- בפרוטוקולים מסוג stop and wait נוצרה בעיה של ניצולת נמוכה, ניתן לשЛОח הודעה אחת בכל פעם בלבד.
- פרוטוקול (GBN) N-Go-Back - בפרוטוקול N GBN לשЛОח מותר לשЛОח יותר מabitלה אחת בלי לחייב אישור. קיימת הגבלה למספר הודעות מקסימלי N ב-pipeline. הטווח של מספרים סידוריים אפשריים לחבילות שנשלחו אבל עדין לא התקבל עליהם אישור מוגדר כ"חלון" בגודל N. תוך כדי פעולה הפרוטוקול החלון זו קדימה.
- פרוטוקול (SR) Selective Repeat - הfrוטוקול מונע שליחה מחדש מיותרת בגלל שהשלוח שולח מחדש רק את החבילות שאבדו או הגיעו למקבל משובשות. לשם כך מקבל צריך לשLOWח ACK לכל חבילה שהגיעה תקינה בנפרד. נדרש כאןחלון בגודל N כדי להגביל את מספר החבילות הבלתי מאושרות ב-pipeline.

תאוצה מסכת קעכ (connectionless UDP)

- פרוטוקול תעבורה פשוט.
- חבילות UDP יכולות ללקת לאיבוד או להגעה של על פי הסדר.
- Connectionless: ללא לחיצת יד, כל חבילה UDP מטופלת בנפרד. חסכו ביצירת החיבור.

- מדוע צריך את פרוטוקול UDP?
- חסכו התקורה ביצירת החיבור.
- פרוטוקול פשוט – חשוב בשבייל מכשירים פרימיטיביים, כמו מכשיר אזעקה.
- בעל header קטן – הודעות קצרות יותר.
- אין בקרת דחיסה ועומס – גורם להעמסת הרשת ואפשרות לניצול מקסימלית.
- משמש לישומי מולטימדיה, בהם אפשרי לאבד חבילות וביהם חשוב קצב מהירות של העברה, יש עדיפות למהירות על פני אמינות.

תקורת אוכוֹרֶטֶת קָעַט (Connection Oriented) :TCP

- קשר נקודת לנקודת point (point to point). שולח אחד ומקבל אחד. אין אפשרות ל- multicasting: העברת מידע משולח אחד להרבה מקבלים בפעולה שליחה אחת אינה אפשרית.
- ה프וטוקול רץ רק במחשי הרצה ולא בנתבים בדרך.
- אמינות: הגעת הודעות בסדר הנכון.
- תומך ב-session דו-כיווני, שני הצדדים שולחים ומקבלים מידע full.
- מקסימלי של סegment שנייה להעברה.
- מוכוונת קשר (connection oriented): קיימת לחיצת יד (handshake) שמאתחלת את מצבו של השולח והמקבל לפני העברת מידע ביניהם.
- בקרת זרימה (flow control): השולח לא ישלח יותר מידע ממנו שהנמען יכול לקבל.
- קיימים חוצצים בשני הצדדים.

TCP

□ פרוטוקול TCP - מאורעות מצד השולח:

- כאשר מתקבל מידע מרמת היישום:
נוצר סגמנט עם מס' סידורי שהוא מספרו הסידורי של ה-byte הראשון באותו סגמנט. את חול ה-timer אם אין מופעל כבר. (ה-timer הוא עבר ההודעה הישנה ביותר שלא התקבל עליה ACK).
- Timeout: שליחה מחדש של סגמנט שגרם לו timeout ואת חול 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 מצד המקלט:

- 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 המקלט מתמודד עם סגמנטים שהגיעו לא לפי הסדר בכר שהוא מתעלם מחבילות ישנות, ושם בחוצצים או מתעלם מחבילות עתידיות.

נקמת לכיאת g-TCP:

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

אקלוראות מקכת האזנס:

- בעומס יתר תורי המתנה בנתבים נעים ארוכים וכך נוצרות השהיות ארוכות. כמו כן, מנוגת 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 החבילות לעד הבא.

TCP אמצעי ok

- ה-pwd מושך כברחת זרימה של השולח, בעוד ה-wdwindow Advertised window (גודל החלון המרבי אותו התחנה מקבלת מוכנה לעבוד) מושך כברחת זרימה של מקבל.
- השולח מתחילה ע"י שידור של סגמנט אחד והמתנה ל-ack. כאשר ה-ack מתקבל גודל ה-pwd הולך לגודל של שני סגמנטים ואז השולח משדר שני סגמנטים.
- כאשר מקבלים ack על הסגמנטים גודל ה-pwd להיות בגודל 4 סגמנטים וכך הלאה. **התנאיות זאת מתארת גידול אקספוננציאלי של גודל חלון ה-pwd.**
- במידה ישן נתבים ו/או חיבורים איטיים ברשת בין השולח למקבל עלולות להיווצר בעיות כאשר חבילות יאבדו או התעכבו כתוצאה מעומס זמן.
- ב כדי לווסת את קצב כתיבת המידע לרשת ע"י השולח כך שכל המידע יוכל למקבל משתמשים, בין השאר באlgoritam "Slow start".
- אלגוריתם "Slow start" מוסיף עוד חלון TCP של השולח, הנקרא גם pwdwindow. כאשר נוצר קשר TCP חדש עם מחשב ברשת אחרת, pwdwindow מוחלט לגודל של סגמנט אחד.
- בכל פעם שמקבל ack, ה-pwd גדל בסגמנט אחד.
- השולח יכול לשדר עד שהוא מגיע למינימום pwdwindow בין ב-pwd ל-wdwindow Advertised window.

מבחן \leftarrow 3dupACKs - \leftarrow TimeOut

רעיון

- הweeney מתחזק האלגוריתם הוא הבדלה בין שני המאורעות שנגרמו עקב עומס בראשת.
- האלגוריתם מבידיל בין חומרתן של שתי המאורעות. שלוש ACK זרים 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 שתרחש *timeout* וגודל ה-CongWin קטן למינימום, ה-CongWin מתחילה לגדול אקספוננציאלית עד שהוא מגיע לחצי גודל ה-CongWin שבו התרחש ה-*timeout* (נקודת הסוף) ואז הוא גדל ליניארית.

אחרי 3 ACK זרים:

- CongWin קטן בחצי (multiplicative decrease)
- החולן لكن גדול באופן ליניארי (additive increase)

אחרי Timeout:

- התחל מחדש חיבור.
- CongWin נקבע ל-1 MSS.
- גדיל מעריכי עד לערך סף, ואח"כ גידול ליניארי.

תרגולים לדוגמה



תכליך ו(תוספת מתיכליך 3)

ברשת aloha slotted ישן בדיק שתי תחנות שרותות לשדר באותו חריז (slot). הרשת משתמשת ב exponential backoff – ג"א, תחנה המזאה התנשאות בוחרת באקראי אחד מטור שני-slot-ים הבאים לשדר, ואם עדין מזאה התנשאות, בוחרת באקראי אחד מטור ארבעת ה slot-ים הבאים לשדר, וכך הלאה, עד לשידור מוצלח. מהי הסתברות לשתי התנשויות בדיק, לפני שידור מוצלח (בהתבה שעד סיום השידור לא תהיה עוד תחנה שטרצה לשדר) ?

מטכון מצליף 1

לאחר שארעה התנגשות ראשונה כל אחת מהתנגשות בוחרת בין 2 slots: $\{0,1\} = K$ ולכן ההסתברות לתנגשות נוספת נספפת היא:

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

$$p(1) \cdot p(1) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

$$\frac{1}{4} + \frac{1}{4} = \frac{2}{4} = \frac{1}{2}$$

לאחר מכן אנו רוצים להבטיח שלא תהיה התנגשות (כי בשאלת נדרש לבדוק את ההסתברות ל-2 התנגשויות בדיק).

כטפון מצליף 1

□ ולכן, לאחר ההתגשות השנייה כל תחנה צריכה לבחור בין 4 *slots*: $\{0,1,2,3\} = K$ ולכן הסתברות לתגשות נוספת היא חיבור הסתברויות של המקרים שבהם שתי התגשות 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 התגשות ניקח את הסתברות ל蹶ה המשלים:

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

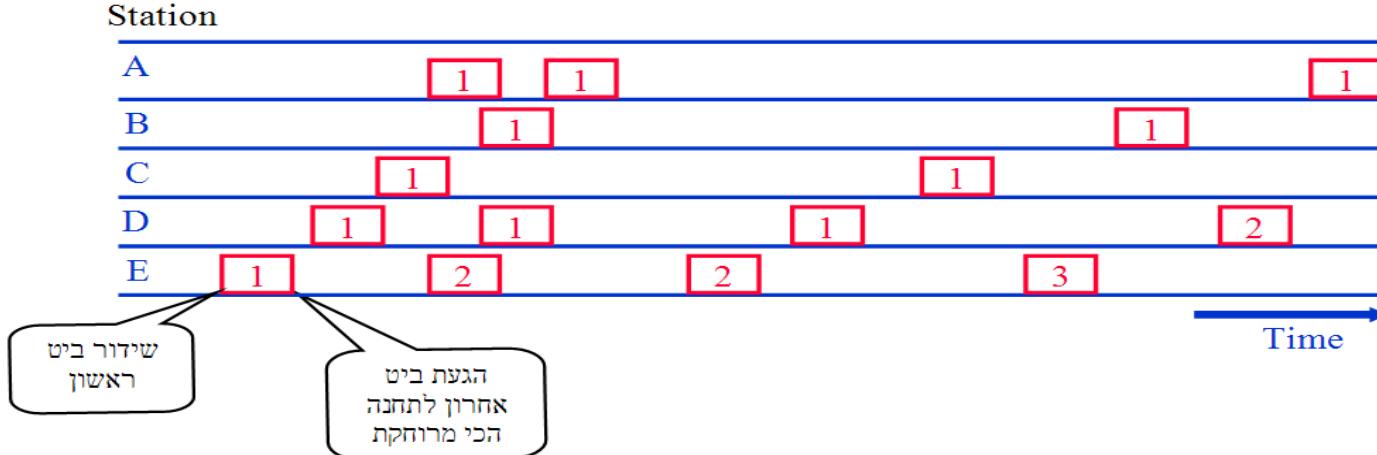
□ ולכן בסה"כ הסתברות ל- 2 התגשויות בד"וק היא:

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

תכליך 2 (תוספת מיכלון 3)

- לפניך איור, הלקוח מההרצאה, המתאר חבילות הנשלחות ע"י חמיש תחנות פעילות ב프וטוקול Aloha.

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

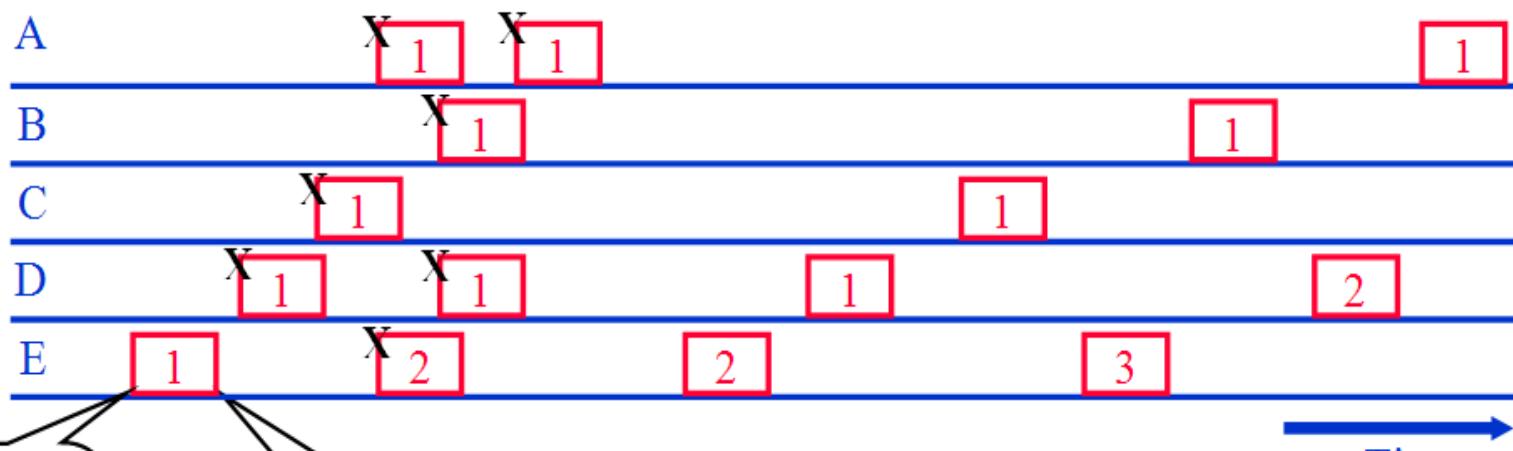


- 2) בפרוטוקול Slotted Aloha אין התנגשויות CAUSED על ידי החלטים כר שלכל תחנה מותר לשדר רק בתחילת חריצ זמן. **בחירה:** נכון / לא נכון וنمוק.
- 3) פרוטוקול CSMA/CD מהו זה שיפור לעומת Aloha ו-Slotted Aloha משום שהתחנה מאזינה לעירוץ לפני שידור וגם בזמן שהוא משדרת על מנת לאוור התנגשויות. **בחירה:** נכון / לא נכון וنمוק.

מטכון מכניג 2

סעיף 1:

Station



שידור בית
ראשון

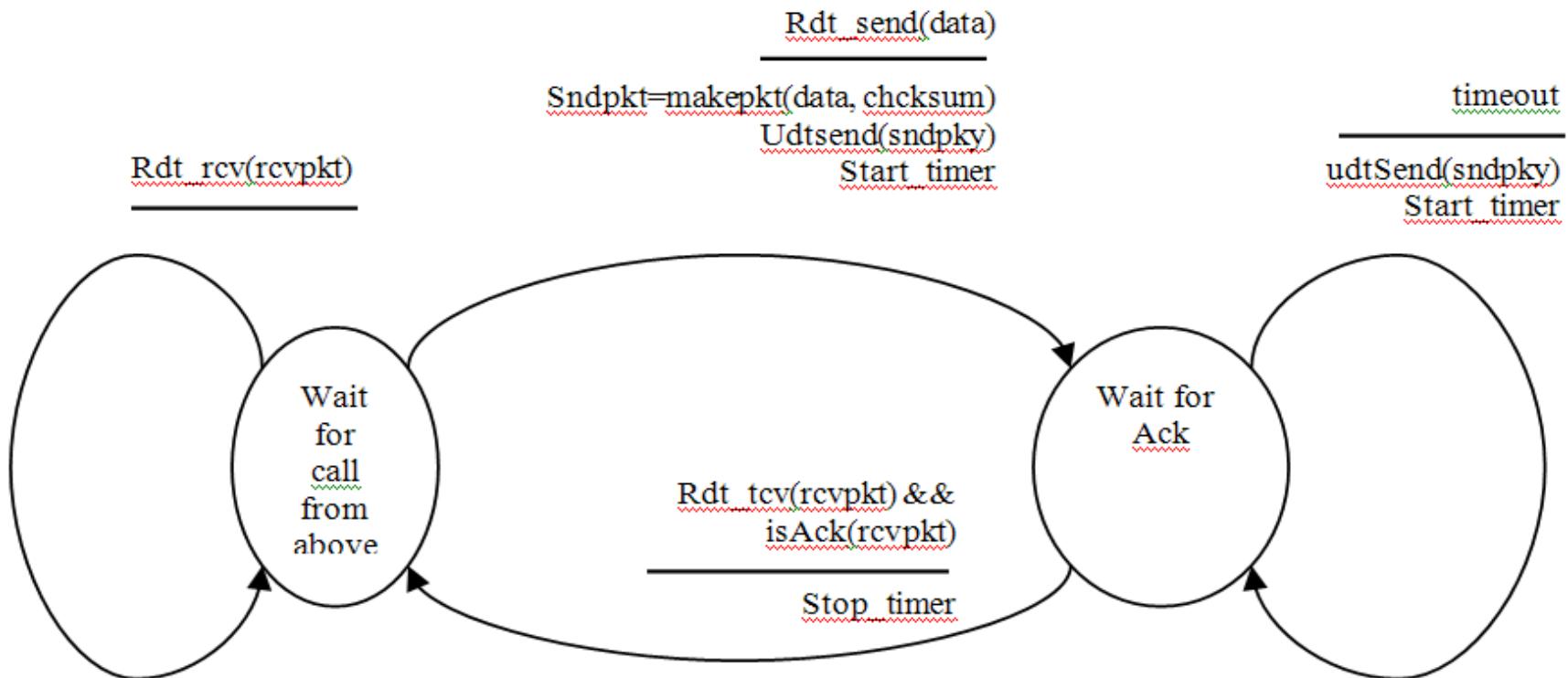
הגעת בית
אחרון לתחנה
הכי מרוחקת

מטכון מצל מג 2

- סעיף 2': לא נכון. ב-Aloha Slotted עדין יתכו התנטשוויות, הרעיון הוא פשוט לחלק לחרצים על מנת להגדיר זמן שידור (ולכן גם התנטשוויות) או פרק זמן בו חבילות בערוץocr וכר להגדיל יעילות.
- סעיף 3': נכון. ב-CD/CSMA התחנה מאזינה לערוץ לפני שידור וגם בזמן שידורocr וכר מאותרות התנטשוויות בעת שהן קורות ואין בזבוז זמן. لكن גם יש שיפור ביעילות.

תכליך 3

להלן דיאגרמת מצבים לשולח בפרוטוקול פשוט לשידור אמין:



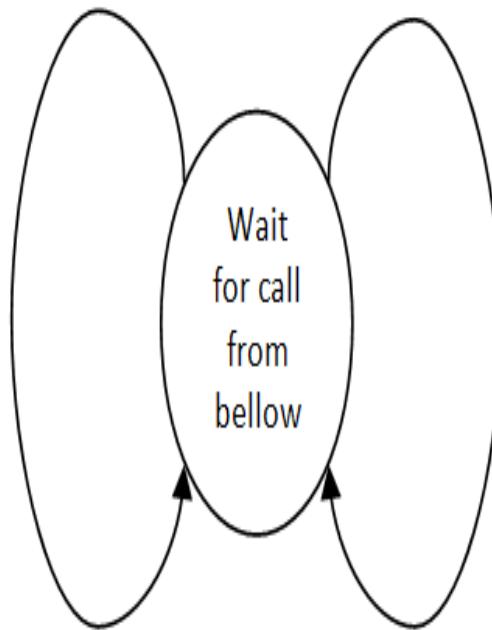
מכפלי 3 (האקס)

- א- הצג דיאגרמת מצבים למקבל.
- ב- באילו הנחות, אם בכלל, מבטיח ה프וטוקול אמינות? (הנחות לדוגמא: שכפול חבילות, הגעת חבילות לפי סדר, השהייה מירבית בערז, שיבוש חבילות, אבדן חבילות...)
- ג- הצג תרשימים מאורעوت שמדגים שהפרוטוקול לא מבטיח אמינות בשימוש מעל פרוטוקול IP.

כתיבון מילויים

◻ סעיף א':

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



```
rdt_rcv(rcvpkt)
  && corrupt(rcvpkt, checksum)
  _____
```

מטכון מצל מג 3

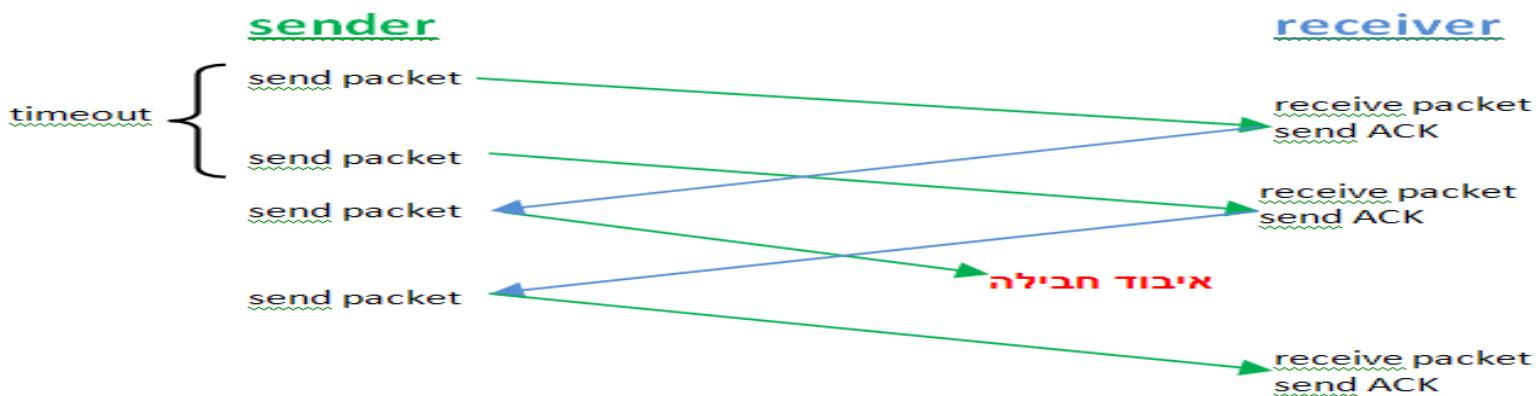
◻ סעיף ב': הפרוטוקול מבטיח אמינות בהנחות הבאות:

1. אין שיכפול חבילות. ה프וטוקול לא מאפשר לבדוק האם החבילות שהתקבלו הן שיכפול אחת של השניה. וכך אם יהיה שיכפול, המקלט ישלח ACK על החבילה עצמה, ו-ACK נוסף על השיכפול. לאחר שהשלוח קיבל את ה-ACK הראשון הוא ישלח חבילה נוספת, וכך יגיע ACK על השיכפול של החבילה הקודמת, והשלוח יבין שהמקלט קיבל את החבילה החדשה, למרות שהחביבה הייתה יכולה להיבז בדרכו.
2. יש השהייה מירבית בערוץ, שלפייה יקבע ה-timeout של השלוח. אחרת אם לא תהיה ההשהייה מירבית יוכל להיווצר שיכפול חבילות – ה-timeout יסתיים והחביבה תשלוח שוב, למרות שהחביבה המקורית הגיעה, רק שלקח לה יותר זמן להגעה. וכך נקלט ACK מיותר וכך יוכל להיווצר גם איבוד חבילה (כמו בשיכפול חבילות).
3. כל השגיאות מתגלות – לאחר מכן מקבל ישלח לאפליקציה מידע שגוי, או שהשלוח קיבל ACK משובש שלא יזהה אותו כ-ACK, אחרי timeout ישלח את החביבה שוב, וייווצר שיכפול.
4. אין אובדן של הודעות ACK – לאחר מכן שלוח יחבר ל-timeout, וישלח את החביבה שוב. וכך המקלט יוכל שיכפול של החביבה שכבר קיבל, ולא יוכל לזהות זאת.

כטalon מצל'יג 3

סעיף ג':

- IP שיר לשכבה ה- Network ואינו מבטיח אמינות, מבטיח רק שה-header יגיע תקין ע"י checksum. יכול לקרות שכפול הודעות, שגיאות בחבילות, איבוד חבילות ולא ניתן לדעת את גודל ההשניה המksamילית.
- תרשימים מאורעווים 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 (בשלילה הראשונה החבילה התעכבה), ושלח שני ACKs, אחרי ה-ACK הראשון ה-sender שלח את החבילה השנייה ולאחר ה-ACK השני (על שכפול החבילה הראשונה) שלח את החבילה השלישית, למרות שהחבילה השנייה נאבדה.

תכליט 4

- ❑ נתיחס למאורע זיהוי אובדן חビלה בפרוטוקול TCP, ונסמן ב- χ את מספר הסegment ה- i מוקדם שעדין לא התקבל עליו אישור (Ack). הרבה מימושי TCP שולחים מחדש רק את segment i , אף כי הסטנדרט מרצה גם לשלח מחדש גם segments שנשלחו אחרי i (וגם עליהם לא התקבל Ack); segments אלו נשלחים כאשר מקבל ה-ACK על segment i . התיחס לצבים הבאים, כאשר תמיד הידוע זורם רק מהשלוח לנמען (ש רק שלוח 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 "יעודי" (לא שותפים).
- I. תן דוגמא ל蹶ה בו עדיף לשלח מחדש את כל segments ולא רק את i , או נמק בקצרה מדוע אין לעולם蹶ה כזה.
- II. תן דוגמא ל蹶ה בו עדיף לשלח מחדש את segment i , או נמק בקצרה מדוע אין לעולם蹶ה כזה.
- III. מה המימוש העדיף לדעתך לצב זה (نمוק בקצרה)?

המשך מכך 4

□ נזין לקליפ

תכליט 4 (האפק)

- (b) תחנת הקרה (ground control) משדרת נתוני חשובים בזמן אמת למעבורת חלל (major Tom), הנמצאת בגובה 35,000 ק"מ בקצב 100Mbps . הנה שהפרעה אופיינית בשידור כזה נמשכת כעשרה שניות.
- תן דוגמא ל蹶ה בו עדיף לשולח מחדש את כל הסגמנטים ולא רק את ח, או נמק בקצרה מדווקא אין לעולם蹶ה כזו.
 - תן דוגמא ל蹶ה בו עדיף לשולח מחדש רק את סגמנט ח, או נמק בקצרה מדווקא אין לעולם蹶ה כזו.
 - מה המימוש העדיף לדעתך למצב זה (נמק בקצרה)?
- (c) השולח והנמען מחוברים דרך האינטרנט.
- תן דוגמא ל蹶ה בו עדיף לשולח מחדש את כל הסגמנטים ולא רק את ח, או נמק בקצרה מדווקא אין לעולם蹶ה כזו.
 - תן דוגמא ל蹶ה בו עדיף לשולח מחדש רק את סגמנט ח, או נמק בקצרה מדווקא אין לעולם蹶ה כזו.
 - מה המימוש העדיף לדעתך למצב זה (נמק בקצרה)?

כטלוון מצליג 4

□ סעיף א:

- I. כשהשלוח נמצא למרחק מקסימלי, שמאפשרת החומרה, מהמקבל, יכולים לKHRות הרבה איבוד סגמנטים. لكن אחרי איבוד סגמנט ח עדיף ישר אחרי שלילה חדשה של הסגמנט וקבלת ה-ACK עליו, לשלוח את הסגמנטים הבאים מחדש, כי בהסתברות גבוהה עוד סגמנטים נאבדו.
- II. למרחוק סביר (לפי החומרה) תהיה תקשורת טובہ בערז תקשורת זה בין השולח למקבל, אבל בכל זאת יכולים מדי פעם להיאבד סגמנטים. מכיוון שאיבוד סגמנטים במקרה כזה לא יהיה תמיד אפשר לשלוח מחדש רק את הסגמנט ח.
- III. עדיף לשלוח רק את סגמנט ח מכיוון שברוב המקרים למרחוק בין השולח למקבל יהיה סביר, והסיכוי שנוכל למקם את השולח והמקבל למרחוק המקסימלי שמאפשרת החומרה מבלתי שהקשר יתנתק לגמרי הוא קטן.

כטלוון מצלים 4

□ סעיף b:

- I. על השידור למעבורת החלל יכולים להשפיע שידורים ממוקרות אחרים ותופעות טבע שונות, لكن זהו ערז תקשורת לא אמין, גם RTT בערז תקשורת זה אחר וכאן עדיף לשלוח מחדש את הסגמנטים שנשלחו מיד אחרי כי כנראה גם הם נאבדו בהפרעה.
- II. אם לשלוח (ground control) יש מעבד חלש שאחראי גם על שליחת המידע, אז כדי לא להעmis עליו עדיף לשלוח רק את הסegment ה-*n* שבוואדיות לא הגע.
- III. עדיף לשלוח מחדש את כל הסגמנטים החל מ-*n* כי ברוב המקרים לשלוח (ground control) יהיה מספיק כוחعيבוד כדי להקצות משאים ולשלוח את הסגמנטיםשוב; וגם בغالל ההפרעות הארוכות יחסית יש הסתברות גבוהה שיותר מסegment אחד נאבד/נפגם; ובغالל ה- RTT הגבוה, עדיף לא לחכות להודעה מהמקבל על כל Segment שנפגם, אלא לשלוח את כל הסגמנטים שנשלחו אחרי *n* מחדש (לאחר שהתקבל ACK על Segment *n*).

כטלוון מצליג 4

□ סעיף C:

- I. אם תשתיית האינטרנט בין השולח למקבל מאפשרת תקשורת רק עם RTT גבוהה, אז עדיף כמה שפחות לחכות לתגובה של מקבל, ולשלוח מחדש את כל הסגמנטים.
- II. אם תשתיית האינטרנט טובה, אז חוץ מה-RTT הנמור, הסתברות לשיבוש/איבוד סגמנטים תהיה קטנה, ולכן אם יקרה שהסמנט נאבד, עדיף לשלוח רק אותו.
- III. ברוב המקרים היום על תשתיית האינטרנט יש RTT נמור, והסגמנטים משתמשים/נאבדים לעיתים רוחקות, לכן אם נאבד סמנט, עדיף לשלוח רק אותו.

מבחן 5

ענה לכל סעיף בנכון/לא נכון? וסביר את תשובה.

1. גודל חלון ה- TCP Advertised Window ב-connection אינו משתנה במשך הזמן.
2. צומת A שולח לצומת B קובץ גדול מעל TCP. בזמן נתון תוקף כדי השילחה מספר הבטים 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 סegment הוא m , איזה sequence number של הסgement הבא הוא בהכרח $m+1$.
4. צומת A שולח לצומת B שני Segments של TCP, כסדראשון.sequence number הוא 90 ובשני הוא 110. נניח שהראשון אבד. מה יהיה ה-ACK של Segment B?
5. הצומת השולח מחשב את הזמן שעובר משילחת חבילת, ורק כאשר הזמן עבר סף מסוים הוא יודיע שחבילת לא הגיע ליעד.

אתגר מכך'יג 5

- ▢ גודל חלון ה- TCP Advertised Window ב-connection.
- ▢ תשובה: לא נכון - הchlון משתנה לפי העומס.
- ▢ צומת A שולח לצומת B קובץ גדול מעל TCP. בזמן נתון תוך כדי השילחה מספר הבטים שלא עשו עליהם ACK לא עולה על גודל ה-Receive buffer.
- ▢ תשובה: נכון - מספר הבטים שלא עשו עליהם ACK תמיד קטן או שווה לגודל הbuffer אצל המקלט.
- ▢ צומת A שולח לצומת B קובץ גדול מעל TCP. אם ה-sequence number של סגמנט הוא m , אז ה-sequence number של הסגמנט הבא הוא בהכרח $1+m$.
- ▢ תשובה: לא נכון - המספר הסידורי הינו m ומספר הבטים בסגמנט $(m+MSS)$ או $(n+m)$ ולא בהכרח $1+m$.

אתגר מצל מג' 5

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

תשובה: **90**

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

תשובה: לא נכון - גם קבלת ACK מזמן איבוד חבילה

תכליט 6

לכל אחת מהטענות הבאות, רשום נכון / לא נכון והסביר בקצרה.

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

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

(3) במהלך שימוש ב- IP/TCP כאשר עובדים ב- IPv4, הנתבים בהם עוברת הودעה מבצעים checksum בשכבה הרשת. נכון / לא נכון

(4) במהלך שימוש ב- IP/TCP כאשר עובדים ב- IPv4, הנתבים בהם עוברת הודעה מבצעים checksum בשכבה ה- transport. נכון / לא נכון

תכליף 6 (האפק)

- (5) במהלך שימוש ב- TCP/IP אשר עובדים ב- IPv6, השירותים מבצעים checksum בשכבה הרשת. נכון / לא נכון
- (6) במהלך שימוש ב- TCP/IP אשר עובדים ב- IPv6, הנתבים בהם עברת הودעה מבצעים checksum בשכבה הרשת. נכון / לא נכון
- (7) במהלך שימוש ב- TCP/IP אשר עובדים ב- IPv6, הנתבים בהם עברת הודעה מבצעים checksum בשכבה ה- transport. נכון / לא נכון
- (8) במהלך שימוש ב- TCP/IP אשר עובדים ב- IPv6, השירותים מבצעים checksum בשכבה ה- transport. נכון / לא נכון

כטלוון מילוי 6

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

נכון

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

נכון

(3) במהלך שימוש ב- IP/TCP כאשר עובדים ב- IPv4, הנتابים בהם עברת הودעה מבצעים checksum בשכבה הרשת.

נכון

הסבר - **Checksum**: מציאת שגיאות פשוטות בלבד ללא מנגןון לתקן שגיאות.

השולח: מחלק את הסגמנט לקטעים של 16 ביט. לחבר אותם ומבצע השלים ל 1-, ואת התוצאה שם בשדה ה-**checksum**.

ה מקבל: מחשב את ה-**checksum** של הסגמנט שקיבל ובודק אם המחשב שווה לזה שהתקבל. אם הוא שונה הtgtלטה טעות, ואם זהה לא הtgtלטה טעות אבל ייתכן שהייתה טעות.

פרק 9 מילוי ו(האפק)

- ב프וטוקול ה-IP אחד השדות הוא שדה סכום הביקורת (checksum) שבו שדה בדיקה לוידוא תקינות הcotרת (header) בלבד.
- لكن כל מי מתעסק בשילוח/קבלת מידע ובניטוב "שכבה הרשות" ו-"שכבה ההובללה" (שרתים, מחשבי קצה ונתבים) חייב לדעת שהມידע שהוא מקבל אמין ולкан הבדיקה.
- במהלך שימוש ב- TCP/IP כאשר עובדים ב- IPv4, הנתבים בהם עברת הودעה מבצעים checksum בשכבה ה- transport.
לא נכון
הסבר – בנתבים קיימות רק 3 השכבות הראשונות: הפיסית, קישור הנתונים והרשות, ושכבת התעבורה אינה קיימת כלל, ומכאן שברור שהתשובה היא לא נכון.
- במהלך שימוש ב- TCP/IP כאשר עובדים ב- IPv6, השירותים מבצעים checksum בשכבה הרשות.
לא נכון
במהלך שימוש ב- TCP/IP כאשר עובדים ב- IPv6, הנתבים בהם עברת הודעה מבצעים checksum בשכבה הרשות. **נכון / לא נכון**
לא נכון

התקן TCPv6 (הנקה)

- 7) במהלך שימוש ב- IP/TCP כאשר עובדים ב- 6vPI, הנתבים בהם עברת הودעה מבצעים checksum בשכבה ה- transport. נכון / לא נכון
לא נכון

הסביר - 6vPI עבר שינוי מ-IPv4: "Checksum" הסרת תcona זו לחלוטין כדי שנוכל להוריד את זמן התהלייר בכל קס (\"nitour" נקודת חיבור בין מחשבים ברשת). לכן אין מבצעים checksum בשכבה הרשות בנתבים וברשתים. כמו שכבר אמרנו שכבה ה- transport לא קיימת בנתבים.

- 8) במהלך שימוש ב- IP/TCP כאשר עובדים ב- 6vPI, השירותים מבצעים checksum בשכבה ה- transport. נכון / לא נכון
נכון

הסביר – פרוטוקול 6vPI מבצע checksum בשכבה ה- transport עבור המידע.

תכליף 7

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

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

(2) חוזר על א' עברו $30 = n$?

(3) חוזר על א' עברו $60 = n$?

כטבון מילוי 7

(1) ללא Pipelining (גודל חלון $n=1$) מהו אחוז הזמן שבו המחשב השולח עוסק בשלוח את החבילות (Utilization) ?

נתון:

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

$$n = 1$$

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

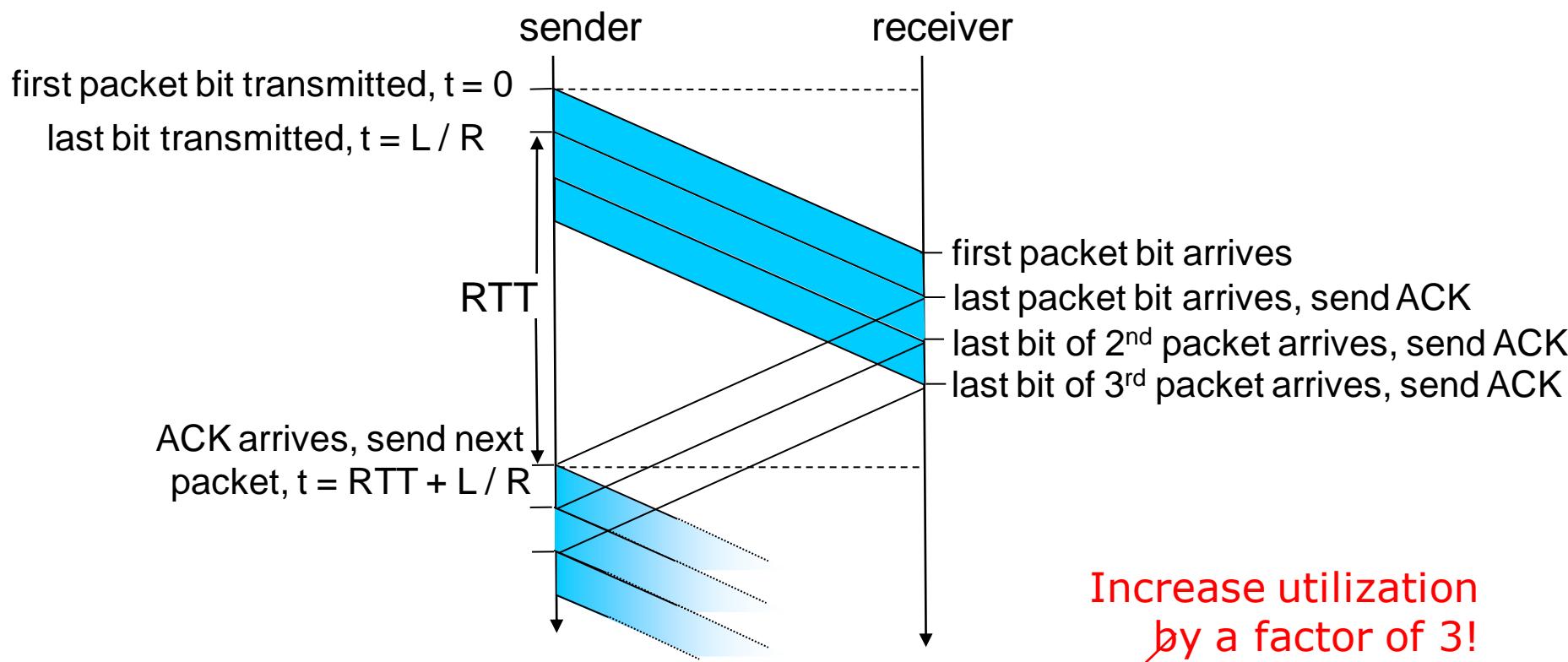
$$R = 100\text{Mbps}$$

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

$\approx 1.96\%$ utilization.

পিলিনিং Pipelining

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



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

7 מכניזם

$$\text{über } n = 30 \quad (2)$$
$$L/R) / (RTT + (L/R)) = ((2.5*8/100)*30) / (10 + (2.5*8/100))$$
$$\approx 58.8\% \text{ utilization}$$

$$\text{über } n = 60 \quad (3)$$
$$L/R) / (RTT + (L/R)) = ((2.5*8/100)*60) / (10 + (2.5*8/100))$$
$$\approx 100\% \text{ utilization}$$