# רשתות תקשורת מחשבים - תרגיל בית תיאורטי 5

## שאלה 1

### A

פקטה בגודל 800 בתים = 6400 ביטים, וזמן השליחה הוא 10^7 מילי שניות לביט 🡸 פקטה נשלחת תוך 6400/(10^7) = 64/(10^5) מילי שניות.  
RTT = 2\*propagation time = 2\*30 מילי שניות = 60 מילישניות.  
גדול החלון הוא N=50.

נצילות הערוץ (U) =

### B

**נעדיף להשתמש בSR כאשר הערוץ רועש.**   
הסיבה לכך שנרצה שפקטות שהגיעו ליעדן לא יזרקו על אף שפקטות בעלות seq נמוך יותר עדין לא הגיעו   
(ב-GBN הן כן יזרקו אצל המקבל).

### C

נבחר חלון בגודל N=3 🡸 2n-1 = 5 מספרים סידוריים.  
נניח והשולח שלח את כל שלושת הפקטות בחלון שלו: הן בעלות seq 1, 2 ו-3.  
**למקבל הגיעו כל 3 הפקטות** (כגודל החלון שלו) ולכן הוא שלח ack עם 1, 2 ו-3 , **והוא מקדם את החלון שלו לseqים** 4, 5 ושוב **1** (זאת מיכוון שהמספרים הסידוריים הם עד 5 והם מעגליים).  
**לצערו של השולח קבל את ה-ackים רק על 2,3 ולכן שולח שוב את הפקטה הראשונה ששלח עם seq 1.  
המקבל אכן מצפה לקבל פקטה עם seq 1 ולכן ישמור אותה** וישלח עליה ack – זאת על אף שהפקטה הראשונה של השולח כבר נשמר אצל המקבל 🡸 השולח והמקבל לא מודעים לכך שהפקטה הראשונה שנשלחה מהשולח התקבלה פעמיים אצל המקבל 🡸 כישלון בפרוטוקול.

## שאלה 2

### A

בפרוטוקול TCP הספירה מתקדמת בגדלים של בתים, לכן ניתן לשלוח קובץ עד גודל של 2^32 בתים, ללא התנגשות של seqים.

### B

MSS=1514, וגודל הheader הוא 90 🡸 גודל פקטה "מלאה" הוא 90+1514 = 1604.  
קצב השליחה הוא 15\*(10^6) ביטים לשניה.  
  
1) אנו מעונינים לשלוח 2^32 בתים 🡸 כמות הפקטות שעלינו לשלוח היא 2^32 / 1514 = 2,836,834 פקטות "מלאות" עם 1514 בתים של מידע, ועוד פקטה אחת עם 620 בתים של מידע.  
  
2) גודל פקטה "מלאה" בביטים הינו 1604\*8 = 12,832, ושליחתה לוקחת לנו 12,832/ (15\*10^6) שניות = 0.000855 שניות. הפקטה החלקית שגודלה בביטים עם הheaderים הוא 710 \* 8 = 5680 יקח לשלוח 5680/(15\*10^6) = 0.000378 שניות.

🡸 משך הזמן שיקח לנו לשליחת כל הקובץ הוא: 0. 000855) 2,836,834 \* ) + 0. 000378 = **2,425.493 שניות.**

(\*) כמות המידע שנשלח מהקובץ היא 2^32, וכמות הheaderים שמתווספים היא 2,836,834+1 \* 90 🡸  
אחוז המידע מהקובץ שנשלח ביחס לכל המידע שנשלח ברשת הוא2^32 / (90 \* 2,836,835 + 2^32) =**0.94389**

## שאלה 3

### i

**אנחנו במצב slow start**. במצב זה העלייה של ה- cwnd היא אקספוננציאלית -> על כל ack החלון גדל כגודל הבתים שאושרו - לצורך פשטות (שהוגדרה גם בשיעור) אנו נניח שגודל התוכן של כל פקטת TCP הוא תמיד ה-MSS.  
  
גדילה אקספוננציאלית קורת גם במצב fast recovery, אך יש הבדל **שלא מאפשר היתכנות של FR** במקרה הזה:

כשנכנסים לFR הנוסחא לthreshold החדש ביחס לcwnd החדש היא:  
newCwnd = newThreshold + 3 \* MSS.  
במקרה זה אנחנו יודעים שגודל החלון החדש הוא לכל היותר 6000, שזהו בדיוק 3\*MSS , נציב בנוסחא: 6000 = newThreshold + 6000 , ונקבל כי ה- threshold החדש היה צריך להיות 0. מצב זה לא ייתכן מיכוון שבאף שלב בריצת האלגוריתם ה- threshold לא נקבע להיות 0.  
(נשיב לב שבמידה ה newCwnd היה קטן יותר, ה newThreshold היה שלילי - לא ייתכן באלגוריתם).

בנוסף, **המצב אינו CA,** מיכוון שגדילת החלון בו היא לינארית, עבור כל ack החלון גדל ב-1/cwnd – ולא כך המצב במקרה זה.

### ii

**הערך המקסימלי אינו ידוע.**

בשלב slow start: הערך של ה-threshold נקבע להיות הערך בו היה הcwnd לפני התחלת השלב חלקי 2 או במידה ומדובר בריצה הראשונה אין ערך threshold – הוא יקבע רק לאחר ה- timeout הראשון.  
בכל מקרה אנחנו לא יכולים לדעת מה ערך הthreshold מיכוון שאנחנו עדין לא הגענו אליו.

### iii

בעקבות 3 duplicate ack הערך החדש של הcwnd נקבע להיות חצי מ ה cwnd הנוכחי + 3\*MSS.  
 + 3\*2000🡸 10,000/2 = **11,000**.  
נשים לב שזאת הגדרת חלון זמנית, הוא גדל בצורה אקספוננציאלית ככל שנקבל עוד ackים כפולים, ולאחר מכאן:

1. במידה ונקבל את הack על הפקטה שעשינו לה retransmit הערך של הcwnd יחזור להיות 5000 (10,000/2) ונעבור למצב של CA.
2. במידה ולא נקבל את הack (כלומר יקרה timeout) החלון יקטן ל-1, ונעבור למצב slow start.