תרגיל תכנות: שכבה 2
(חלק ראשון)

0512.4491 מעבדה מתקדמת
בתקשורת מחשבים

מטרה. בתרגיל זה נמשיך את המימוש מהתרגיל הקודם ע"י מימוש שכבה 2. הפעם תשתמשו בכרטיס רשת וירטואלי NIC (המשמש כשכבה 1) שאנו נספק, שכבה 3 שכתבתם בתרגיל הקודם, ובמודול נוסף שאנו נספק, המהווה את שכבות 4+5.

מבנה. הסבר מפורט על מבנה התוכנית, המתודות העומדות לרשותכם והמשתנים הסטטיים העומדים לרשותכם מפורט באתר http://www.eng.tau.ac.il/~tom/. בסוף המסמך מצורף תיאור זרימת המידע בתוכנית ומבנה חבילות ETHERNET וחבילת ARP.

שכבה 2 תממש את הפונקציות sendToL2 ו recvFromL2. מבנה הקוד ישתנה למימוש מקבילי. ביתר פרוט, יהיו שני חוטים: חוט אחד יהיה אחראי על משלוח הודעות והשני על קבלת הודעות. החוט של המשלוח יהיה החוט הראשי (של מודול ה main). החוט השני ייווצר כאשר ה main קורא למתודה connect של כרטיס הרשת (הפעולה מדמה חיבור פיזי של כרטיס הרשת לרשת), והוא ימתין לחבילות (שמגיעות באופן אסינכרוני); כשמגיעה חבילה, החוט יטפל בה מרמה 1 ועד רמה 4.

התכנית הראשית. תקבל בשורת הפקודה כקלט ארגומנט יחיד, שיהיה כתובת הIP למשלוח הודעת הICMP דוגמא לשימוש הינה: Netlab02.exe 192.168.2.1.

לאחר האתחול (שכאמור יוצר חוט נוסף הממתין בשכבה 1) מאתחלת payload לבחירתה (מחרוזת תווים כפי שמוצג ב main כדוגמא) ושולחת אותו לשכבה 4, שבמקרה שלנו היא ICMP ע"י קריאה ל SendToL4. שכבה 4 תעטוף את הpayload הזה ב ICMP ECHO Header ותעביר אותו לשכבה 3 (שלכם מהתרגיל הקודם) ע"י קריאה ל SendToL3, שבתורה תיצור חבילת IP שתועבר לשכבה 2 (שעליכם לממש כעת) ע"י קריאה ל SendToL2. תפקידה של שכבה 2 בכיוון זה הוא ליצור מסגרת אתרנט (Ethernet frame) שלמה למעט 4 בתי ה CRC שבזנבה. כלומר, צריך ליצור את כותרת האתרנט (Ethernet header) ולשלוח את ההודעה ע"י קריאה לפונקציה lestart של כרטיס הרשת הווירטואלי (NIC) הרושמת את ההודעה על הקו. לשם כך, בין היתר, על שכבה L2 לקבוע את כתובת ה MAC של היעד. פעולה זו תבוצע ע"י קריאה לפונקציה arpresolve של שכבת העזר L2_ARP (הנתונה לשכבה L2 ואותחלה מראש על ידי התוכנית הראשית) אשר מממשת מודול ARP. את שכבה זו אנו נממש במעבדה הבאה.

שימו לב: בתרגיל הקודם קריאת RecvFromL4 נעשתה על ידי התוכנית הראשית בכיוון זרימה הפוך לזה שמוצג כאן. נדרש לבצע עדכון לפונקציה RecvFromL3 כך שתתמוך בכיוון הזרימה המתואר במסמך זה. משמעות הדבר הינה שקריאה לפונקציה RecvFromL4 תטופל כאילו הקריאה נעשתה על ידי שכבה L3, ולכן הקריאה הקודמת מתוך התוכנית הראשית תטופל כך, באופן שלא היינו מצפים לו. אי לכך, על מנת לבצע recv מתוך ה main יש לקרוא לפונקציה readFromL4. פונקציה זו הינה blocking וממתינה להגעת חבילת ICMP REPLY.

מציאת כתוב MAC

ב IP, מציאת כתובת שכבה 2 (כתובת ה MAC) עפ"י כתובת שכבה 3 (כתובת IP) מתבצעת ע"י מודול ARP. אתם תממשו מודול זה בתרגיל הבא. בתרגיל זה נממש גרסה בסיסית מאוד למודול זה, המשתמשת בטבלת תרגום סטטית.

כלומר, נממש פונקציה אשר תחזיר לנו טבלה בפורמט טקסטואלי אשר עליו אנו נבצע את השאילתות הרצויות. מבנה הטבלה יהיה בדיוק כמבנה הטבלה המוחזרת ע"י הקריאה לפקודה arp -a מ command prompt. (ראו נספח להנחיות לבניית טבלה).

שאלה חשובה היא איזה כתובת IP יש לשלוח לתרגום. בהינתן כתובת יעד, יש להבחין האם היא בתוך או מחוץ לרשת המקומית בה אנו נמצאים (מודול ה ARP אינו יודע לתרגם כתובות מחוץ לרשת המקומית). החלטה זאת אנו מחשבים ע"י ביצוע bitwise AND של כתובת ה IP עם ה- subnet mask שלה. אם היעד נמצא על הרשת המקומית שלנו, נשלח לתרגום את כתובת ה IP של היעד. [1] אבל אם היעד איננו מקומי, אז נשלח לתרגום את

כתובת ה IP של הנתב המקומי (את שמו אנו מוצאים ע"י פקודת myDefaultGateway). כך, ההודעה תישלח בשכבה 2 אל הנתב שיקדם אותה אל היעד שלה.

יש לשים לב לפונקציונליות ולנכונות הפונקציה המתוארים להלן:

- <u>אופן מימוש הפונקציה:</u> הבנאי של שכבה L4 מאתחל buffer מקומי ונועל מנעול בעת אתחול השכבה. ברגע שמגיעה חבילת ICMP REPLY, עוברת בהצלחה את שרשרת השכבות ומגיעה לפונקציה recvFromL4 הפונקציה מקלפת את ה ICMP Header ורושמת את תוכן ה payload לתוך ה buffer. לאחר מכן הפונקציה משחררת את המנעול וגורמת לכך שהפונקציה readFromL4 (אשר נמצאת במצב blocking) תצליח לנעול את המנעול ולקרוא את תוכן ה buffer. מיד לאחר שחרור המנעול, recvFromL4 נועלת אותו מחדש (וכך מתכוננת לחבילה הבאה) ונמצאת כעת בעצמה במצב blocking. בזמן הזה, הפונקציה readFromL4 מעתיקה את תוכן ה buffer למצביע שקיבלה (לו ה main הקצה מראש מקום. לאחר מכן, הפונקציה משחררת את המנעול כך שהפונקציה recvFromL4 יכולה להמשיך ריצתה.
- באם הגיע חבילת ICMP REPLY ונשמרה ב buffer אולם readFromL4 לא נקראה, ואז חבילה נוספת הגיעה, החבילה החדשה תדרוס את החבילה הישנה.
- באם readFromL4 כלל לא נקרא, אולם הגיע חבילת ICMP REPLY החבילה שהגיע תשמר.

<u>דיבוג התכנית.</u>

- כדי לעקוב אחרי ריצת התכנית, עליכם להכניס הדפסות של תוכן החבילות בצורה קריאה וברורה לבודק, בתוך תנאי if המקבל את המשתנה הגלובלי במחלקה, debug (אשר נקבע בעת אתחול המחלקה).
- ניתן לקבוע בתוכנית הראשית את רמת הדיבוג (debug level) ע"י הזנת true או false לבנאי השכבות בהם מעוניינים לראות הדפסות.
- בנוסף, לכרטיס הרשת ניתנה אפשרות להוספת filter בפורמט של Winpcap filtering expression syntax לשימושכם בעת דיבוג התוכנית.

<u>שימו לב:</u> הוספת ה filter הינה פיצ'ר שהוכנס במטרה להקל על הליך דיבוג התוכנית בלבד, כלומר, ההגשה הסופית תיבדק כאשר לא מוכנס כלל filter וכאשר ה NIC במצב promiscuous.

<u>שימו לב:</u> כיוון ששני החוטים עלולים לנסות להדפיס בו זמנית, יש להשיג את המנעול print_mutex שנוצר בכרטיס הרשת (שאנו מספקים) לפני הדפסה ולשחררו בסופה. דוגמא כיצד להשתמש בו מוצגת להלן:

```
pthread_mutex_lock(&NIC::print_mutex);
cout << "Hello World!" << endl;
pthread_mutex_unlock(&NIC::print_mutex);
```

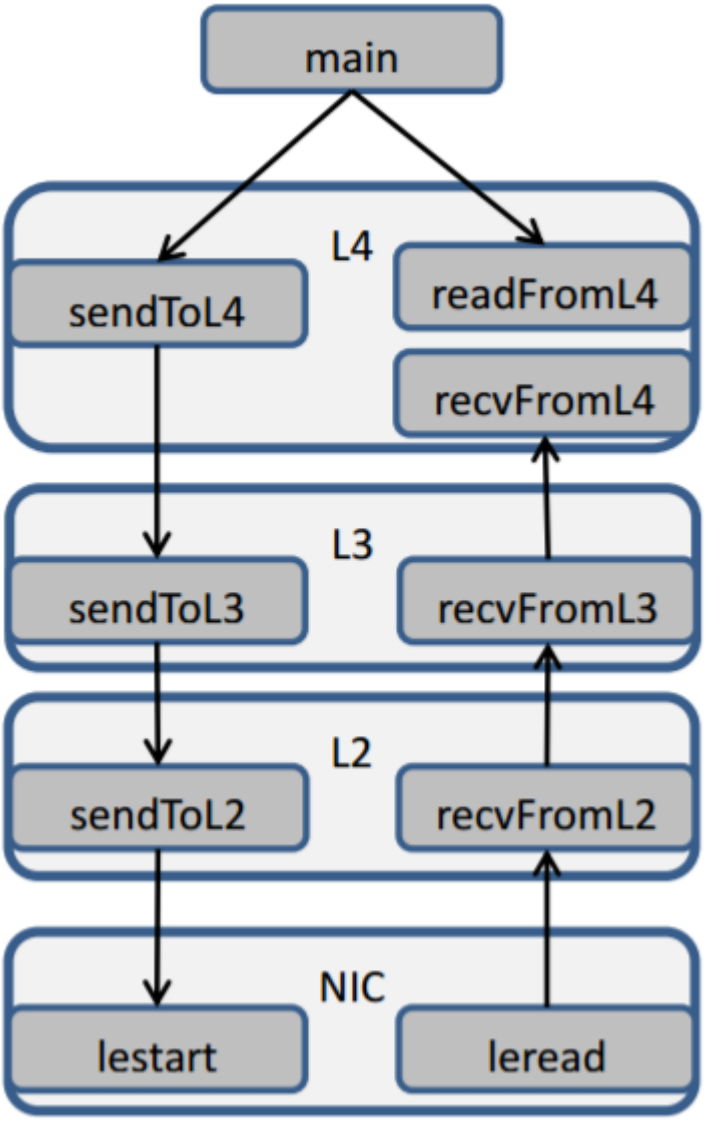
[1] בתרגיל זה אנו יודעים בוודאות כי הכתובת נמצאת בטבלה. בעולם האמתי (ובתרגילים הבאים) הנחה זאת אינה מתקיימת תמיד. במקרה זה נאלץ לשלוח הודעת ARP לכל התחנות ברשת המקומית על מנת לגלות מי מהם מחזיק בכתובות המבוקשת ולעדכן את הטבלה בהתאם.

מה אתם מקבלים. כאשר תורידו את החומר לתרגיל, תמצאו ספריה ובה:

- ספריות (ריקות) בשם Debug ו Release – כרגיל, לעבודה.
- WinPCap: מכילה מודולים בינאריים מועילים שלא תשתמשו בהם ישירות.
- Pthreads: מכילה מודולים בינאריים מועילים בהם תשתמשו לתמיכה במקביליות התוכנית.
- NetlabTAU: זהו האזור בו עלים להתמקד. הוא מכיל 3 תת-ספריות:
  - Debug, release: מכילים את הספרייה הבינארית איתה תעבדו בתמיכה בגרסת **VS2013**.
  - Debug10, release10: מכילים את הספרייה הבינארית איתה תעבדו בתמיכה בגרסת **VS2010**.
  - Include: מכילה את קבצי ההגדרות (*.h).
- קובץ המקור של הלקוח תחת השם main.cpp. הלקוח שולח חבילת ICMP מסוג ECHO ליעד המצוין כארגומנט בשורת הפקודה ומדפיס את תוכן התשובה.

הגשת התוכנית: אין צורך להגיש בנפרד חלק זה של המעבדה, יש להגיש דו"ח מסכם אחד.

תרשים זרימת המידע בתוכנית:



כיצד ליצור עותק מקומי של טבלת ה- **ARP** :

1. נפתח את ה-CMD.
2. נשלח פקודת ping לשני מחשבים. האחד ברשת הפנימית שלנו, והאחר מחשב מרוחק (במקרה שלנו שרת ה-DNS של גוגל).
3. נריץ את הפקודה הבאה: arp -a > arpTable.txt. בקובץ שייווצר תישמר תכולת טבלת ה- arp. חשוב לא לשנות את שם הקובץ. בנוסף, יש לדאוג כי הקובץ נשמר בתיקייה של התוכנית (אין צורך להגיש את הקובץ הזה).
4. מהקובץ שיצרתם יש להפיק שני MAC addresses. הראשון של מחשב ברשת הפנימית שלכם, והשני של הנתב.
5. את כתובת ה- IP של הנתב שלכם תוכלו לדעת מתוך פקודת myDefaultGateway.
6. כתובת ה- IP של המחשב ברשת הפנימית שלכם מוזנת ב- Main.cpp.
7. כעת, משנתונה לכם טבלת ה- arp וכתובת ה- gateway של הנתב וכן כתובת ה- IP של המחשבת ברשת הפנימית שלכם, תוכלו לקבל את ה- Address Mac הרצוי עבור כל אחד מן המקרים.

<u>חשוב:</u> את שלבים 4-6 אין להפיק באופן ידני, כלומר הכתובות הללו אינן מובנות בתוכנית. זכרו כי התוכנית תיבדק על מחשב נפרד ולכן הזנה ידנית של כתובות דינה להיכשל בעת הבדיקה.