

פרק 8

פרוטוקול TCP

תכונות הפרוטוקול

- ❖ קשר דו כיווני אמין וסדור בין תהליכים במחשבים שונים
- ❖ העברה מהירה ככל האפשר של מידע
- ❖ מניעת עומס מיותר על הרשת, עומס שעלול לנבוע ממשלוח מנות שיאבדו וייווצר צורך לשלוח שוב
- ❖ יצירת ופירוק קשר באופן מפורש ומוסכם על ידי שני הצדדים
- ❖ תכונות מתקדמות שלא נדון בהן, כגון הודעות דחופות

בזכות התכונות הללו, פרוטוקול ההעברה הנפוץ ביותר

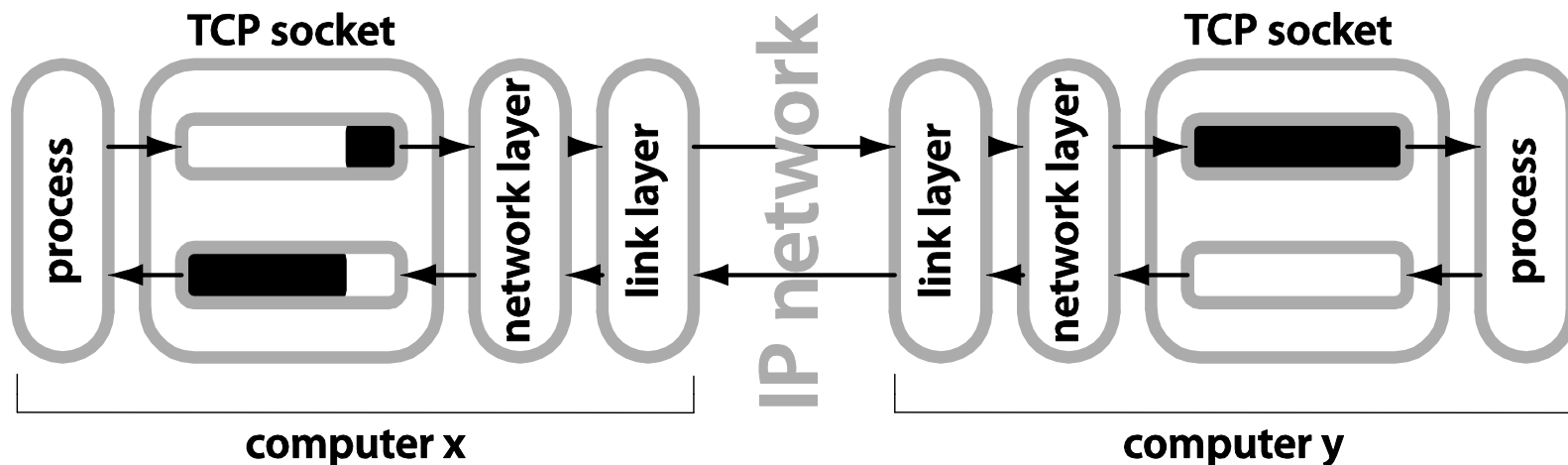
המימוש מורכב, מכיון ש...

- ❖ פרוטוקול הרשת IP אינו אמין ואינו סדור, גם אם פרוטוקולי המיקשר אמינים וסדורים
- ❖ נתב שמקבל מנות בקצב גבוהה מזה שהוא יכול לטפל בהן (למשל בניתוב מערוץ מהיר לאיטי) פשוט מוחק אותן
- ❖ IP שולח מנות בדידות וכל מנה מנותבת בנפרד; כאשר טבלאות הניתוב משתנות, מסלולים משתנים, ומנה מאוחרת ברצף המידע עשויה להקדים מנה מוקדמת

נגזרות מהדרישות

- ❖ מספור סודר על מנות על מנת שניתן יהיה לשחזר בצד המקבל את הרצף לפי סדר
- ❖ משלוח חוזר של מנות שאבדו
- ❖ אישורי קבלה על מנת שניתן יהיה לזהות מנות שלא הגיעו אחרי פרק זמן סביר ולשלחן שוב
 - מכיון שאי אפשר לדעת האם מנה אבדה או רק מתעכבת הרבה, מנות עלולות להגיע יותר מפעם אחת!
- ❖ חוצצים למשלוח מנות באופן יעיל

הוצעים ב-TCP



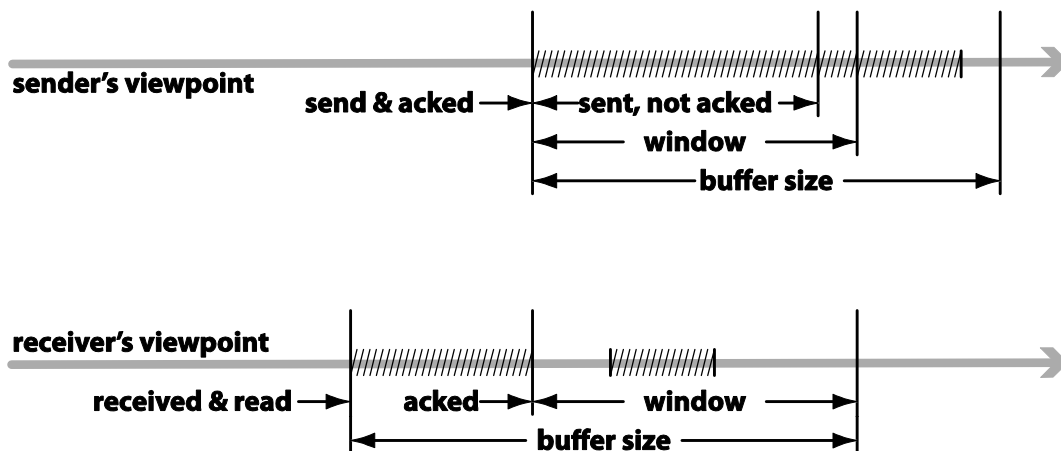
❖ החוצץ בצד המקבל מאפשר לשולח לשלוח כמות גדולה של מידע לפני שהוא מקבל אישור למנה כלשהי; בהעדר חוצץ כזה

- השולח יכול היה לשלוח מידע שאולי אין היכן לאחסן: עומס ללא צורך
- לשלוח מנה רק כשהקודמת מאושרת: לא מנצל את רוחב הפס של הרשת

❖ החוצץ הזה מאפשר גם לקבל הודעות שהתהליך המקבל עסוק

❖ החוצץ בצד השולח מאפשר לתהליך לרוץ בשטף ולנצל את הרשת

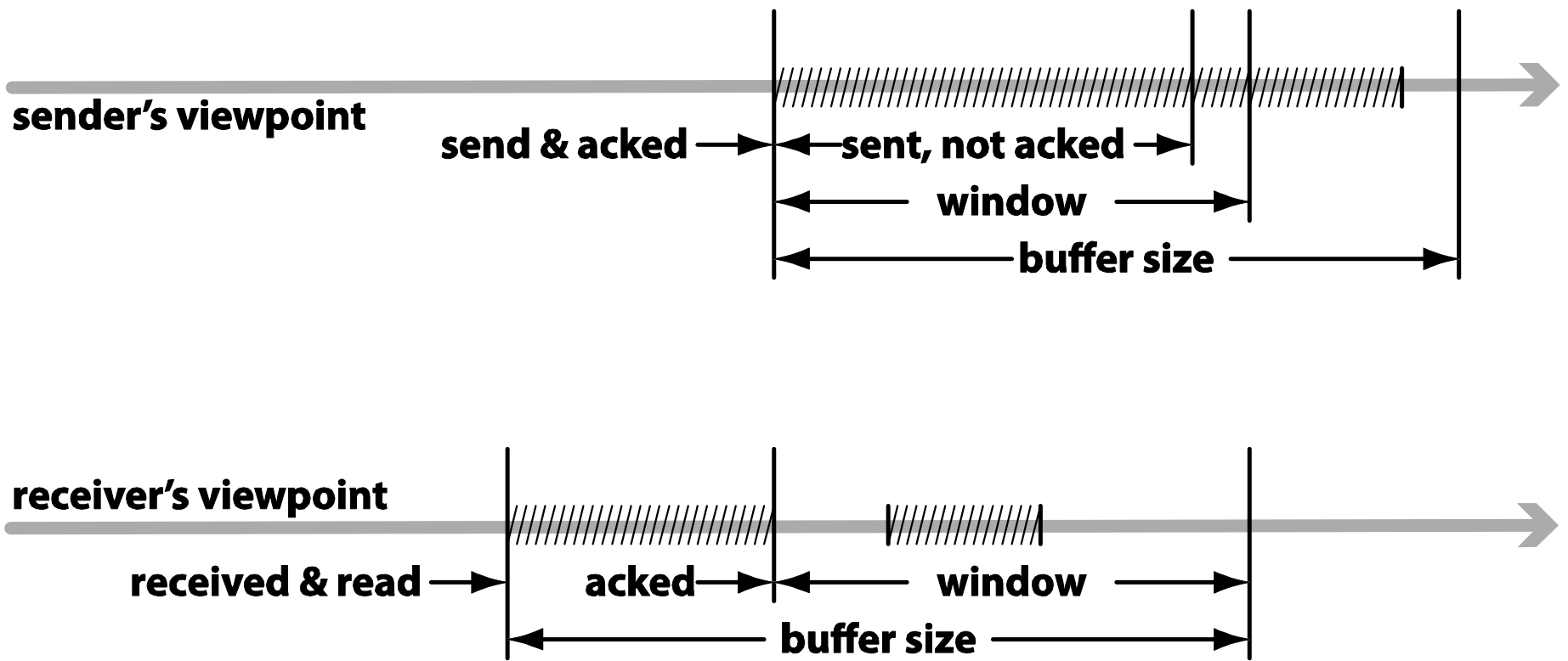
תחילת TCP



❖ לכל מנת מידע מצורפת תחילת עם השדות הבאים

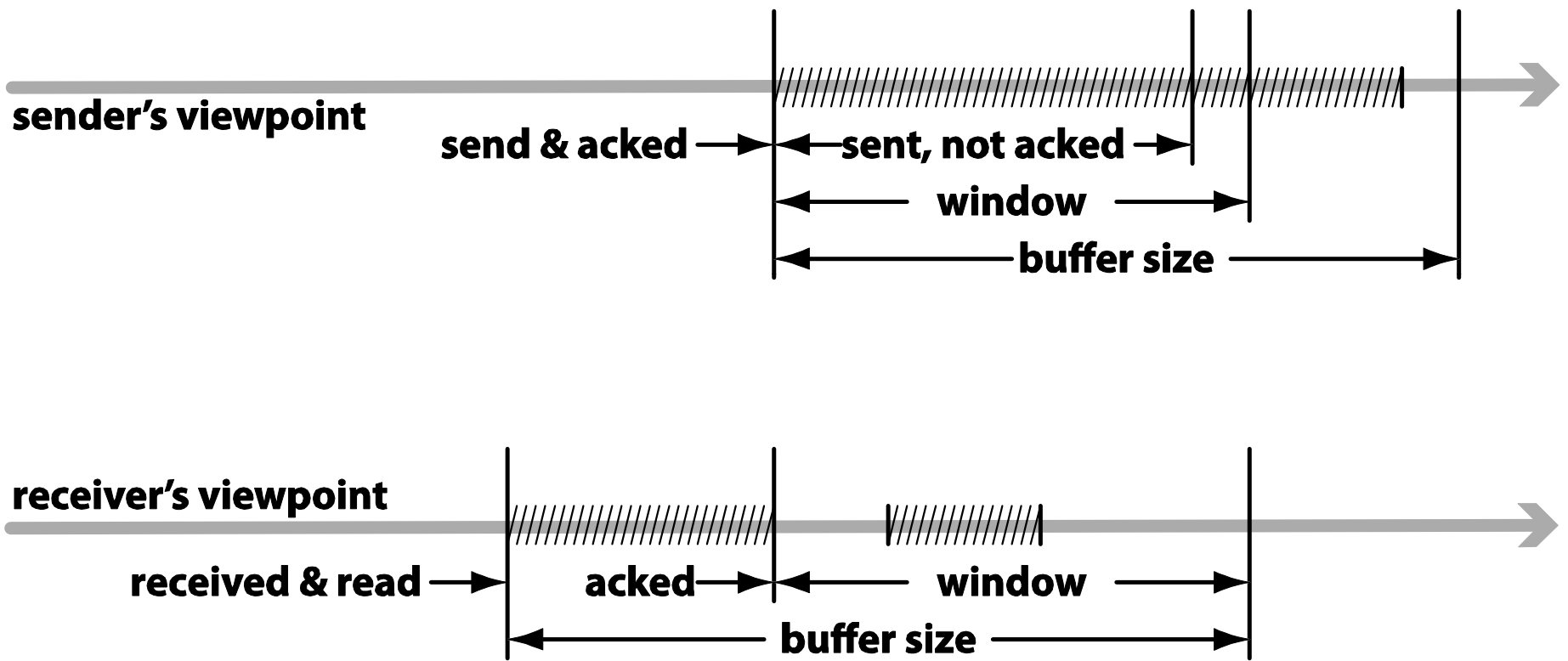
- כמות המידע שהמנה נושאת, בבתים, 0 הוא ערך מותר
- מיקום המידע ברצף מודולו 2^{32} ; השדה הזה מהווה את השדה הסודר
- מיקום הבית האחרון שנקלט ללא חורים על ידי הצד ששולח את המנה; אין דרך לאשר קבלת מנה שנושאת נתונים שנתונים קודמים להם חסרים
- גודל חלון מודיע לצד השני כמה בתים שלא אושרו מהמשך הרצף חוצץ הקבלה מסוגל להכיל; שדה של 16 סיביות, בבתים או יחידות גדולות יותר
- סכום בדיקה לגילוי שגיאות ושדות שמציינים מצבים מיוחדים

חלונות מחליקים



מצב פשוט: מצב הקשר נראה זהה לשני הצדדים (לא טיפוס)

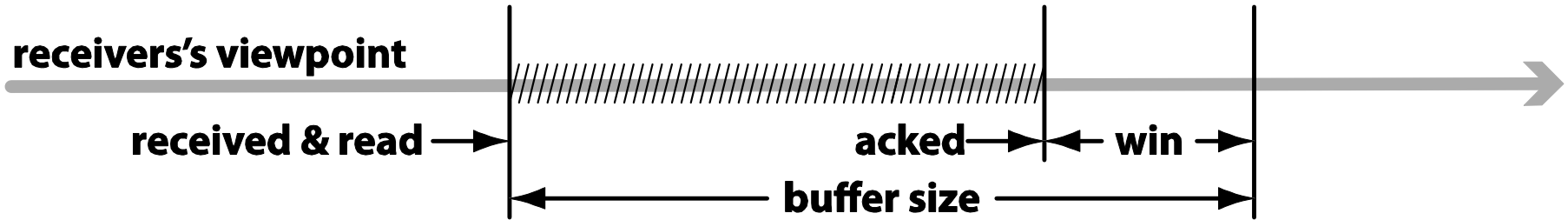
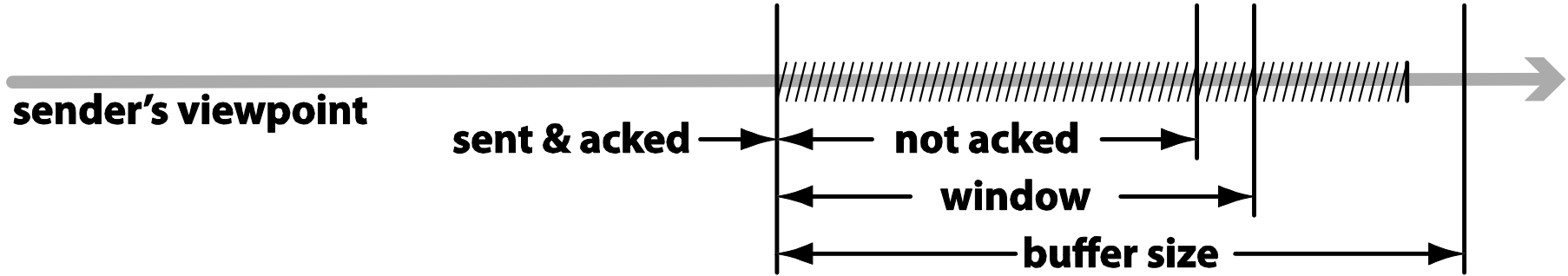
חלונות מחליקים



מצב פשוט: מצב הקשר נראה זהה לשני הצדדים (לא טיפוס)

מה קורה כאשר מתקבלת מנה שממלאה את החור?

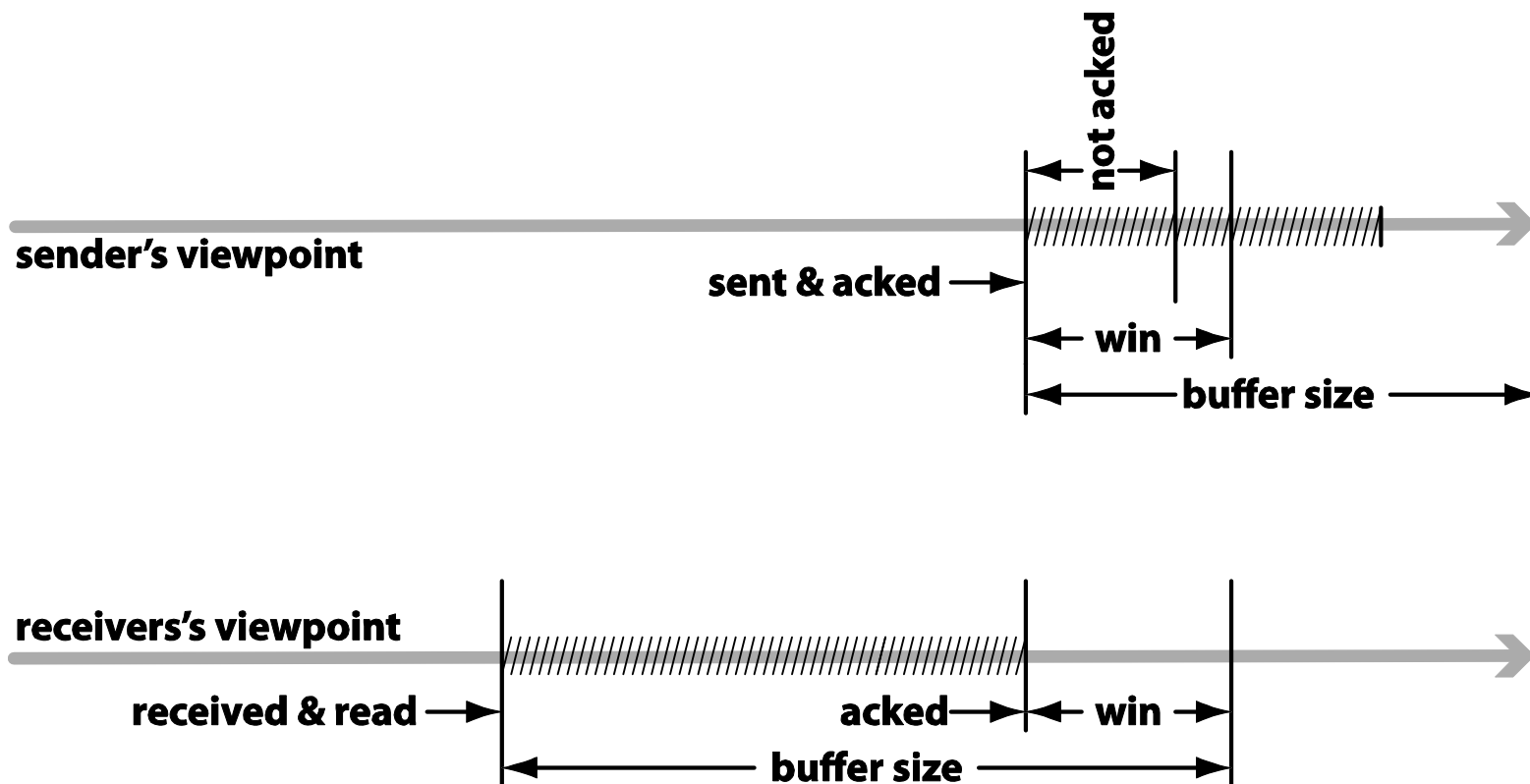
החזר התמלא



המקבל שולח אישור (עדיין לא התקבל)

מה יקרה כאשר השולח יקבל את האישור?

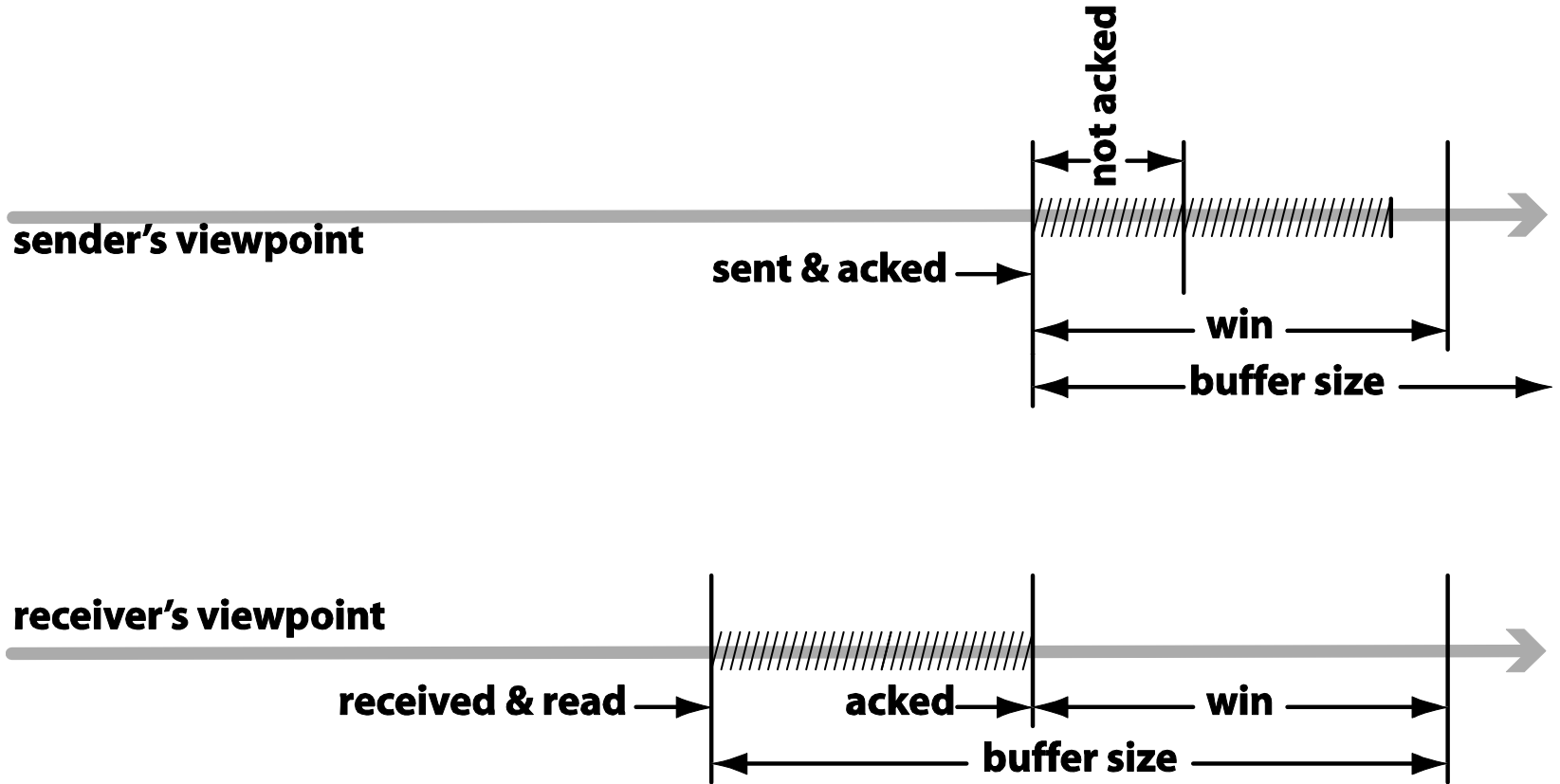
האישור התקבל



כאשר המקבל יקרא מהחוצץ, הוא ישלח עדכון חלון

מה יקרה כאשר השולח יקבל את עדכון החלון?

עדכון החלון התקבל



מה יקרה כאשר השולח יעביר נתונים נוספים לשקע?

אובדן מנות

❖ שירות יקיצה מזכיר לשולח לשלוח שוב מנות שלא אושרו (שירות שידור חוזר, retransmit)

- שליחת מנה כאשר השירות כבוי מפעילה אותו לנקודת זמן בעתיד
- כאשר הוא פוקע שולחים שוב את המנה הישנה ביותר שלא אושרה ומתזמנים יקיצה לזמן רחוק יותר
- פרק הזמן ליקיצה גדל פי קבוע בכל פעם, עם גג
- הגדלת פרק הזמן מיועדת למנוע שידור חוזר תכוף מדי בערוץ איטי
- כאשר מנה מאושרת, מתזמנים שוב יקיצה עבור המנה הישנה ביותר שעדיין לא אושרה

גם אישורים הולכים לאיבוד

- ❖ התוצאה של אישור אבוד הוא חלון סגור
- ❖ שירות יקיצה בשם persist מזכיר לשולח לברר האם החלון נפתח
- ❖ השירות מופעל כאשר יש מידע לשלוח והחלון סגור (או קטן מדי)
- ❖ בזמן יקיצה, השולח שולח בית בודד אם החלון סגור וכמה שאפשר אם הוא פתוח, ומתזמן יקיצה נוספת
- ❖ כאשר המנה הזו מגיעה, המקבל מאשר אותה עם גודל חלון עדכני
- ❖ שירות היקיצה מכובה כל אימת שמגיע עדכון חלון שמאפשר לשלוח מנה שלמה
- ❖ המקבל מאשר גם מידע שכבר נתקבל ואושר, מחשש שהאישור המקורי אבד

אופטימיזציות 1:

סינדרום החלון האילי

- ❖ חלון כמעט סגור מלמד בדרך כלל שהחוצץ בצד המקבל מלא במידע שהתהליך המקבל לא קרא עדיין
- ❖ במצב כזה אין טעם לשלוח מידע נוסף: זה לא יועיל למקבל ומנות קטנות מעמיסות על המחשבים והרשת
- ❖ על מנת למנוע משלוח מנות קטנות:
 - המקבל אינו מכריז על חלון קטן (פחות ממנה סטנדרטית או פחות מרבע גודל החוצץ שלו) אלא על 0
 - השולח משהה משלוח מנות קטנות עד ליקיצת `persist`
- ❖ התנהגות השולח מעכבת תקשורת ללא צורך כאשר למקבל חוצץ קטן מאוד: כדי למנוע זאת, שולחים מייד אם גודל החלון לפחות חצי מהחלון הגדול ביותר שהוכרז עד כה

אופטימיזציות 2:

האלגוריתם של Nagle

- ❖ השולח משהה משלוח מידע חדש שאינו ממלא מנה שלמה כאשר לא כל המנות שנשלחו אושרו
- ❖ כאשר נשלחות כמויות מידע גדולות, העיכוב במשלוח קטן ומונע עומס
- ❖ כאשר ערוץ התקשורת מהיר (רשת מקומית) ואישורים מתקבלים כמעט מייד, אין בדרך כלל עיכוב כלל
- ❖ האלגוריתם הזה פוגע בביצועים של יישומים אינטראקטיביים מסויימים וניתן לכבות אותו על ידי קריאה ל-`setsockopt`

אופטימיזציות 3:

השהיית אישורים ועדכוני חלון

❖ מנה שאינה נושאת נתונים אלא רק אישור ועדכון חלון מעוכבת
עד ש:

- נוצר צורך לשלוח מנה של נתונים, או
- חלון הקבלה גדל למנה שלמה, או
- חלפה חמישית שניה

❖ האופטימיזציה הזו מונעת משלוח מספר עצום של עדכוני חלון
כאשר התהליך הקורא מבקש נתונים מועטים בכל קריאת מערכת

❖ ההשהיה מוגבלת על מנת למנוע מנות כתוצאה מיקיצות
persist-ו retransmit בצד השני

אופטימיזציות 4:

מניעת עומס

- ❖ עומס על נתבים וקווי תקשורת גורם לאובדן מנות
- ❖ יקיצות `persist` ו-`restrasmit` מעידות בדרך כלל על עומס (ורק לעיתים רחוקות על השחתת מידע בתווך)
- ❖ השולח מקטין את קצב שליחת החבילות כאשר יקיצות כאלה גורמות לו לחשוב שהרשת עמוסה
- ❖ המנגנון מבוסס על התנהגות כאילו החלון קטן ממה שהוא באמת, אך לא נתאר אותו באופן מפורט

אופטימיזציות 5:

שליחה חוזרת מהירה

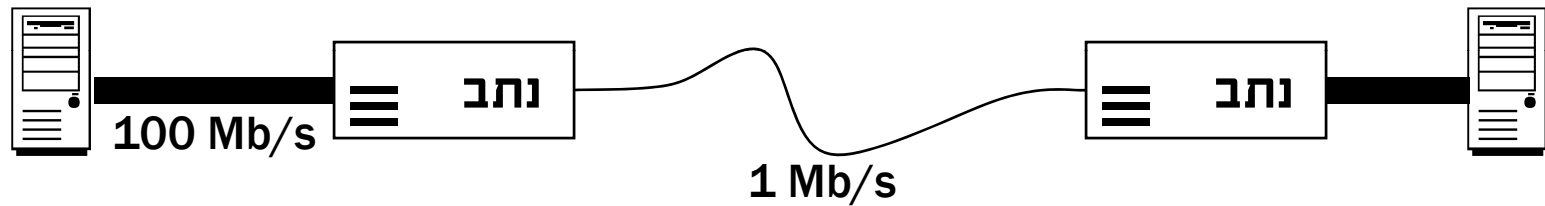
- ❖ השולח מסיק שמנה בודדת אבדה כאשר הוא מקבל שלושה אישורים זהים ברצף
- ❖ ההנחה כאן היא שמנה אבדה ומנות עוקבות התקבלו וגרמו למשלוח אישורים
- ❖ במקרה כזה השולח משגר שוב את המנה שכנראה אבדה ואינו מקטין את קצב השליחה (מהאופטימיזציה הקודמת)

אופטימיזציות 6:

קביעת גודל חוצים

- ❖ בהנחה שהתהליך הקורא לא מעכב מידע בחוצץ, תפקיד החוצץ הוא לאפשר לשולח לשלוח מידע לפני קבלת אישורים
- ❖ מה צריך להיות גודל החוצץ? קצב הערוץ כפול פעמיים ההשהיה
- ❖ למשל,
 - ערוץ לווני שמנה חוצה ברבע שניה ברוחב פס של 100 Mb/s
 - אישור למנה מגיע חצי שניה אחרי שליחתה
 - השולח צריך לשלוח 50 Mb שהם 6 MB לפני קבלת אישור
 - מאותו רגע הוא מקבל אישורים בקצב השליחה
 - החוצץ צריך להיות בגודל 6 MB על מנת לנצל את רוחב הפס של הערוץ
- ❖ חוצץ של 64 KB מאפשר לשלוח רק במאית מרוחב הפס

קביעת גודל חוצים - חוצץ גדול מדי



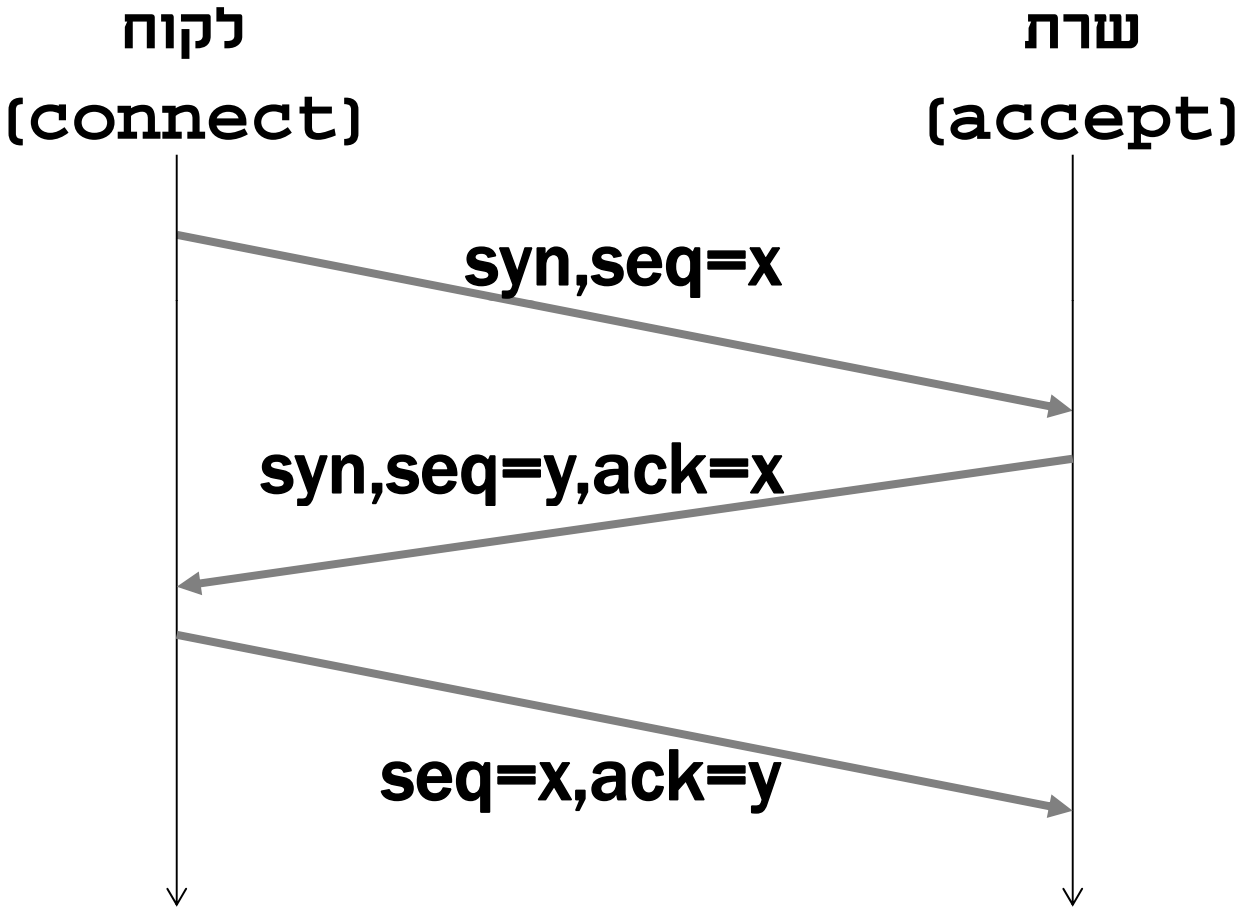
- ❖ מאידך, חוצץ קבלה גדול מדי מאפשר לשולח להציף נתבים במנות שהנתב אינו יכול להעביר
- ❖ החוצץ צריך להתאים בגודלו לרוחב הפס האפקטיבי של הערוץ מקצה לקצה ולא לרוחב הפס של מקטעים בדרך

אופטימיזציות 7:

התחלה איטית

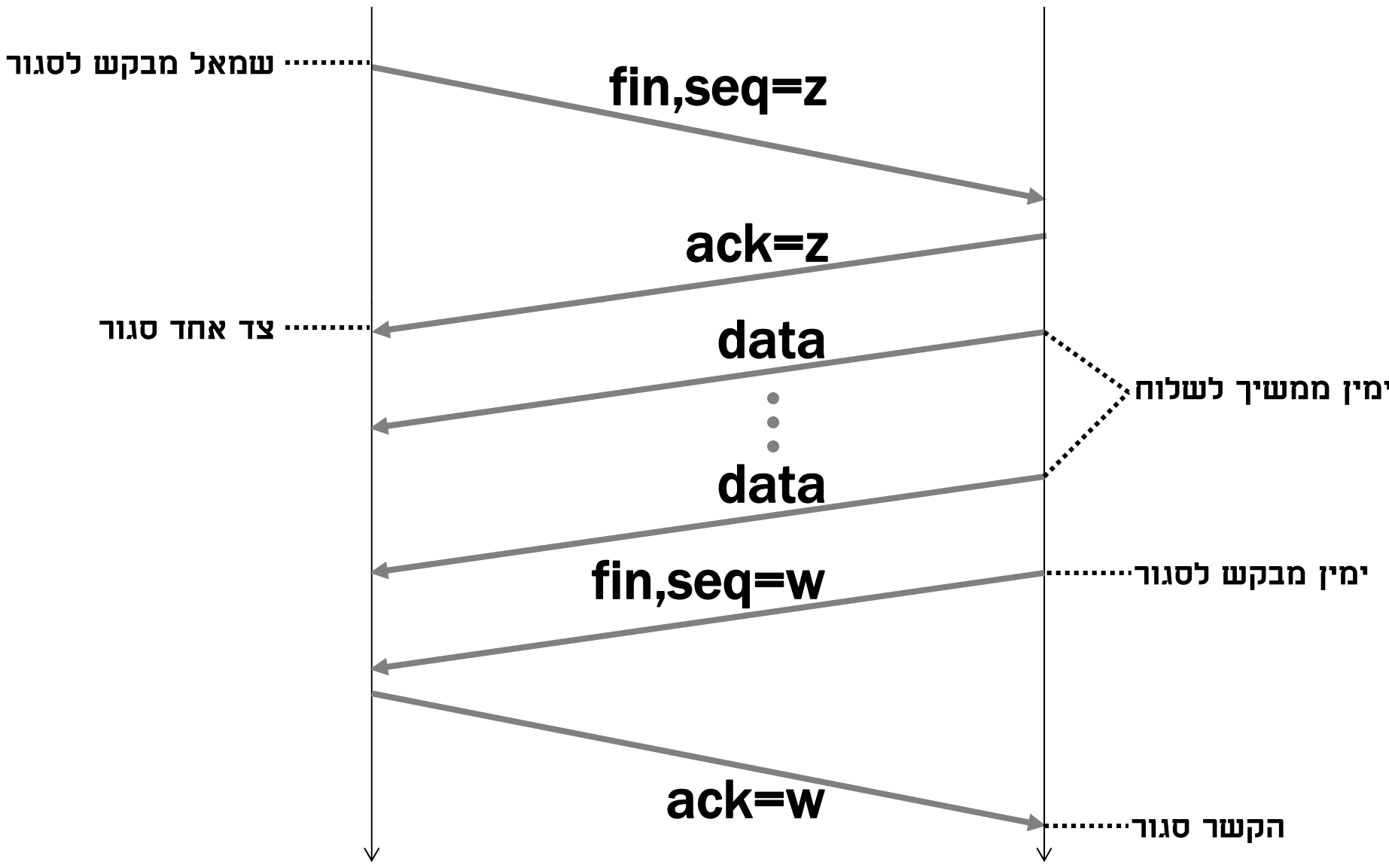
- ❖ במצב יציב של הקשר מנה נשלחת כל אימת שמגיע אישור שפותח את החלון למנה נוספת
- ❖ בתחילת הקשר החלון פתוח לרווחה והשולח עלול לשלוח חלון מלא בבת אחת לנתב קרוב
- ❖ אם הנתב לא יכול לשלוח את המנות במהירות, חלקן יאבד
- ❖ על מנת למנוע את זה, השולח משתמש בחלון עומס שמגביל את כמות המידע שנשלח ולא אושר
- ❖ חלון העומס מתחיל במנה אחת וגדל באחת כל אימת שמתקבל אישור
- ❖ קצב שליחת המנות גדל כך אקספוננציאלית לקצב של הערוץ

יצירת קשר



הרצפים מתחילים ממיקום אקראי כדי להקשות על התחזות

פירוק קשר



מנות מקשר מפורק

- ❖ מנות שהתעכבו ברשת עלולות להגיע לאחר שהקשר פורק
- ❖ אם האישור ל-fin אובד, מנת fin נוסף תגיע בהמשך למחשב שאישר קבלת fin ושמבחינתו הקשר סגור
- ❖ אי לכך, מערכת ההפעלה שומרת על מבנה הנתונים שייצג את הקשר זמן מה לאחר סגירתו כדי שניתן יהיה לטפל במנות כאלו
- ❖ כדי שניתן יהיה לטפל במנות כאלו, מערכת ההפעלה אינה מתירה ליצור קשר נוסף עם אותם פרמטרים (כתובת ip ומספר שער מקומיים ומרוחקים) באותו פרק זמן
- ❖ מערכות הפעלה רבות אינן מאפשרות למחזר את הפרמטרים המקומיים; ניתן להורות להן למחזר על ידי קריאה ל-setsockopt

מנות מקשר שנשכח

- ❖ שמירה לזמן מה על מבנה הנתונים של הקשר לאחר סגירתו מאפשרת לטפל ברוב המנות שמגיעות אחרי סגירה
- ❖ אבל לעיתים מנות עלולות להגיע זמן רב לאחר סגירה
- ❖ בנוסף, תקלות במחשבים אחרים (או ניסיונות שיבוש מכוונים) עלולות לגרום לקבלת מנות של קשרים שלא היו קיימים מעולם
- ❖ מערכת ההפעלה עונה למנות כאלה במנת מיוחדת שמסומנת בסיבית reset
- ❖ קבלת מנה כזו צריכה לגרום לשולח להפסיק לשלוח מנות בקשר

מודעות לקיום קשר

- ❖ כאשר אין צורך להעביר מידע בקשר TCP, מנות אינן נשלחות כלל
- ❖ התנהגות זו גורמת לחוסר מודעות להתנתקות הקשר עקב נפילת אחד הצדדים ועקב התנתקות של הרשת
- ❖ ביישומים שבהם דרושה מודעות לקיום או אי-קיום הקשר, ניתן להורות למערכת ההפעלה לשלוח מנה מדי פעם לבדיקת קיום
- ❖ המנגנון מופעל באמצעות שירות היקיצה `keepalive`