



TEL AVIV UNIVERSITY

# מערכות הפעלה

מרצה אורח: סיון טולדו

סמסטר א' תשע"ב

## הרצאה 9

### מערכות קבצים (המשך)

### רשתות תקשורת ו-IP

# נפילות והתאוששות

- ❖ השגרות של מערכת הקבצים מניחות הנחות מסוימות אודות מבנה הנתונים על הדיסק; נכונות השגרות תלויה בקיום ההנחות
- ❖ נפילה פתאומית עלולה להשאיר את המערכת במצב לא תקין
- ❖ הנחות שחייבות להתקיים (הפרה היא חוסר עקביות חמור):
  - בלוק שהוא חלק מקובץ אינו מסומן כפנוי
  - בלוק אינו ממופה כחלק משני קבצים או יותר
  - אין מצביע במדריך ל-inode שמסומן כפנוי
  - ...
- ❖ הנחות פחות חשובות שניתן לתקן בזמן שהמערכת פועלת:
  - אין בלוק שאינו ממופה כחלק מקובץ וגם אינו מסומן כפנוי
  - כל inode שאין אליו מצביע, מסומן כפנוי

# גישות להתאוששות

- ❖ כתיבה זהירה
- ❖ עדכונים רכים
- ❖ כתיבה עצלה
- ❖ מערכות מתאוששות
- ❖ שימוש בזיכרון לא נדיף
- ❖ זיהוי הצורך בהתאוששות: הדלקת סיבית מיוחדת במערכת הקבצים מכריזה שהמערכת ירדה באופן מסודר והיא קונסיסטנטנטית. הכתיבה הראשונה למערכת קבצים היא כיבוי הסיבית. אם היא כבויה, דרושה התאוששות

# כתיבה זהירה

- ❖ פעולות שמשנות את מבנה הנתונים בדיסק מתבצעות מיידית (לפני שקריאת המערכת חוזרת) והבלוקים נכתבים לדיסק בסדר שמבטיח שלא ייווצר חוסר קונסיסטנטיות חמור
- ❖ למשל, בזמן מחיקת קובץ קודם ימחק המצביע ל-inode מהמדריך, אחר כך ימחקו ההצבעות לבלוקים מה-inode, ואז יסומנו ה-inode והבלוקים כפנויים
- ❖ הגישה משמשת את FAT, FFS, LFS, ועוד
- ❖ גישה זו משפיעה לרעה על הביצועים כאשר קבצים נוצרים ונמחקים באופן תכוף; פחות ב-LFS כי הכתיבות רצופות
- ❖ אחרי נפילה, תהליך התאוששות רץ ברקע ומתקן את הבעיות הלא חמורות (סימון בלוקים כפנויים וכדומה)

# כתיבה עצלה

- ❖ מבצעים שינויים על עותקים של בלוקים בזיכרון וכותבים לפי סדר שנוח מבחינת תזמון הדיסק; מתעלמים מבעיית הקונסיסטנטיות
- ❖ כאשר המערכת עולה אחרי נפילה, יש צורך להריץ תהליך התאוששות שמתקן את מבנה הנתונים
- ❖ לפני שהתהליך מסיים לא ניתן להשתמש במערכת הקבצים כיון שהנחות בסיסיות על מבנה הנתונים עלולות שלא להתקיים
- ❖ בשימוש בלינוקס בעבר (מערכת הקבצים ext2), גישה לא נפוצה במערכות הפעלה מסחריות
- ❖ ביצועים מצוינים, התאוששות איטית (עד שעות במערכות קבצים גדולות מאוד)

# מערכות מתאוששות

(journaling file system / recoverable file system)

- ❖ שימוש בטכנולוגיה של מסדי נתונים על מנת לוודא שפעולות שמשנות את מבנה הנתונים ודורשות שינוי במספר בלוקים מתבצעות במלואן או לא בכלל, גם אם מתרחשת נפילה
- ❖ כל פעולה כזו מוגדרת כ**תנועה** (transaction) וכל תנועה מקבלת מזהה
- ❖ כל שינוי בבלוק של מבנה הנתונים (לא בבלוק נתונים של קובץ!) נרשם בקובץ יומן מיוחד עם מזהה התנועה
- ❖ בסיום כל פעולה נרשמת רשומה מיוחדת ביומן עם מזהה התנועה
- ❖ מערכת הקבצים לא מבצעת את השינויים בבלוקים מייד, וגם לא כותבת לדיסק את רשומות היומן מייד, אבל היא מוודאת שבלוק נכתב לדיסק רק אחרי כל רשומות היומן שרלוונטיות אליו

# אחרי נפילה של מערכת מתאוששת

- ❖ מערכת הקבצים לא קונסיסטנטית ואי אפשר להשתמש בה לפני הרצת תהליך התאוששות (כמו בעצלות, לא כמו בזהירות)
- ❖ יתכן שיש בדיסק שינויים ששייכים לתנועות שאולי לא בוצעו במלואן ואולי אפילו לא נכתבו ליומן בדיסק במלואן
- ❖ יתכן שחסרים בדיסק שינויים ששייכים לתנועות שמתוארות ביומן בדיסק במלואן

# התאוששות מערכת מתאוששת

❖ קריאת היומן מהדיסק

❖ זיהוי כל התנועות שנכתבו בחלקן אבל לא במלואן ליומן

▪ בעזרת היומן, מחזירים את הבלוקים הרלוונטיים למצבם הקודם

❖ זיהוי כל התנועות שנכתבו ליומן במלואן אבל אולי לא בוצעו

מול הדיסק במלואן (חשוב לנסות למזער את מספרן)

▪ בעזרת רשומות היומן מבצעים שוב את השינויים בבלוקים הרלוונטיים

▪ יתכן שהשינויים כבר בוצעו; אי אפשר לדעת

▪ לכן, רשומות היומן חייבות לתאר עדכונים אידמפוטנטיים

❖ כלומר, כל רשומת יומן מאפשרת לבצע עדכון בדיסק וגם

להחזיר את המצב לקדמותו



# סיכום מערכות מתאוששות

- ❖ התאוששות מהירה בגלל שצריך בדרך כלל לבצע שוב או לבטל רק תנועות מהשניות האחרונות לפני הנפילה
- ❖ ביצועים מצוינים: מעט אילוצים על סדר הכתיבה (יומן לפני בלוק)
- ❖ מערכות מורכבות מאוד בגלל הצורך לאפשר זיהוי מהיר ומדויק של תנועות שצריך לבטל או לבצע שוב (זיהוי מקורב יאט את ההתאוששות), בגלל הצורך לכפות אילוצים, ובגלל הצורך להתמודד עם מצבים מורכבים, כמו נפילה בזמן התאוששות
- ❖ מערכות נפוצות מאוד במערכות הפעלה מודרניות כולל חלונות (NTFS), רוב הגרסאות המסחריות של יוניקס, וגרסאות חדשות של לינוקס (XFS, JFS, ext3, ext4, btrfs, ...)

# שימוש בזיכרון לא נדיף

- ❖ אחסון בלוקים של מבנה הנתונים (ואולי אף בלוקים של נתונים מקבצים) בזיכרון מיוחד לא נדיף במקום בזיכרון הראשי הנדיף
- ❖ לאחר נפילה פשוט כותבים את הבלוקים הללו לדיסק
- ❖ הזיכרון הזה צריך לשרוד לא רק תקלות חומרה (הפסקות חשמל למשל) אלא גם תקלות תוכנה שעלולות לכתוב לתוכו: עדיף להגן עליו מכתובה רוב הזמן ולשחרר את ההגנה רק בשגרות של מערכת הקבצים
- ❖ ניתן למימוש בעזרת זיכרון מגובה בסוללה או אל-פסק או בעזרת זיכרון מגנטי/הבזק מהיר
- ❖ בשימוש בשרתי קבצים ייעודיים; לא נפוץ במערכות הפעלה רגילות

# מערכות קבצים לזכרון הבזק

1. רציפות קריאה כמעט ואינה משנה - גישה אקראית מהירה
  2. כתיבה רצויה בבלוקים שלמים (לדוגמה 1MB)
  3. בלוקים נשחקים בכתיבה חוזרת (מתקלקלים אחרי אלפי עד מאות אלפי כתיבות) ולכן צריך לאזן את השימוש
- ❖ לכן: מערכת קבצים יעודית, לרוב מאסכולת log-structured
    - JFFS2, YAFFS, בלינוקס, exFAT בחלונות
  - ❖ בפועל, התקני זכרון הבזק לרוב כבר כוללים בקר המבצע מיפוי-מחדש פנימי, ולכן עבור מערכת ההפעלה, בעייה 3 לא רלוונטית ובעייה 2 לא פתירה! לכן רבים מוותרים ופשוט משתמשים במערכת מסורתית שתוכננה לדיסק קשיח מגנטי.
  - ❖ בעזרת פקודת TRIM, מערכת ההפעלה מספרת לבקר אודות שטח אכסון שאינו בשימוש.

## פרק 7

# רשתות תקשורת ו-IP

# הרצוי לעומת המצוי

- ❖ ברמת היישום דרוש קשר אמין בין תוכניות שרצות על מחשבים שונים שאינם מחוברים ישירות זה לזה, אלא בעקיפין דרך רשת תקשורת מורכבת (כמו האינטרנט)
- ❖ ברמת החומרה ניתן לתקשר רק בין מחשבים שמחוברים פיזית, התקשורת לא תמיד אמינה, והיא מתבצעת לפעמים במנות (חבילות מידע בדידות בגודל קבוע או מוגבל)

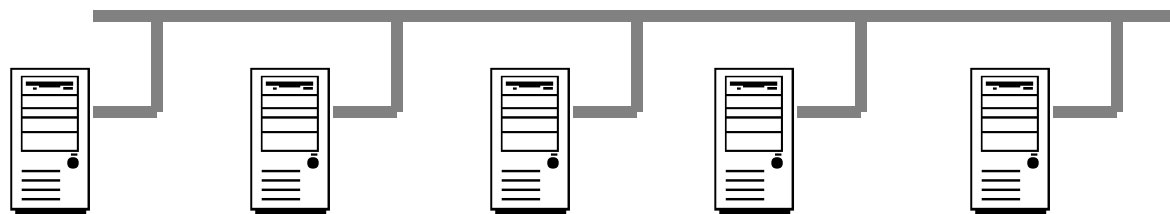
# על הפער מגשרים בשכבות

- ❖ השכבה הפיזית: המאפיינים הפיזיים של ערוץ תקשורת (רמות מתח, סוג מחבר, מספר חוטים, תדרים, וכדומה)
- ❖ שכבת המיקשר מגדירה פרוטוקולים להעברת מנות מידע בין מחשבים שמחוברים פיזית בתווך
  - פרוטוקול מגדיר את מבנה המנה, שיטת מיעון, ועוד
- ❖ שכבת הרשת מגדירה פרוטוקולים לניתוב מנות בין מחשבים ברשת שאינם מחוברים בהכרח פיזית זה לזה
- ❖ שכבת ההעברה מגדירה פרוטוקולים לתקשורת בין תהליכים במחשבים שונים
- ❖ שכבת היישום מגדירה פרוטוקולים ליישומים ספציפיים, כמו העברת קבצים (FTP ו-HTTP), דואר אלקטרוני (SMTP), ועוד

# פרוטוקולי האינטרנט

FTP	HTTP	SMTP	יישום
TCP		UDP	העברה
IP			רשת
אתרנט		PPP	מיקשר
אתרנט		מודם	פיזית

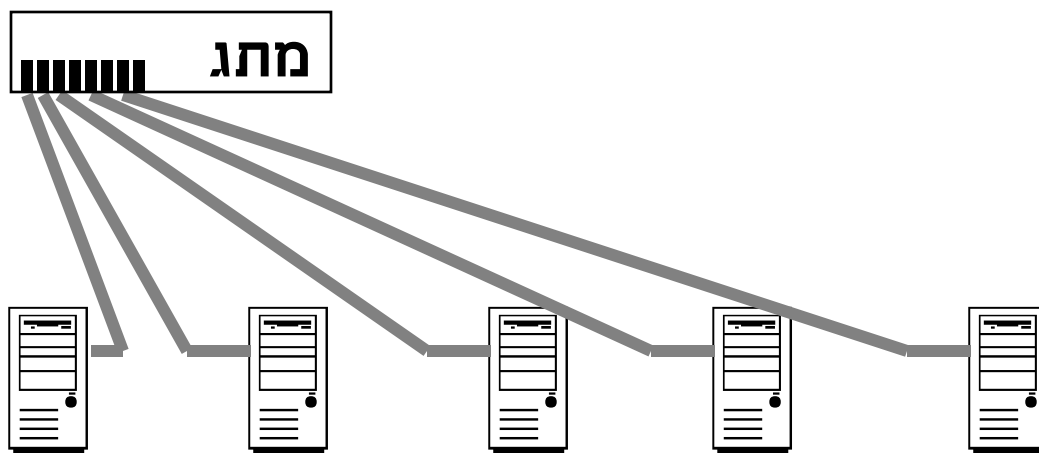
# השכבה הפיזית ושכבת המיקשר: אתרנט



- ❖ מקטע מחבר מספר מחשבים בבניין יחיד (מגבלת מרחק)
- ❖ לכל מחשב במקטע יש כתובת אתרנט
- ❖ כל מחשב יכול לתקשר עם כל אחד מהמחשבים האחרים במקטע
- ❖ מנגנון לגילוי שגיאות
- ❖ ניתן לקלוט רק אם מחשב אחד בלבד משדר
- ❖ במקור קצב של 10 Mb/s עם חיווט מסורבל, כיום כ"כ Mb/s 100 או 1GB/s עם חיווט נוח (דומה לחוטי טלפון); רוחב פס יותר גדול בשרתים



# מקטע אתרנט ממותג



- ❖ מבחינת המחשבים המחוברים אין הבדל בין מקטע ממותג ורגיל
- ❖ המתג מעביר כל מנה ליעדה
- ❖ מאפשר תקשורת בו-זמנית בקצב מלא בין זוגות מחשבים

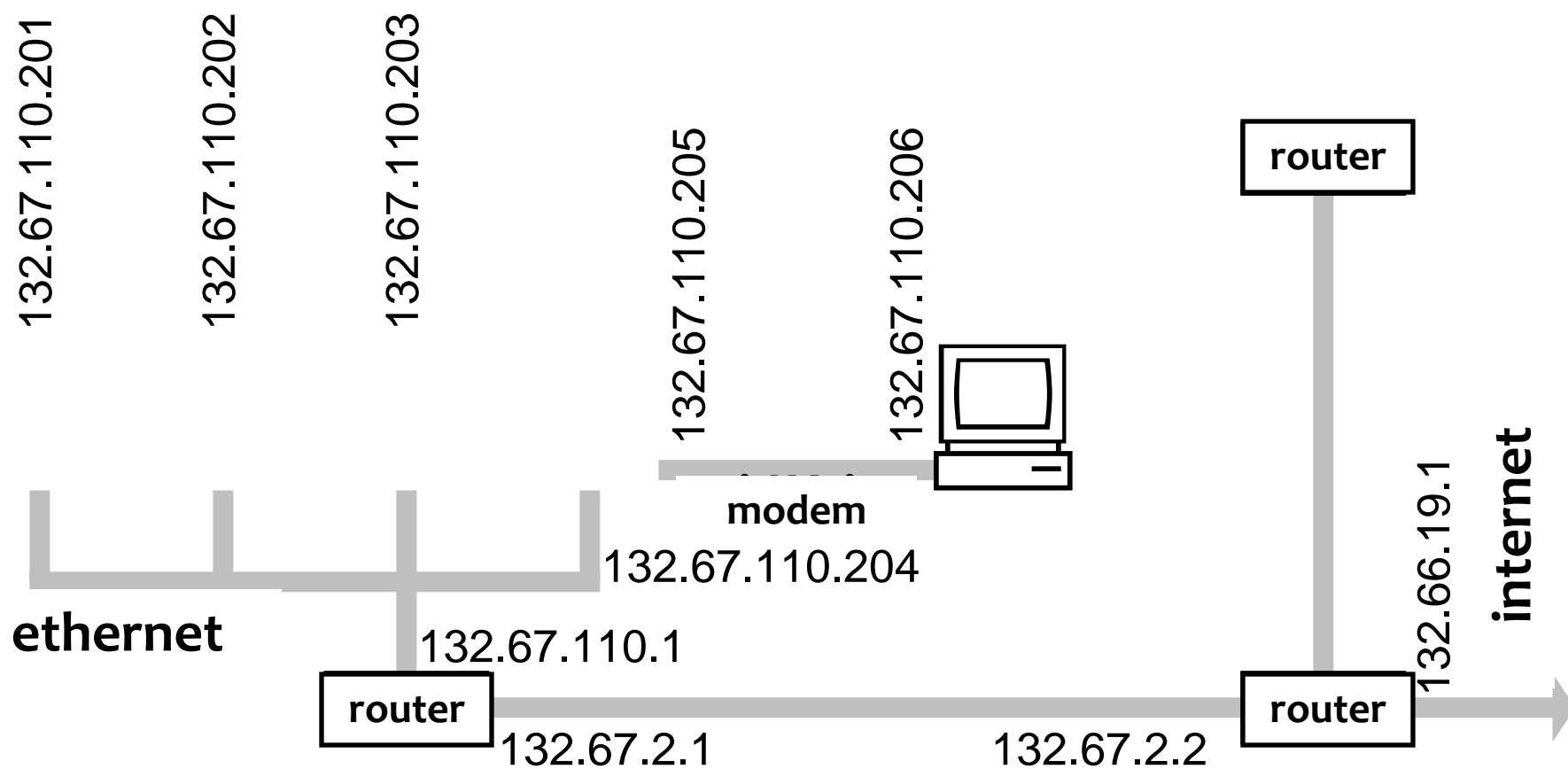
# אתרנט אלחוטי (WiFi)

- ❖ מבנה דומה לסגמנט אתרנט; כל התחנות (מחשבים) שומעות זו את זו
- ❖ קצת יותר מסובך כי אי אפשר להבטיח שמשדר יזהה תמיד התנגשות (תחנות נסתרות, hidden terminal problem)
- ❖ יותר קל לצותת ולכן אבטחת מידע ואבטחת גישה יותר חשובות
- ❖ מנגנוני אבטחה הראשונים (WEP) לא היו מספיק טובים וקל לפרוץ אותם
- ❖ קצבי נתונים החל מ-1MB/s ועד ל-104MB/s, תלוי באיכות הקליטה ובגיל החומרה

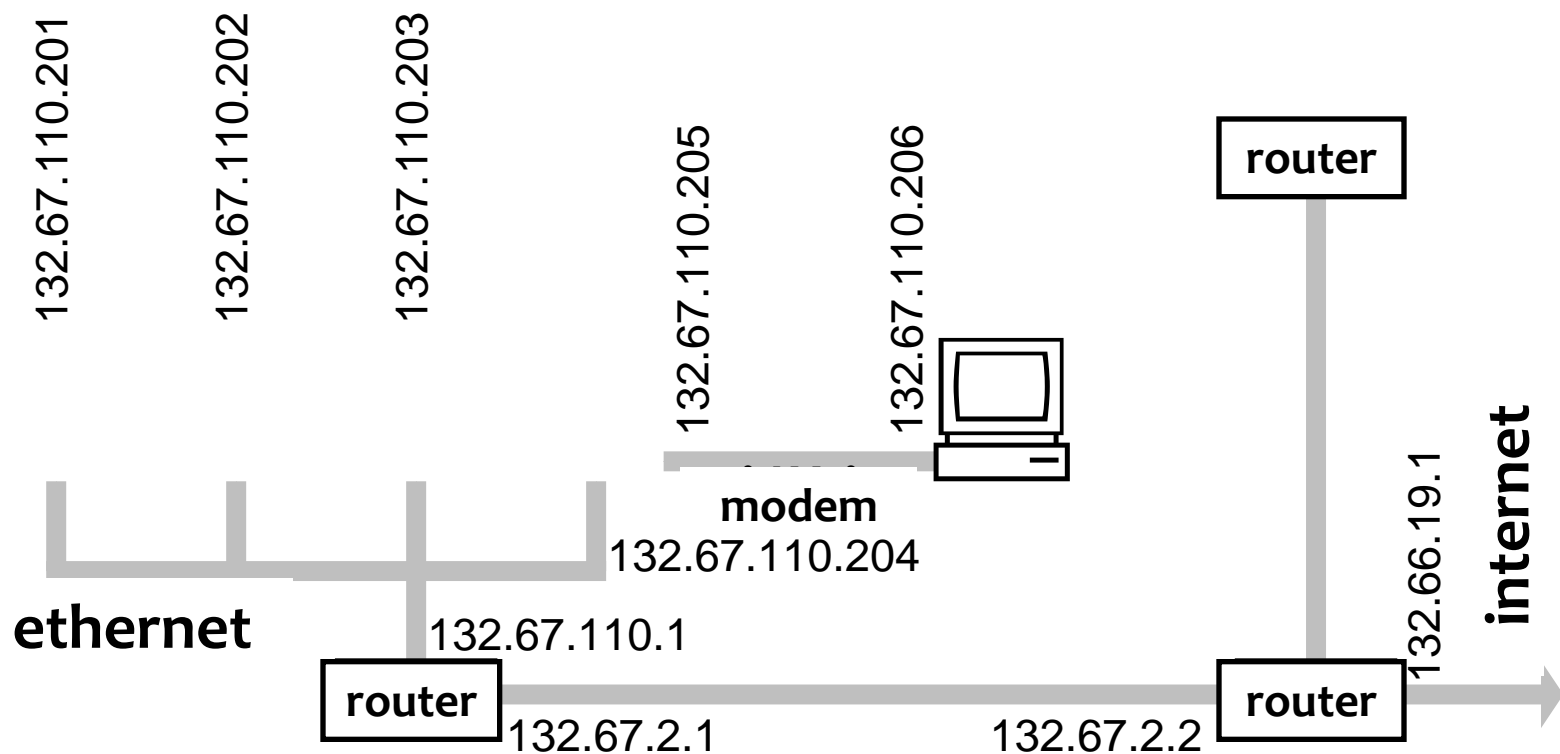
# רשתות תקשורת אחרות

- ❖ רשתות מקומיות דועכות: token-ring, decnet
- ❖ מודמים אנלוגיים: הגיעו עד 56Kb/s על קו טלפון אנלוגי
- ❖ מודמים לקווי טלפון סיפרתיים: ISDN (קו טלפון ספרתי בקצב של 2 x 64 Kb/s)
- ❖ ADSL (תקשורת ספרתית על קו טלפון אנלוגי, מאות Kb/s עד מספר Mb/s, לא סימטרי), מודמים לשימוש בתשתית הוידאו בכבלים, סיב אופטי עד הבית
- ❖ רשתות תקשורת לאזוריים נרחבים

# פרוטוקול הרשת IP



# פרוטוקול הרשת IP



- ❖ היפר-גרף שבו המחשבים הם צמתים, קשתות מחברות זוגות צמתים והיפר-קשתות מחברות מספר צמתים
- ❖ לקצוות של קשתות יש שמות בני 32 סיביות
- ❖ עקרונית שמות הם ייחודיים (מעשית לא)

# עקרונות ניתוב מנות ב-IP

- ❖ מנה מפלסת את דרכה ברשת מהמקור ליעד
- ❖ כל מחשב במסלול מחליט על הקשת הבאה שהמנה תעבור ועל המחשב הבא במסלול (באחד מקצות הקשת הבאה) על פי היעד שמצוין במנה; המסלול אינו נקבע מראש
- ❖ מחשב שמבצע החלטות ניתוב כאלה נקרא נתב; בדרך כלל אין לו תפקידים משמעותיים אחרים, אך כל מחשב כיום מסוגל לנתב
- ❖ נתב מקבל החלטות ניתוב בעזרת **טבלת ניתוב**
- ❖ כתובות IP מוקצות בצורה שמאפשרת להשתמש בטבלאות ניתוב קומפקטיות

# טבלאות ניתוב

- ❖ כל כניסה בטבלה מתארת החלטת ניתוב לקבוצה של כתובות
- ❖ קבוצת הכתובות מתוארת על ידי תחילית
  - כתובת יעד
  - מסיכה של 32 סיביות מכריזה איזה סיביות ביעד שייכות לתחילית
  - דוגמה: יעד 127.0.0.0 ומסיכה 255.0.0.0 מתארות את כל הכתובות שהסיביות הבכירות שלהן 01111111
- ❖ ההחלטה מיוצגת על ידי
  - שם של קשת (מנהל התקן והתקן בשכבת המיקשר)
  - כתובת IP של המחשב הבא במסלול, שצריך להיות מחובר לקשת
  - אם הכתובת חסרה, ניתן להגיע לכל היעדים בקבוצה ישירות דרך הקשת
- ❖ הכניסה עם התחילית הארוכה ביותר שמתאימה ליעד של המנה קובעת את הניתוב

# טבלאות ניתוב והקצאת כתובות IP

- ❖ טבלת ניתוב מכילה כללי ניתוב ויוצאים מן הכלל
  - כניסה בטבלה מייצגת כלל ניתוב עבור היעדים שהתחילית מתארת
  - כניסה עם תחילית יותר ארוכה (ותואמת) מייצגת יוצא מן הכלל
- ❖ השאיפה היא לטבלאות עם מעט כללים ומעט יוצאים מן הכלל
- ❖ השאיפה מתקיימת מבחינת נתב מסוים אם כל אחד מהמחשבים שמחוברים אליו הוא המחשב הבא הנכון עבור קבוצת יעדים שניתנת לתיאור במספר קטן של תחיליות
- ❖ לפיכך, כתובות IP מוקצות בצורה היררכית שמשקפת את מבנה הרשת



# דוגמה להקצאת כתובות IP

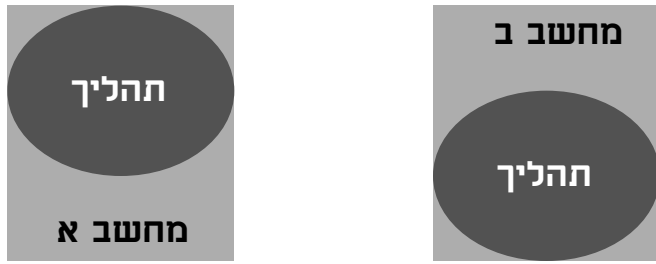
- ❖ הרשת האוניברסיטאית הישראלית קיבלה מהגוף המנהל של האינטרנט תחום כתובות מסויים
- ❖ הרשת האוניברסיטאית מקצה לכל מוסד תת-תחום מתוך הכתובות שהוקצו לה
- ❖ אם כל מוסד מקבל תחום כתובות עם תחילית אחת ואם לכל מוסד יש נתב אחד בכניסה לרשת שלו, אז כל היעדים של מוסד מיוצגים בטבלאות הניתוב במוסדות אחרים בכניסה אחת בלבד
- ❖ כל מוסד מקצה תתי-תחומים קטנים יותר לפקולטות ובתי ספר, שמקצים כתובות למחשבים בודדים

# תחזוקת טבלאות הניתוב

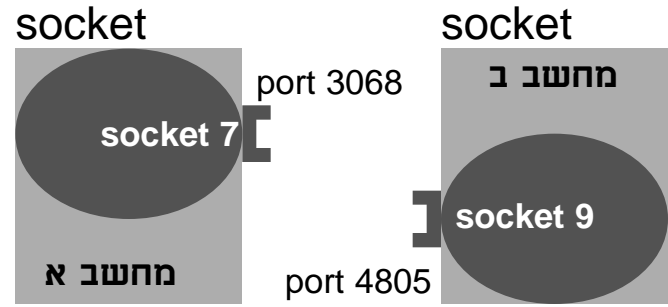
- ❖ במחשבים אישיים וביתיים תחזוקה ידנית (בעזרת הפקודה route ותסריטי אתחול) ועדכון הטבלה בזמן התחברות לרשת (חלק מפרוטוקולים כמו DHCP ו-PPP שמקצים כתובות דינמית)
- ❖ בנתבים: תחזוקה ידנית או השתתפות באלגוריתמים מבוזרים לקביעת טבלאות ניתוב (על ידי מציאת מסלולים קצרים, למשל)
- ❖ טבלאות הניתוב צריכות להתעדכן כתוצאה משינויים ברשת כמו נפילה של קווי תקשורת
- ❖ בכל מקרה, חשוב שטבלאות הניתוב לא ייצרו מעגלי ניתוב שמנות עלולות להסתחרר בהם; הסתחררות כזו מבזבזת משאבים ומונעת ניתוב מנות ליעדן

# מבוא לשכבת ההעברה: שקעים

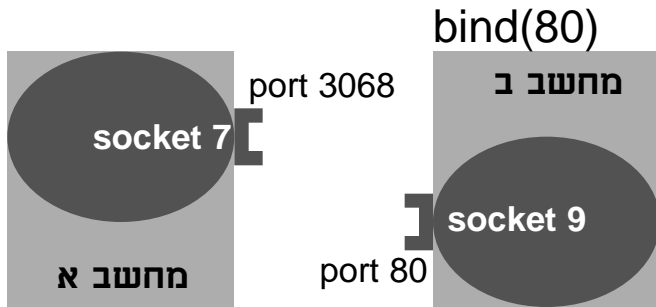
- ❖ כתובת IP מציינת מחשב; על מנת ליצור קשר בין תהליכים צריך כתובת לתהליך
- ❖ תהליך יכול ליצור שקע, עצם שניתן לשלוח ולקבל דרכו מידע
- ❖ שקע חדש מקבל כתובת שרירותית, אבל ניתן לקשור אותו לכתובת מסוימת; הכתובת נקראת **שער**
- ❖ בפרוטוקול UDP ניתן לשלוח מנדעים (מעין מברקים) לשער מסוים בכתובת IP מסויימת
- ❖ בפרוטוקול TCP ניתן לחבר שני שקעים בקשר קבוע ואמין



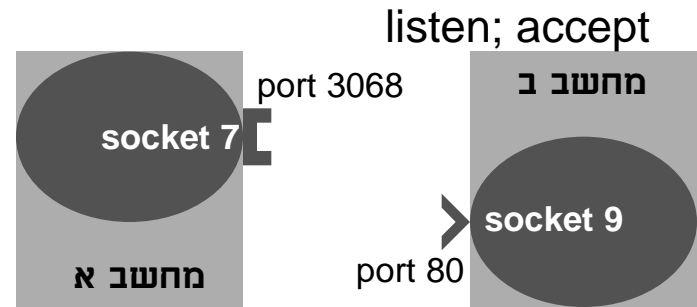
1



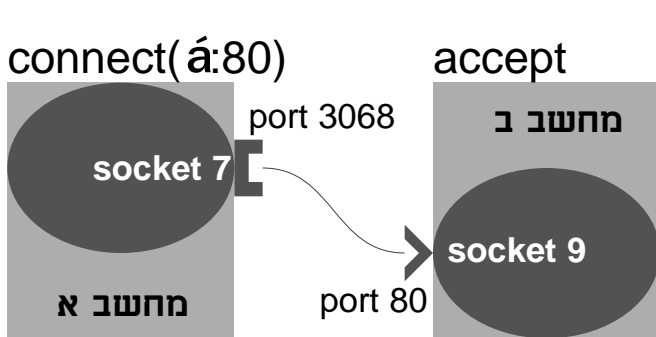
2



3

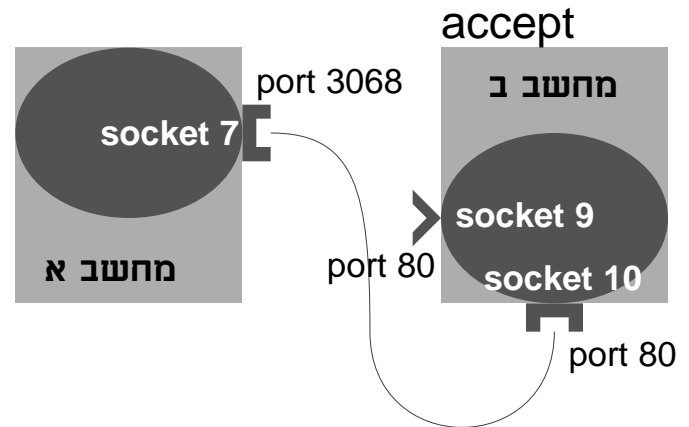


4



5

6



# תרגום כתובות IP (NAT)

- ❖ להרבה רשתות אין מספיק כתובות IP ציבוריות (כלומר שאפשר לנתב אליהן מנה מכל מחשב ברשת)
- ❖ למשל לבתים פרטיים מוקצה כתובת אחת לכל היותר
- ❖ הרשת משתמשת בטווח של כתובות שמותר להשתמש בהן שימוש חוזר בתוך ארגונים אבל אסור להן להופיע באינטרנט הציבורי (private addresses)
- ❖ הנתב שמחבר את הרשת הפרטית לאינטרנט מתרגם את הכתובות הפרטיות בכל מנה יוצאת לכתובת שלו וממציא מספר שער
- ❖ הוא זוכר את המיפוי ומתרגם בחזרה מנות נכנסות
- ❖ בעייתי ודורש קינפוג כאשר המנה הראשונה בקשר מגיעה מבחוץ

# זרימת מידע במחסנית הפרוטוקולים

- ❖ קריאת מערכת מעבירה מידע מתהליך (שכבת היישום) לשכבת ההעברה דרך שקע
- ❖ קריאת המערכת מזהה את פרוטוקול ההעברה על פי סוג השקע וקוראת לשגרה מתאימה של אותו פרוטוקול
- ❖ השגרה הזו מפרקת את המידע למנות או מצרפת אותו למידע קודם או מתייחסת אליו כמנה, מוסיפה למנות תחילית עם שדות של הפרוטוקול, ומעבירה אותן לשגרה של שכבת הרשת
- ❖ השגרה של שכבת הרשת מחליטה איך לנתב את המנה, מוסיפה לה תחילית IP, ומעבירה אותה לשגרה של פרוטוקול המיקשר המתאים
- ❖ השגרה של שכבת המיקשר מתרגמת את כתובת ה-IP לכתובת פיזית של מחשב שמחובר לקשת היוצאת, מוסיפה תחילית, ומעבירה לבקר לשליחה

# פרוטוקול ARP

- ❖ פרוטוקול לתרגום כתובות IP לכתובות פיזיות של מחשבים במקטע אתרנט
- ❖ ממומש כחלק משכבת המיקשר, לתרגום כתובות רשת לכתובות מיקשר; פרוטוקולים דומים דרושים למימושי מיקשר אחרים
- ❖ מחשב שצריך לבצע תרגום כזה שולח הודעת שאילתה לכל המחשבים במקטע (הודעת broadcast)
- ❖ מחשב שמזהה את כתובת ה-IP שלו בשאילתה כזו עונה בהודעה עם כתובת האתרנט שלו (MAC address)
- ❖ שכבת המיקשר מטמינה את התרגומים הללו בטבלה לזמן מה על מנת לחסוך בהודעות ARP (שמעכבות משלוח מנות)