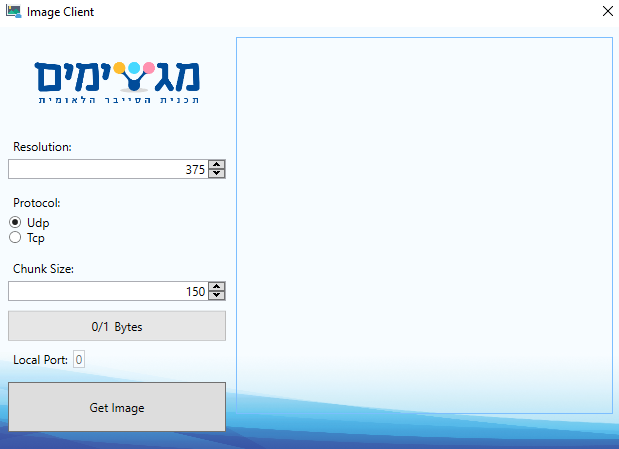
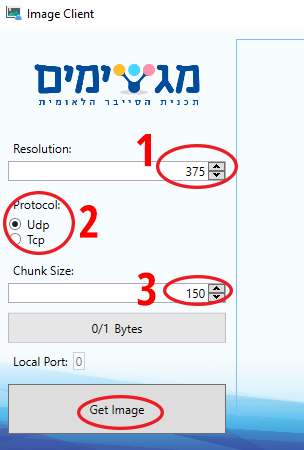
שיעור 8 – המשך TCP

**תרגיל: מסביב לעולם ב-2 פרוטוקולים**

בתרגיל זה נשתמש בתוכנה אשר עושה דבר אחד בלבד: מורידה תמונה ומציגה אותה על המסך. אבל איך היא עושה זאת? בזה אתם תשלטו. תוכלו להחליט באיזה פרוטוקול שכבת התעבורה תשתמש התוכנה – UDP או TCP ותוכלו לשלוט בפרמטרים נוספים הקשורים לחיבור. בעזרת התוצאות וניתוח ההסנפות נבין קצת יותר טוב מה ההבדל בין שני הפרוטוקולים.

**1. הרצה ראשונה של התוכנה**

1. הורידו את את הקובץ של תוכנת הורדת התמונה (ImageClient.zip) מתיקיית תרגיל הבית. פתחו את קובץ ה-zip לתוך תיקיה והיכנסו לתוך התיקיה.
2. הריצו את התוכנה ע"י לחיצה כפולה. במידה ויש לכם אנטי וירוס על המחשב, הבדיקה עשוי לקחת כחצי דקה.
3. התוכנה אמורה להיראות כך:
4. לחצו על Get Image כדי לקבל תמונה. אם לא הצלחתם לקבל תמונה, יתכן כי הפורט חסום ברשת שלכם. נסו לעבור לרשת ביתית או להוטספוט.

**הוראות שימוש בתוכנה:**יש לקבוע את כל הפרמטרים:

**1. קביעת רזולוציית התמונה:** רזולוציית התמונה קובעת את גודל התמונה, בפיקסלים רוחב/גובה. לדוגמא, אם נבחר 100 נקבל תמונה בגודל 100X100. ככל שהמספר יותר גבוה, התמונה שיורדת תהיה גדולה יותר, ולפיכך גם איכותית יותר ושוקלת יותר. הערך המקסימלי: 2048.

**2. קביעת הפרוטוקול:** קביעת פרוטוקול בשכבת התעבורה – UDP או TCP.

**3. קביעת גודל חתיכה:** רלוונטי רק ל-UDP. מכיוון ש-UDP לא מבצע סגמנטציה בעצמו, אנו קובעים את גודל החתיכות (בבתים) אליהן  
תחולק התמונה בהעברה. זהו לא מספר החתיכות אלא גודל כל חתיכה.

כשאנחנו מוכנים נלחץ על **Get Image** והתמונה תרד מהשרת.  
מעל התמונה מוצג הפורט המקומי שנפתח לצורך התקשורת האחרונה.

**2. משחקים עם התוכנה!**

1. קחו מספר דקות כדי לשחק עם התוכנה. נסו לשנות את הפרמטרים ולראות מה קורה בכל מצב.

**3. הורדת תמונה ב-TCP**

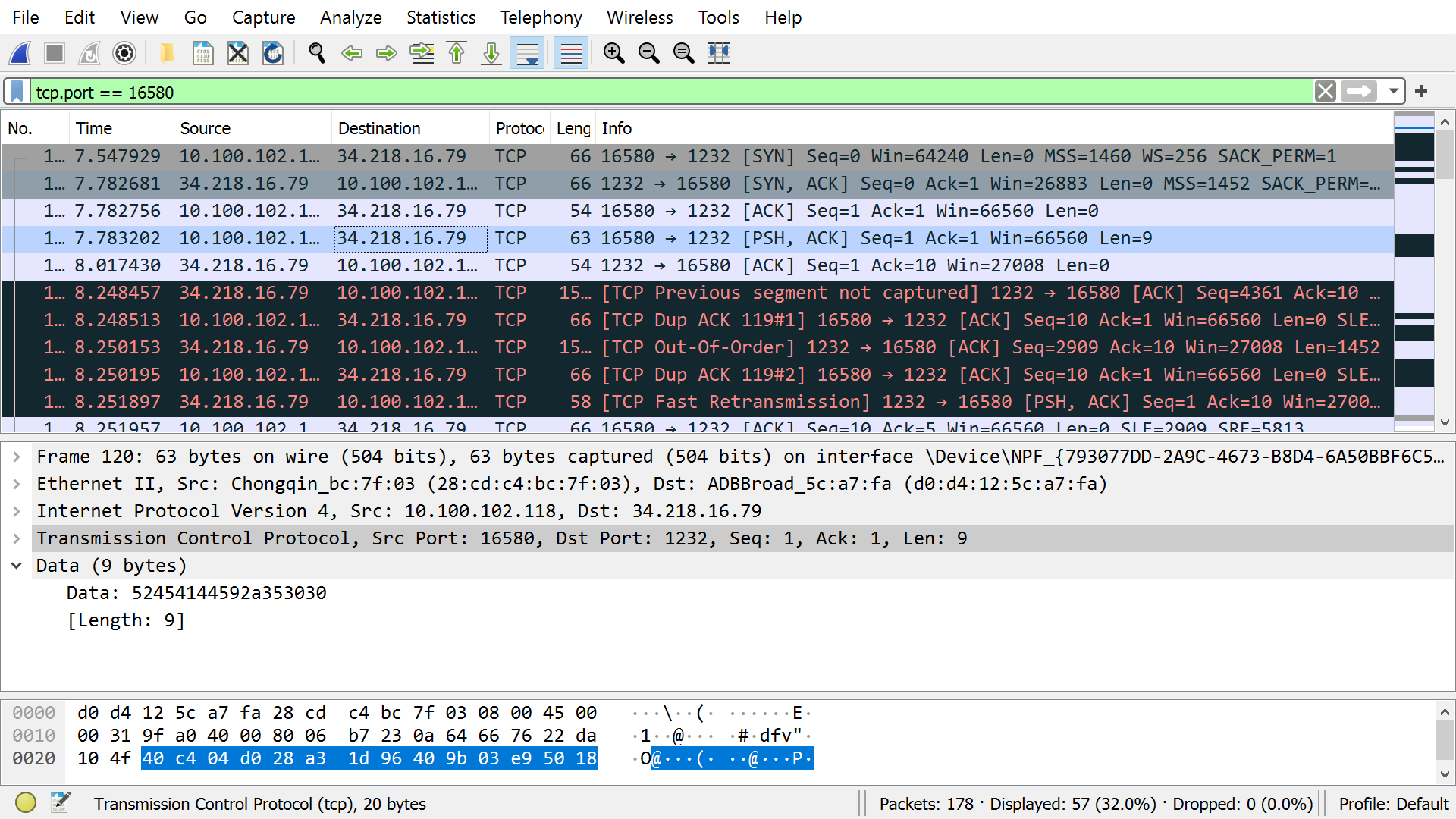
1. כששדה TCP מסומן, נסו להוריד את התמונה בגדלים שונים. האם התמונה תמיד יורדת במלואה וללא טעויות? האם התוצאה הזאת התאימה לציפיות שלכם? הסבירו למה.

|  |
| --- |
| כן, התמונה תמיד מגיעה טוב ובלי בעיות כמו צבעים לא נכונים, סדר לא נכון של חלקים, ובלי חסרים. זה מה שציפיתי כי לTCP יש מערכת למניעת בעיות כאלה- בדיקת צ'קסאם שמובילה לאנשהו, SEQ שאחראי על הסדר ובכללי הרבה retransmission |
| *תשובה* |

1. התחילו הסנפה ב-Wireshark, בצעו הורדה חדשה של התמונה (גודל 500), ועצרו את ההסנפה.
2. מכיוון ש-Wireshark לא מזהה את הפרוטוקול שלנו, הדרך היחידה שלנו לסנן את ההסנפה היא באמצעות הפורט. הסתכלו בפורט שמופיע בתוכנה ליד Local Port וכתבו פילטר מתאים. שימו לב שאתם מסננים גם חבילות נכנסות וגם יוצאות הכוללות את הפורט.

\* שימו לב שבכל הורדה של התמונה מתחלף הפורט (כפי שלמדנו, הפורט המקומי הוא אקראי).

1. זהו את לחיצת היד המשולשת בשיחה, וצרפו צילום מסך שלה.



1. מלאו את הטבלה הבאה לגבי מאפייני ה-Socket מצד הלקוח:

|  |  |
| --- | --- |
| **Dst IP** | **Src IP** |
| 34.218.16.79 | 10.100.102.117 |
| **Dst Port** | **Src Port** |
| 1232 | 16580 |

1. מהו ה-ISN אשר בחר הלקוח?

|  |
| --- |
| 681778581 |
| *תשובה* |

1. במסך העליון של Wireshark, בחרו את ההודעה הראשונה שנשלחה מהלקוח (תחילת לחיצת היד). במסך האמצעי של Wireshark, פתחו את הפירוט לגבי פרוטוקול TCP. כאשר אתם מסמנים את שדה ה-Sequence Number, בדקו מהו ערך ההקסא שמסומן בחלון התחתון של Wireshark. כתבו אותו פה בהקסא ולאחר מכן המירו אותו לדצימלי.

|  |
| --- |
| HEX = 28 a3 1d 95 DEC = 681778581 |
| *תשובה* |

שימו לב! אם המספר שכתבתם בסעיף ו' הוא 0, הרי שזה לא ה-ISN האמיתי!   
ה-ISN האמיתי הוא זה שמצאתם בסעיף ז'. Wireshark מאפס את כל ה-ISN כדי להקל על המשתמש. מכיוון שאנחנו רוצים לראות את ה-Seq האמיתיים, **נוריד את סימון ה-V ב:  
Edit->Preferences->Protocols->TCP->Relative sequence numbers**

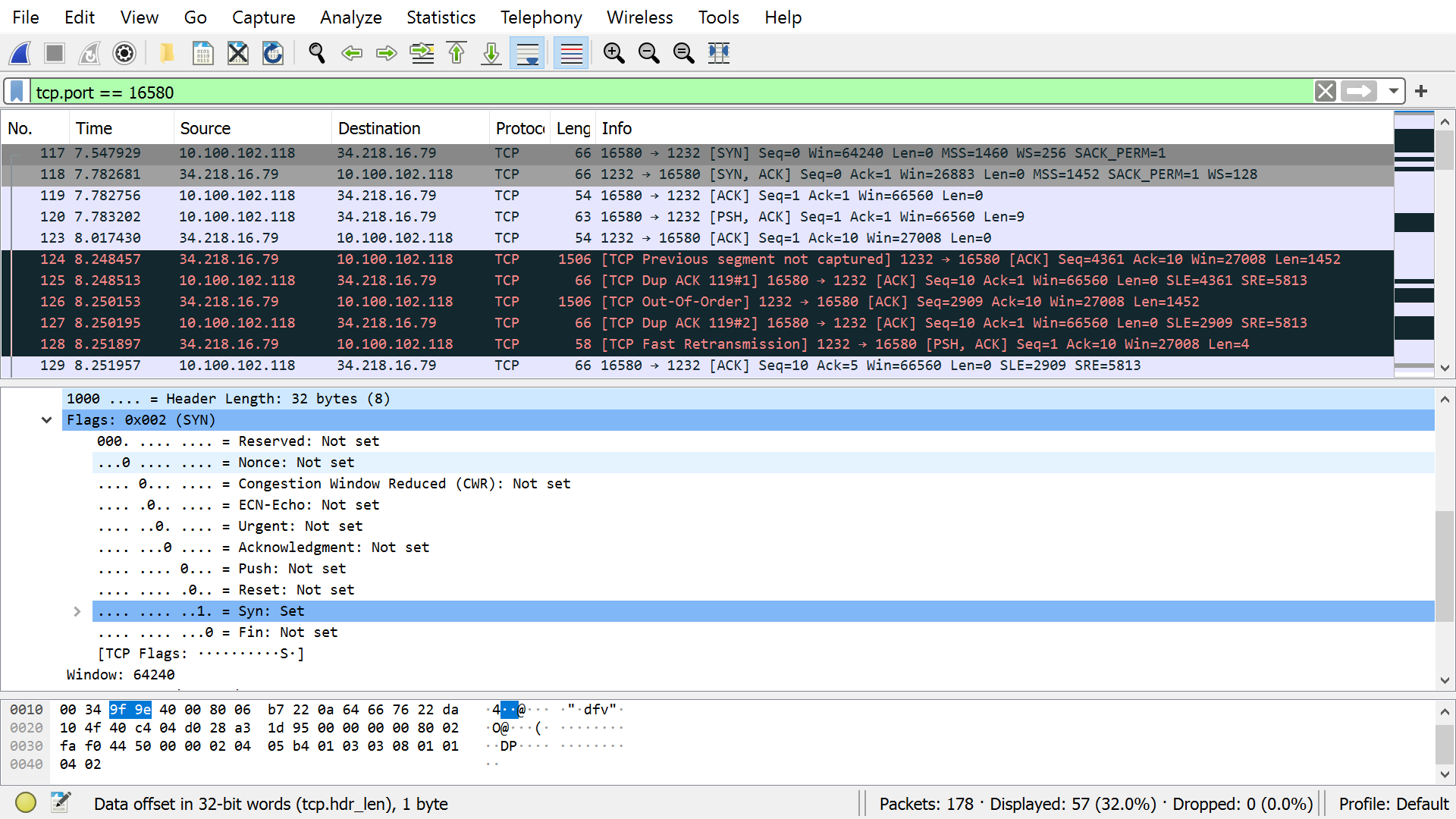
1. מהו ה-ISN אשר בחר השרת?

|  |
| --- |
| 1083900904 |
| *תשובה* |

1. מהו גודל ה-Header של TCP (בבתים)?

|  |
| --- |
| 32 |
| *תשובה* |

1. בתוך חבילת ה-SYN הראשונה בתעבורה, פתחו את פירוט ה-Flags שמציג Wireshark. צרפו צילום אשר מראה כי הדגל של SYN דלוק.



1. שימו לב ל-Seq Number בחבילות אשר **יוצאות מהמחשב שלנו** אל השרת. מה קורה למספר ה-Seq? האם הוא משתנה? כיצד תוכלו להסביר זאת?

|  |
| --- |
| בהתחלה כן, הוא עבר מ-0 ל-10. מאחרי זה הוא לא השתנה בכלל (חוץ מבFIN שהוא עב ר ל11).  זה כנראה בגלל שאנחנו שלחנו רק קצת חבילות בהתחלה – מה אנחנו רוצים לקבל ומאז רק אישרנו שקיבלנו את החבילות שלו כי אנחנו לא מנסים להעלות לשרת משהו, במילים אחרות- נגמר לנו המידע להעביר לשרת אבל לשרת לא נגמר המידע להעביר לנו אז אנחנו רק מאשרים |
| *תשובה* |

1. מצאו את ההודעה (שכבת האפליקציה) ששלח הלקוח לשרת וכתבו אותה:

|  |
| --- |
| 52454144592a353030 |
| *תשובה* |

1. בעזרת מחשבון, בצעו חיסור בין מספר ה-Seq **האחרון** **ששלח השרת** לבין מספר ה-Seq **הראשון ששלח השרת.** כתבו את התרגיל ואת התוצאה.מה לדעתכם מייצגת התוצאה?

|  |
| --- |
| 1083938231 – 1083900904 = 37,327 (יצא הפוך כי עברית...), התוצאה מייצגת את מספר הבייטים שנשלחו אלינו מהשרת, שזה גודל המידע פלוס נספחים כמו SYN, FIN וכו' |
| *תשובה* |

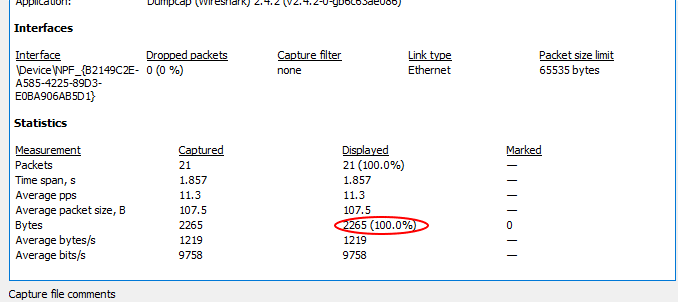
**3. חישוב תקורה (Overhead)**

תקורה, או Overhead, זה "המס" שאני צריך לשלם כדי לקיים דבר מה. למשל, אם נסיעה לשיעור במגשימים לוקחת לי שעה לכל כיוון, הרי שיש לי שעתיים של תקורה לעומת שלוש שעות של שיעור. ברשתות, אנו קוראים תקורה לכל המידע שאני צריך להעביר שאינו ההודעה עצמה שרציתי להעביר. במקרה שלנו, מדובר **בכל השכבות** שהן לא שכבת האפליקציה. בתרגיל זה נחשב את התקורה ב-TCP לעומת UDP.

1. בצעו סינון אשר יציג רק את החבילות שנשלחו מהשרת ללקוח. כתבו את הפילטר שכתבתם:

|  |
| --- |
| ip.src == 34.218.16.79 |
| *תשובה* |

1. כעת נרצה למדוד כמה בתים מופיעים בפילטר הנוכחי, כלומר כמה בתים הועברו מהשרת אלי כדי שאקבל את התמונה. לחצו בתפריט התוכנה על Statistics → Capture File Properties   
   (אם אין לכם אופציה כזאת, חפשו את Summary Statistics →).



1. תחת הכותרת **Displayed** חפשו את השורה Bytes והסתכלו על הערך תחת Displayed.

|  |
| --- |
| 79642 (65.3%) |
| *תשובה* |

1. כעת נבצע חיסור בין מספר זה לבין גודל התמונה הנקי (מוצג בתוכנה). **זוהי התקורה** להעברת תמונה בגודל 500X500 תחת TCP. כתבו את התוצאה.

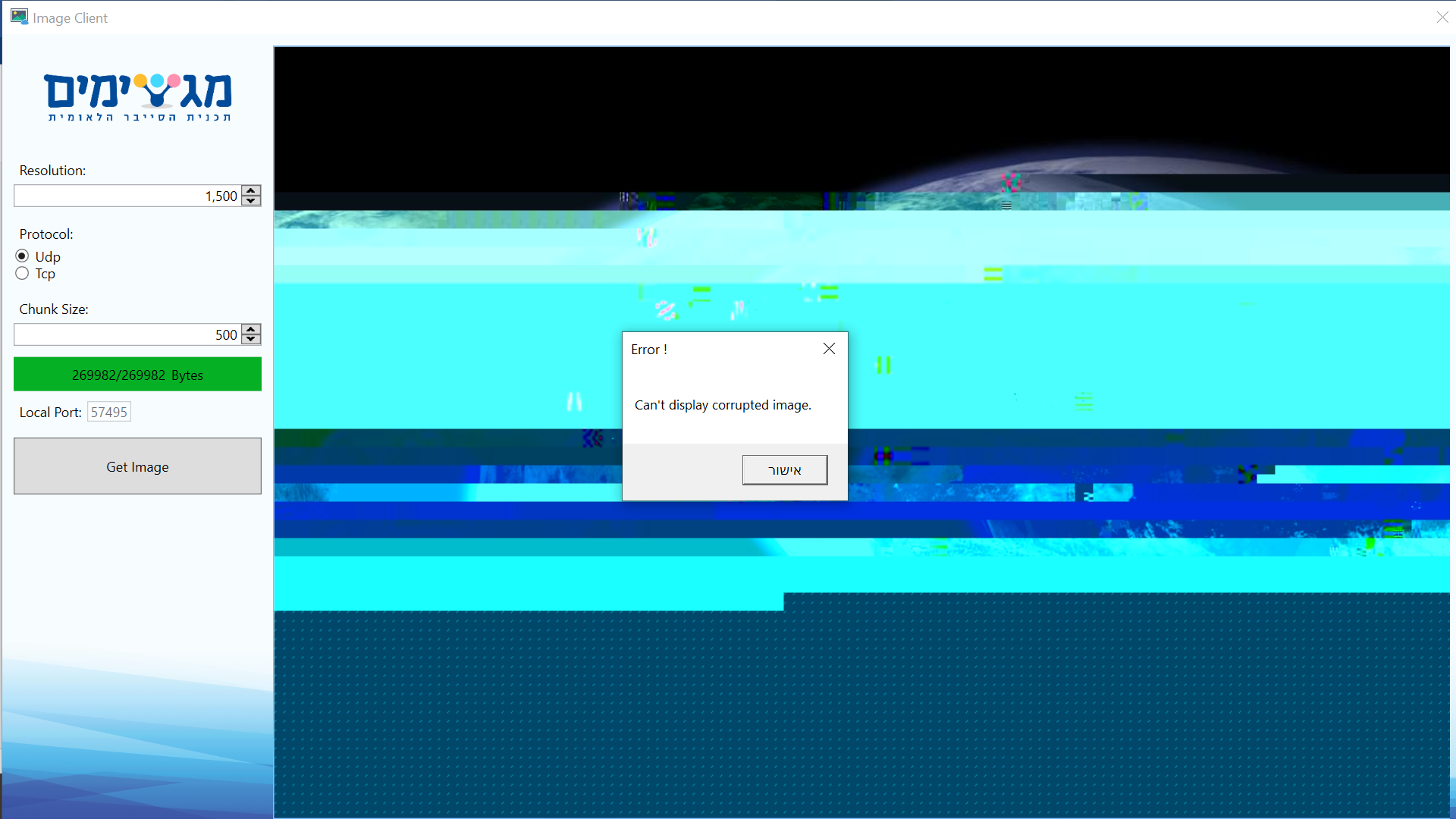
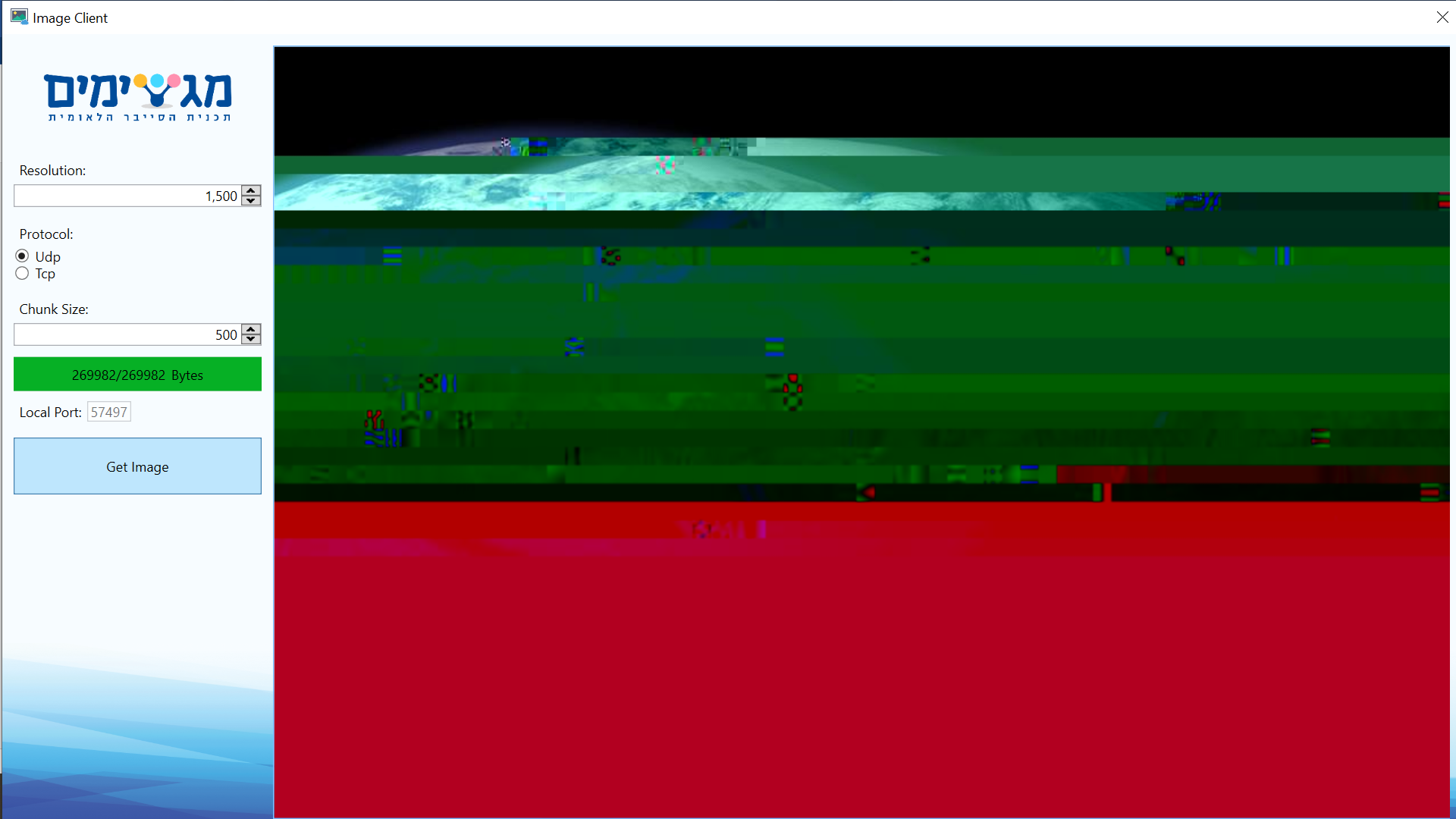
|  |
| --- |
| 42,320 |
| *תשובה* |

**4. הורדת תמונה תחת UDP**

1. תחת UDP, הורידו תמונה בגודל 200. שחקו עם שדה ה-Chuck size החל מ-1000 וכלפי מטה. האם תמיד קיבלתם תמונה תקינה? כיצד תסבירו זאת?

|  |
| --- |
| לא, בגלל שUDP לא מתקן טעויות- לפעמים הייתה בעיה והתמונה הייתה מקולקלת ולא ניתנת להצגה (בעייה בצ'קסאם?), לפעמים התמונה הגיעה לא טוב- לא בסדר הנכון ושנראה כאילו יש עליה פילטר צבע... ועוד כל מני בעיות, ככל שהצ'אנק סייז יותר גדול יש פחות בעיות שאי אפשר להציג את התמונה ובכללי פחות בעיות, וזה הגיוני כי יש פחות חבילות ופחות סיכוי לטעויות |
| *תשובה* |

1. כעת בחרו גודל תמונה 1500. נסו מספר פעמים עם Chunk size=500, ומספר פעמים עם Chunk size=20. מה ההבדלים בביצועים? כיצד תסבירו זאת? השתמשו בצילומי מסך כדי להסביר את ההבדלים בין שני המקרים.



|  |
| --- |
| אז... עם 500 קיבלתי לפעמים בעיות של תמונה מקולקלת ולפעמים תמונות ממש גרועות כמו שאפשר לראות בצילומי מסך. ב20 קיבלתי כל הפעמים שניסיתי שאי אפשר להציג את התמונה חוץ מפעם אחת שלא קיבלתי כלום – תמונה לגמרי שקופה. |
| *תשובה* |

1. נבצע חישוב תקורה ב-UDP. נבחר בגודל תמונה 500 וב-Chunk size=1400 (דומה ככל האפשר ל-TCP). נבצע הסנפה ונוריד את התמונה. נבצע פילטר להודעות ששלח השרת ב-UDP בלבד. באותה דרך שבסעיף 3-ב', נמדוד את כמות הבתים המוצגים. הציגו את החישוב של התקורה (הגודל המוצג פחות גודל התמונה).

|  |
| --- |
| 1,180 |
| *תשובה* |

1. באיזה מהפרוטוקולים התקורה להעברת התמונה גדולה יותר? האם תוצאה זו הגיונית לפי מה שלמדנו? הסבר מדוע.

|  |
| --- |
| TCP, כן כי לTCP יש הרבה הרבה יותר מידע בהדר, ולכן הוא יותר אמין, איטי וכבד |
| *תשובה* |