CRC, ARQ, חלון הזזה

הרצאה 4

22 אפריל 2025

נושאים להיום

- שגיאות
  - גילוי שגיאות CRC
  - תיקון שגיאות
- שליחה אמינה ARQ
  - עצור וחכה
  - חלון הזזה

Cyclic Redundancy Check

קל לממש

פרוטוקולי עורק משתמשים בו

- CRC-32 באתרנט, 802.5, PKzip, ...
- CRC-CCITT ב-HDLC
- CRC-8, CRC-10, CRC-32 ב-ATM

ליותר מידע

- ערך בויקיפדיה
- https://en.wikipedia.org/wiki/Cyclic_redundancy_check

יותר טוב מסיבית זוגיות ו-checksum

- (למשל, 32 סיביות לשלוח 12000)

Cyclic Redundancy Check (CRC)

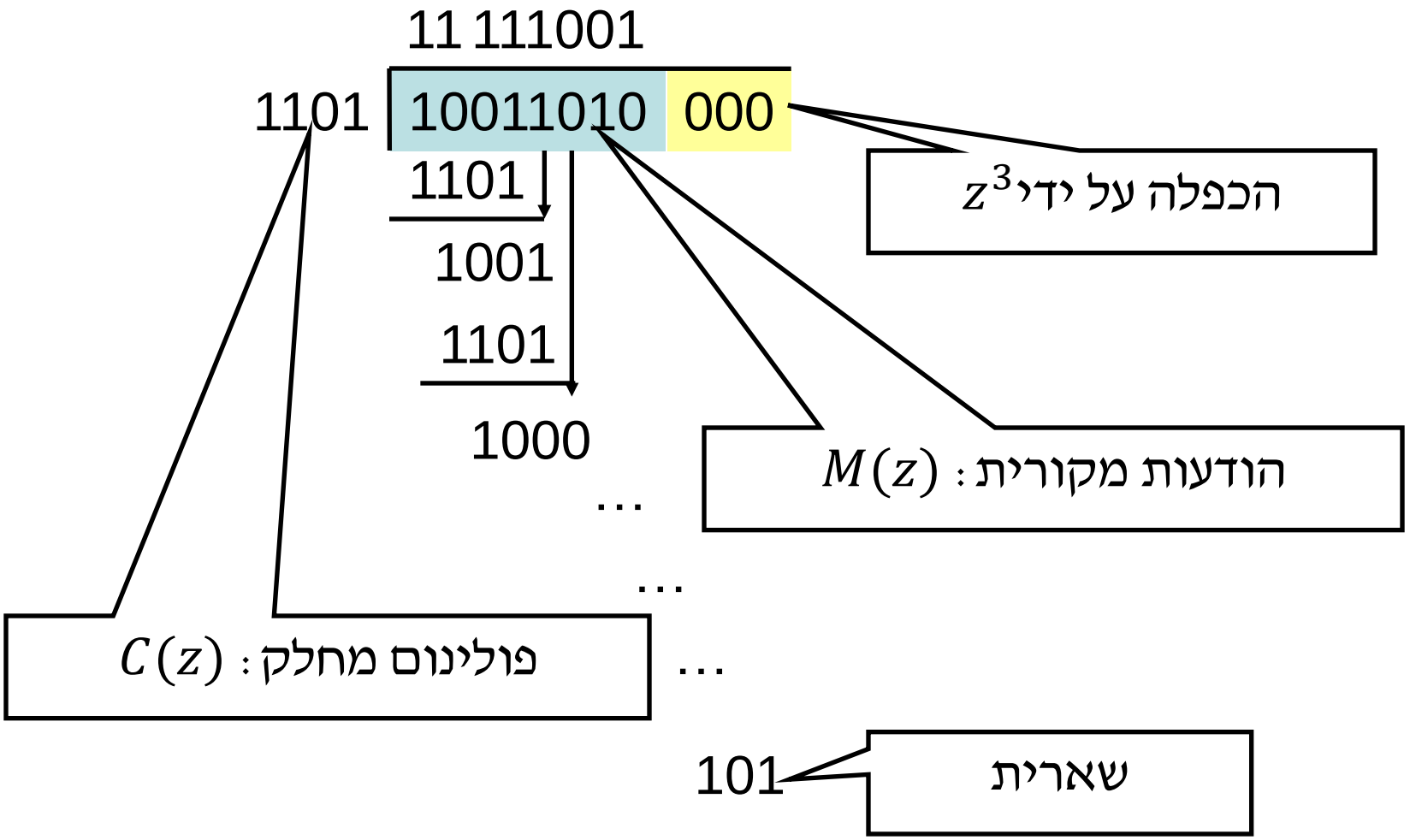
- נתייחס להודעה עם $(n+1)$ סיביות כאל פולינום במעלת n
  - מתמטיקה פולינומים מודולו 2
  - ערכי הסיביות בהודעה הינן המקדמים
  - הודעה $= 10011010$
  - פולינום:

$$M(z) = (1 \times z^7) + (0 \times z^6) + (0 \times z^5) + (1 \times z^4) + (1 \times z^3) + (0 \times z^2) + (1 \times z^1) + (0 \times z^0) = z^7 + z^4 + z^3 + z^1$$

אלגוריתם CRC

1. השולח והמקבל מחליטים על פולינום מחלק $C(z)$ במעלה k
   - דוגמה: $k = 3$
   - $C(z) = z^3 + z^2 + 1$
   - מקדמים: 1101
2. סיביות גילוי השגיאות הינן השארית של $(M(z) \times z^k)$ מחולק על ידי $C(z)$
3. התוצאה הינה פולינום שליחה בגודל $n + k$ סיביות $P(z)$ שמתחלק על ידי $C(z)$

חישוב CRC לדוגמה



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1					1	1	1	1	1	0	0	1			
2	1	1	0	1	1	0	0	1	1	0	1	0	0	0	0
3					1	1	0	1							
4					0	1	0	0	1						
5						1	1	0	1						
6						0	1	0	0	0					
7							1	1	0	1					
8							0	1	0	1	1				
9								1	1	0	1				
10								0	1	1	0	0			
11									1	1	0	1			
12									0	0	0	1	0	0	0
13												1	1	0	1
14												0	1	0	1

חישוב CRC לדוגמה

$z^3 \times M(z)$	10011010	000
שארית +		101
$P(z)$הודעה לשלוח	10011010	101

- המקבל בודק ש-$C(z)$ מחלק את ההודעה שהתקבלה

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1					1	1	1	1	1	0	0	1			
2	1	1	0	1	1	0	0	1	1	0	1	0	1	0	1
3					1	1	0	1							
4					0	1	0	0	1						
5						1	1	0	1						
6						0	1	0	0	0					
7							1	1	0	1					
8							0	1	0	1	1				
9								1	1	0	1				
10								0	1	1	0	0			
11									1	1	0	1			
12									0	0	0	1	1	0	1
13												1	1	0	1
14												0	0	0	0

גילוי שגיאות על ידי CRC

- צריכים לבחור מחלק $C(z)$ טוב
- יש הרבה אפשרויות תקניות:
  - CRC-8, CRC-10, CRC-12, CRC-16, CRC-32,
  - CRC-32: $0x04C11DB7$ (1 0000 0100 1100 0001 0001 1101 1011 0111 or $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$)

מגלה:

כל שגיאה של 1 סיבית כל עוד ומקדמי z^k ו-z^0 שוות 1	כל שגיאה של 2 סיביות כל עוד ו- $C(z)$ בנוי מ-3 רכיבים	כל מספר אי זוגי של שגיאות אם $(z+1)$ מחלק $C(z)$	כל שגיאת רצף בגודל $\leq k$

מימוש CRC

- קל למימוש בחומרה
  - חיסור בבסיס 2 הוא פשוט XOR
  - רגיסטר הזזה עם k סיביות עם שערי XOR לפני כל מקדם 1 ב-$C(z)$ (חוץ מהראשון)
  - מזיזים את ההודעה לשבב, השארית מתמלאת ברגיסטרים

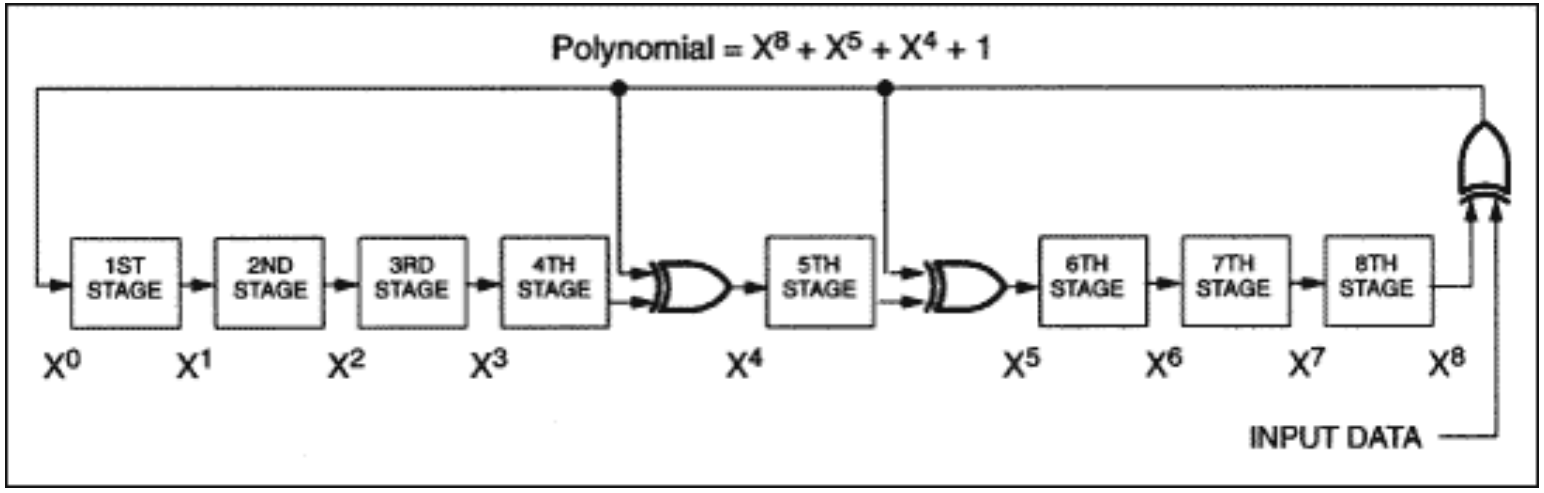
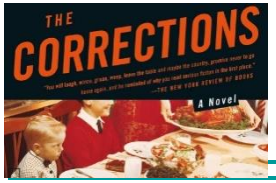


Image source: https://www.analog.com/en/resources/technical-articles/understanding-and-using-cyclic-redundancy-checks-with-maxim-1wire-and-ibutton-products.html



קודים לתיקון שגיאות

ניתן להשתמש במידע מיותר לתקן שגיאות מסוימות

- בדרך כלל דורש יותר מידע מיותר

שיקול דעת הנדסי

- גילוי שגיאות מחייב שליחה מחדש
- תיקון שגיאות מחייב שליחת יותר סיביות תמיד

Forward Error Correction שימושי כאשר:

- הסיכוי לשגיאות **גבוה** (למשל, חיבור אל-חוטי)
- השהייה **ארוך מדי** לשליחה מחדש יעילה (למשל, שידורי לוין)

עד כה

- שגיאות
  - גילוי שגיאות CRC
  - תיקון שגיאות
- שליחה אמינה ARQ
  - עצור וחכה
  - חלון הזזה

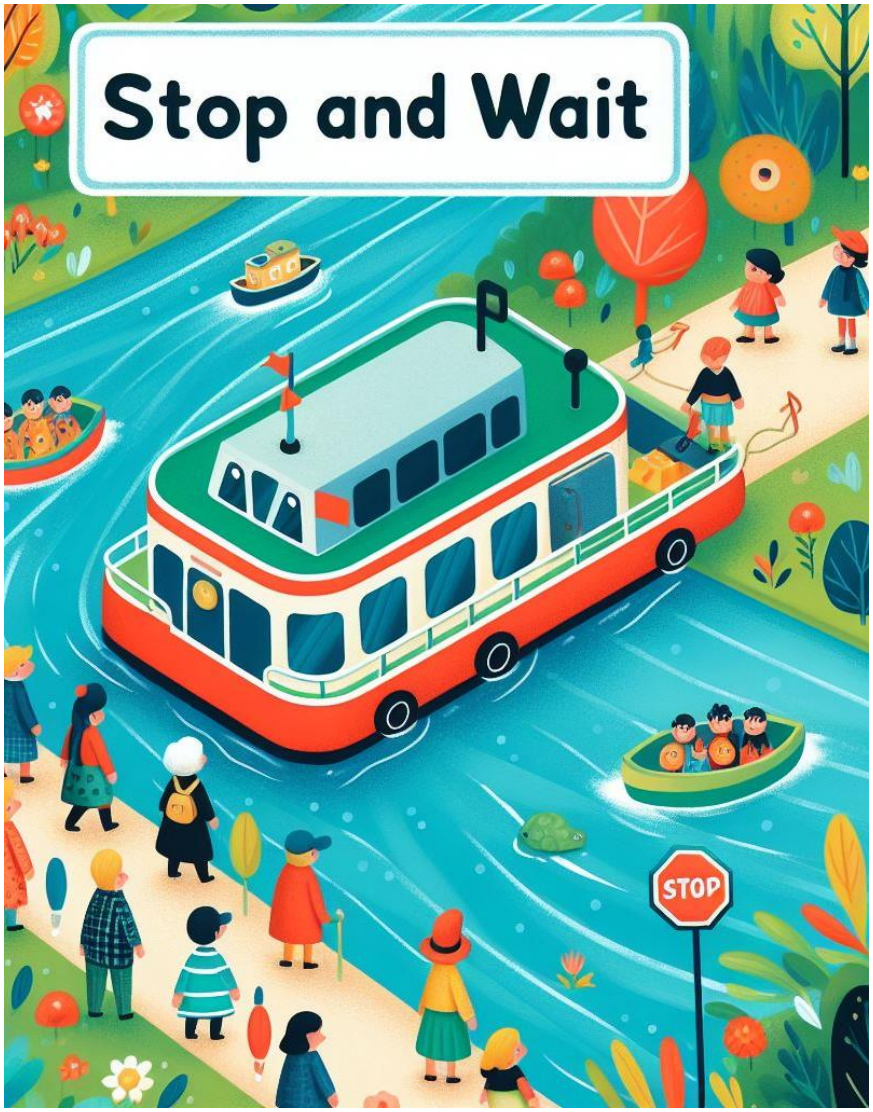
מצאנו שגיאה. מה עושים עכשיו?

- מה כדאי לשולח ולמקבל לעשות?

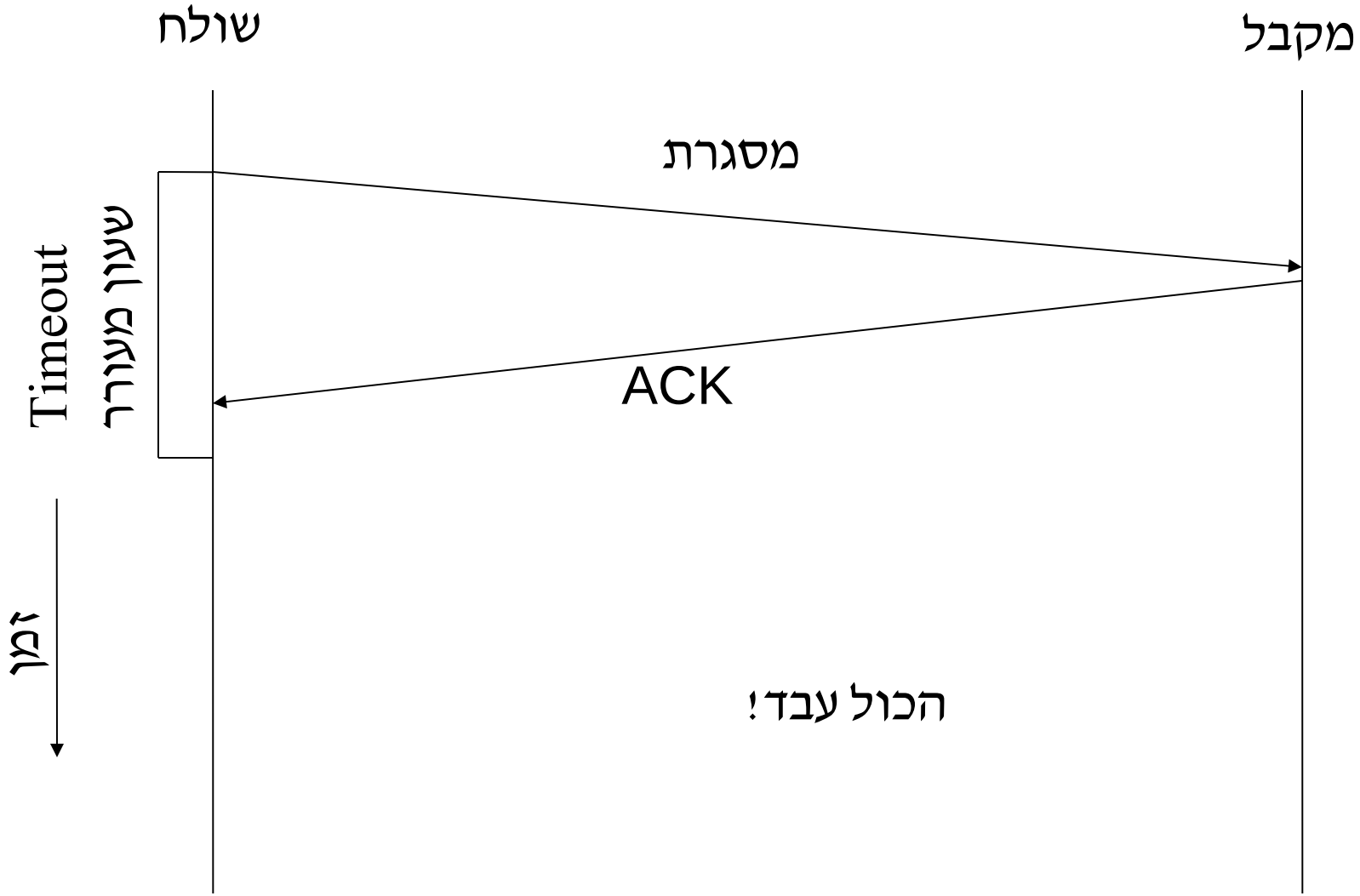
- אישורי קבלה (ACK) ☑
  - מנת בקרה ושליטה קטנה (מעט נתונים)
  - כאשר השולח מקבל ACK, הוא יודע שהמקבל קבל מסגרה
- שעון מעורר ⧖
  - אם השולח לא קבל ACK אחרי זמן "סביר", הוא שולח המסגרת מחדש
- השיטה הכללית: *Automatic Repeat Request (ARQ)*
  - שליחה חוזרת אוטומטית

עצור וחכה – שיטה פשוטה

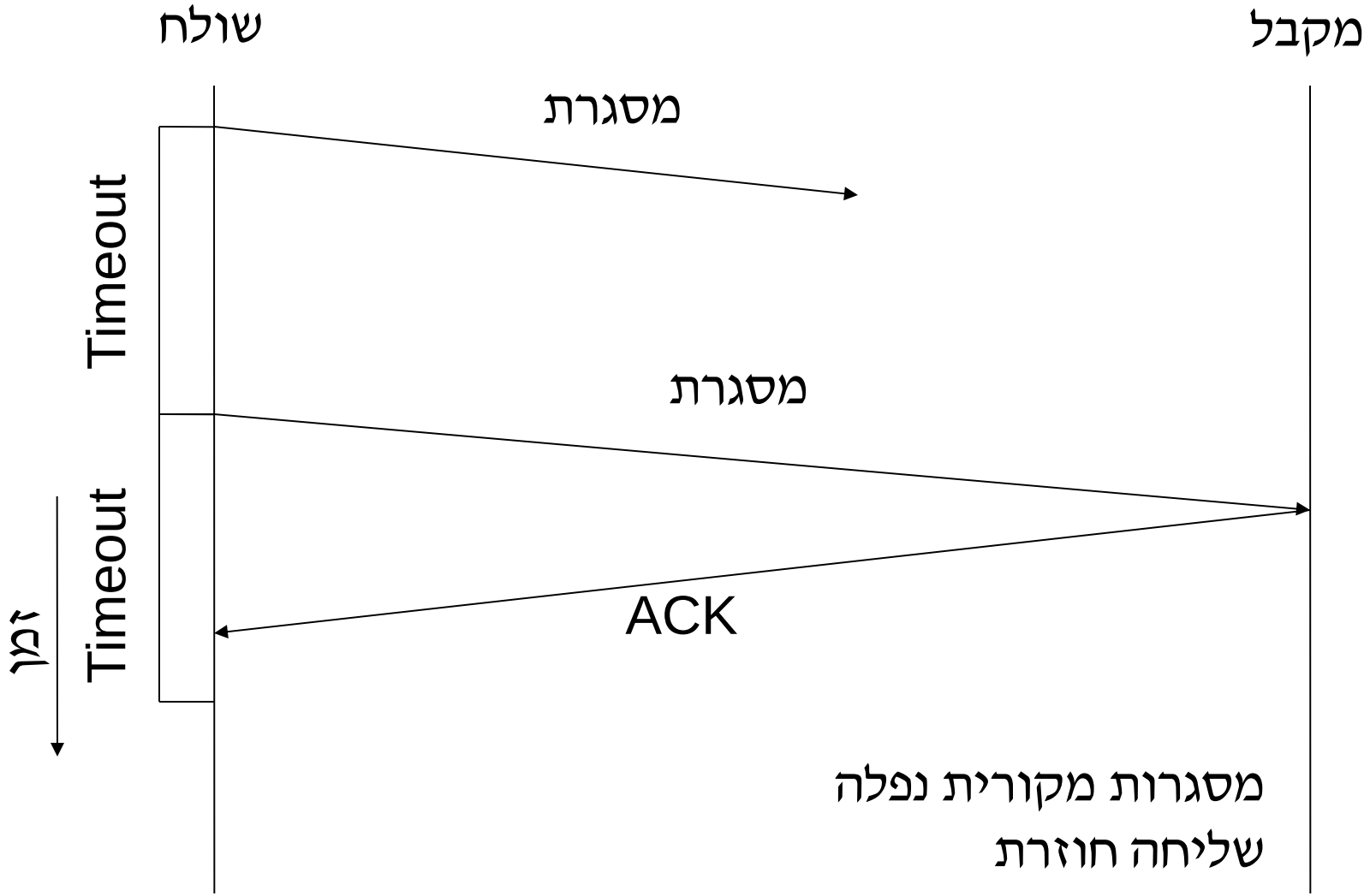
1. אחרי שליחת מסגרת אחת, השולח יחכה ל-ACK
2. אם ה-ACK לא מגיע, השולח ישלח שוב



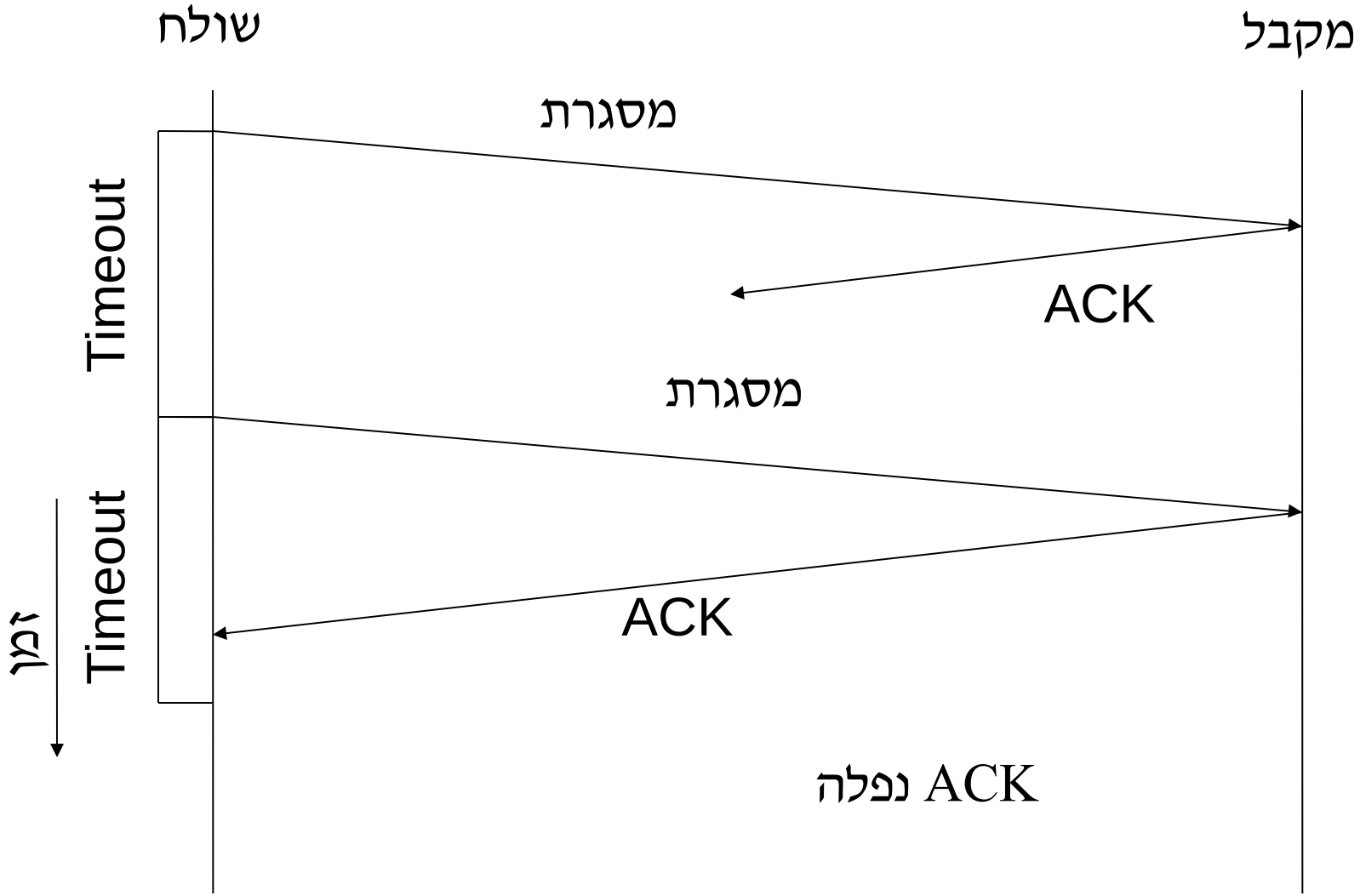
עצור וחכה: מקרה 1



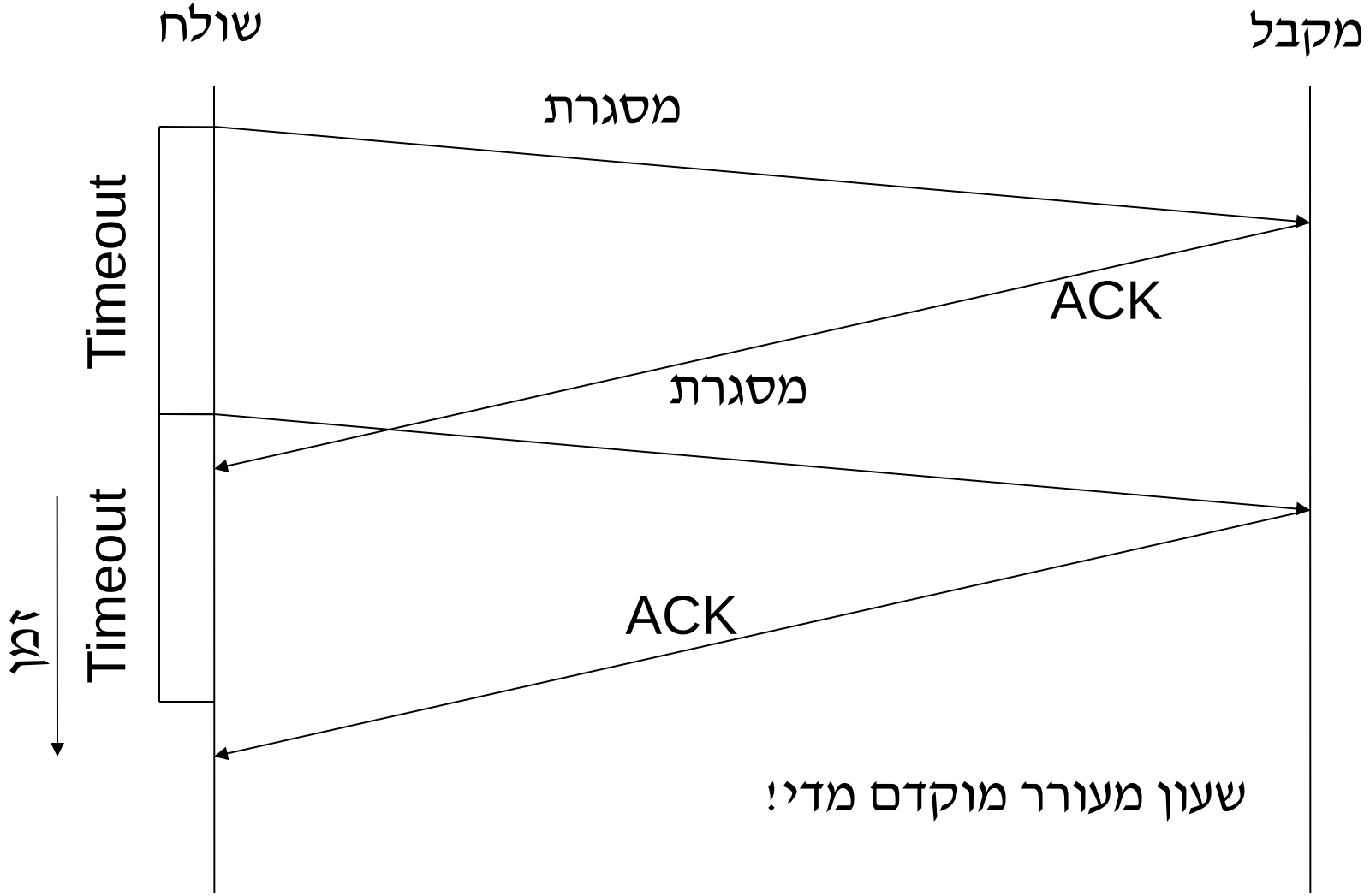
עצור וחכה: מקרה 2



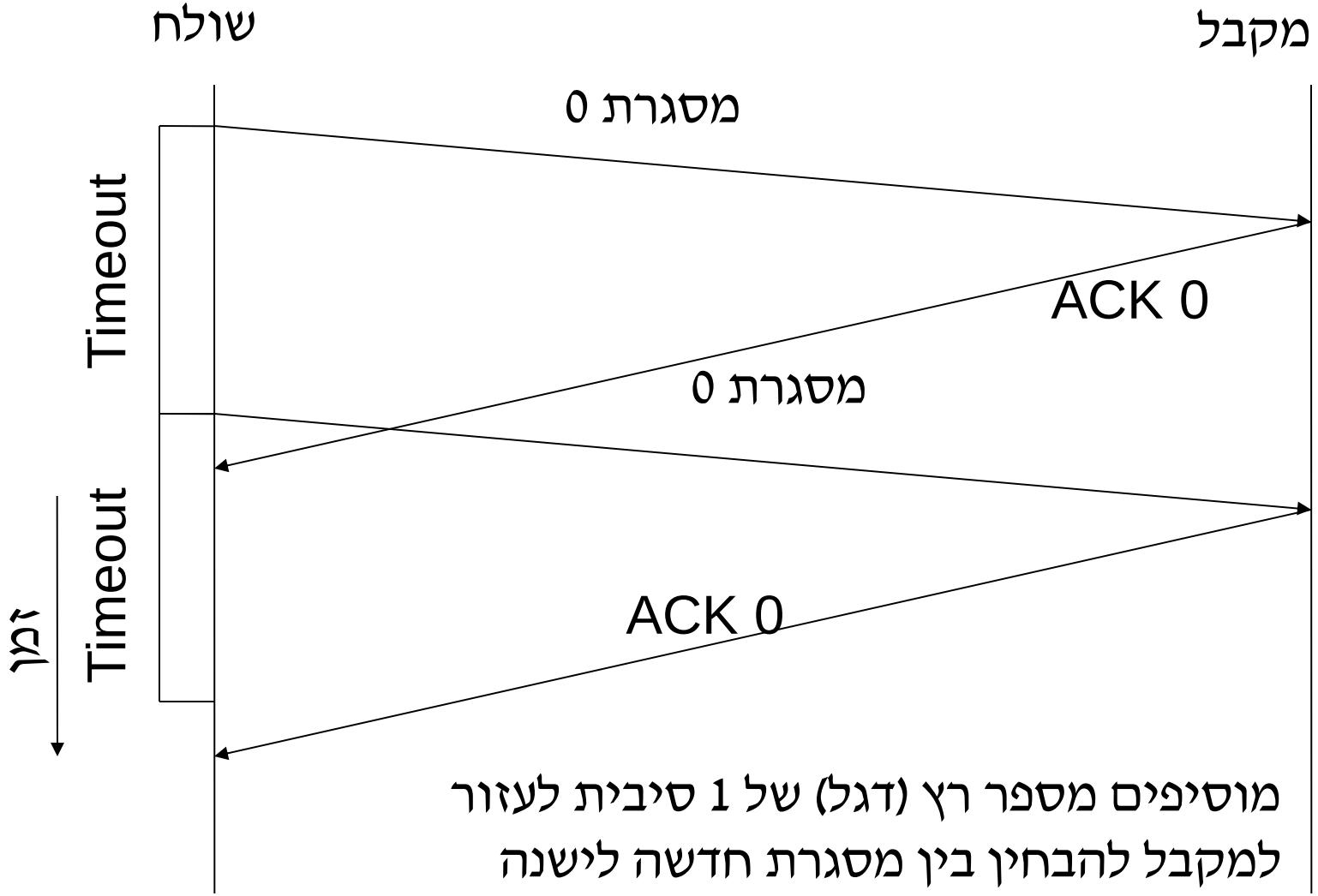
עצור וחכה: מקרה 3

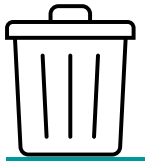


עצור וחכה: מקרה 4



Sequence numbers : מספרים רצים



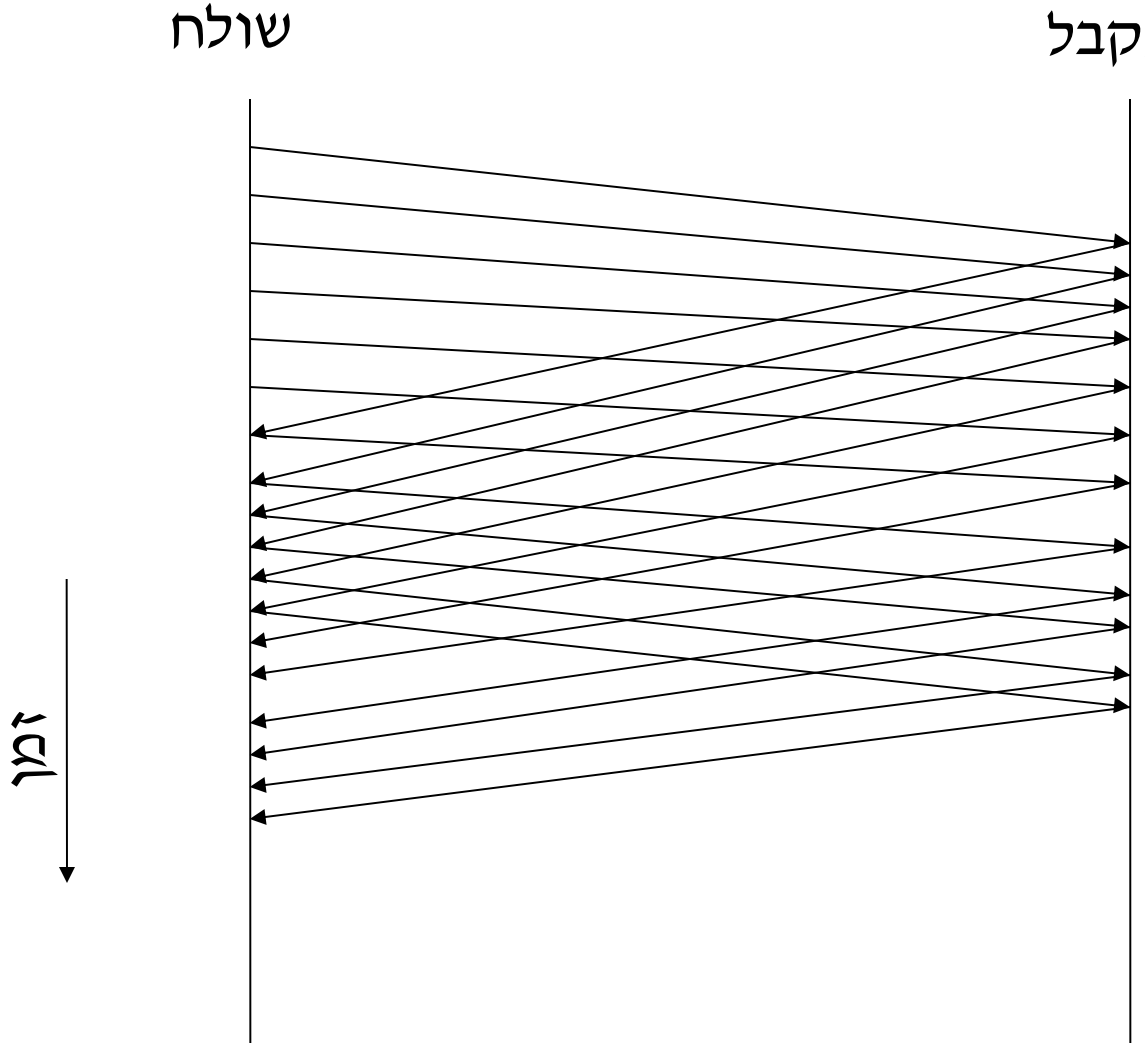


עצור וחכה

- שימוש לא יעיל בקיבולת החיבור
- שולח 1 מסגרת כל RTT

- דוגמה
  - חיבור 10Mbps
  - RTT 16ms
  - קיבולת Delay x Bandwidth שווה 20KB בערך
  - עצור וחכה עם מסגרות 1KB גורם ל-5% ניצול של החיבור

פתרון יותר יעיל



$$N = \frac{\frac{RTT}{2} \times BW}{frame\ size}$$

השולח מוכן לשלוח מסגרת $N + 1^{st}$ כאשר ה-ACK הראשון מגיע

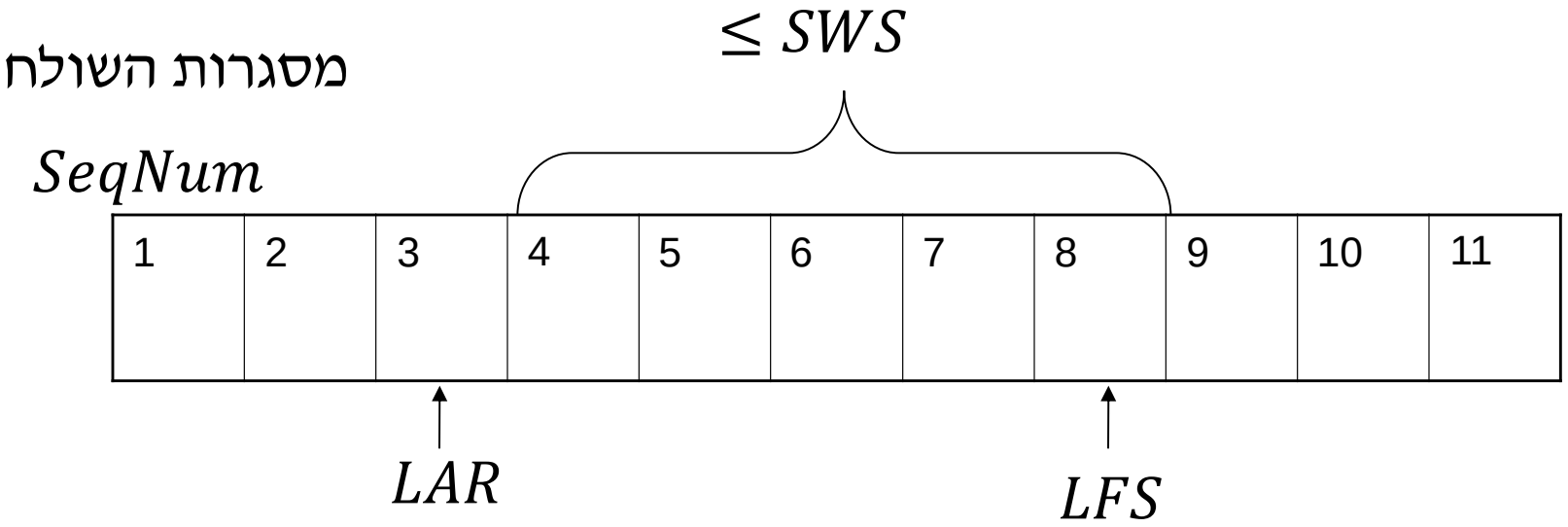
עד כה

- שגיאות
  - גילוי שגיאות CRC
  - תיקון שגיאות
- שליחה אמינה ARQ
  - עצור וחכה
  - חלון הזזה

אלגוריתם לחלון הזזה

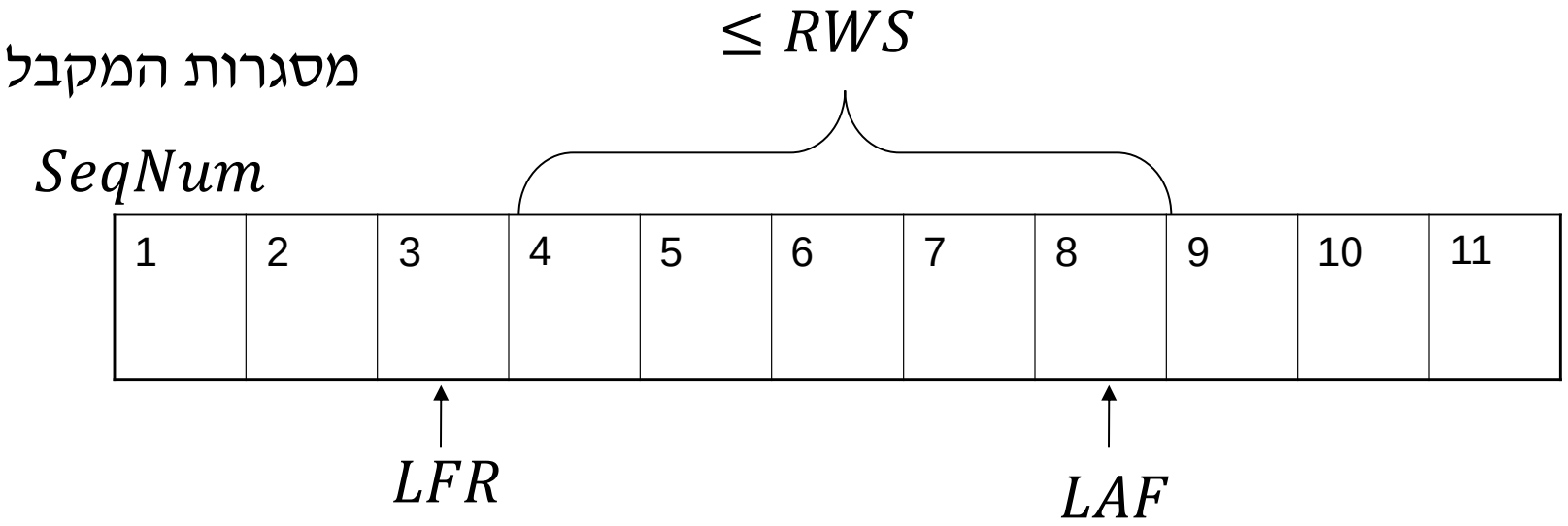
- השולח מדביק מספר מנה על כל מסגרת: $SeqNum$
  - נניח כעת ש-SeqNum יכול גדול עד אינסוף
- גודל חלון השליחה Send Window Size (SWS)
  - גבול עליון לכמות המסגרות השולח מוכן לשלוח ללא אישור קבלה
- אישור אחרון שהתקבל
  Last ACK Received (LAR)
  - מספר הרץ של מנת ה-ACK האחרון שהגיעה
- מסגרת אחרונה שנשלחה
  Last Frame Sent (LFS)

אלגוריתם השולח



- $LFS - LAR \leq SWS$
- משייך שעון מעורר (timeout) לכל מסגרת שנשלחה
  - שולחת מחדש אם לא הגיע מנת ACK לפני שהשעון יפוג
- כאשר מנת ACK מגיעה, מקדמים את LAR
  - כלומר, ניתן לשלוח עוד מנות

אלגוריתם המקבל



- גודל חלון הקבלה Receive Window Size (RWS)
  - מספר המנות שהגיעו לא לפי הסדר שהמקבל יקבל
- מנה מקובלת מרבית Largest Acceptable Frame (LAF)
- מנה הגבוה ביותר שהגיעה Largest Frame Received (LFR)
- $LAF - LFR \leq RWS$

אלגוריתם המקבל

- כאשר מסגרת עם מספר $SeqNum$ מגיעה
  - אם ($SeqNum \leq LFR$) או ($SeqNum > LAF$) זורקים לפח
  - אחרת, המנה מתקבלת
- נגדיר: $SeqNumToAck$ מספר מנה לאשר
  - מספר הרץ הגבוה ביותר כך שכל המסגרות שקדמו לה התקבלו
- השולח שולח ACK($SeqNumToAck$)
- $LFR = SeqNumToAck$
- $LAF = LFR + RWS$

סיום

- שגיאות
  - גילוי שגיאות CRC
  - תיקון שגיאות
- שליחה אמינה ARQ
  - עצור וחכה
  - חלון הזזה