Reliable file transfer over ICMP protocol (ICMP Tunneling)

פירוט הפרויקט

תיאור כללי:

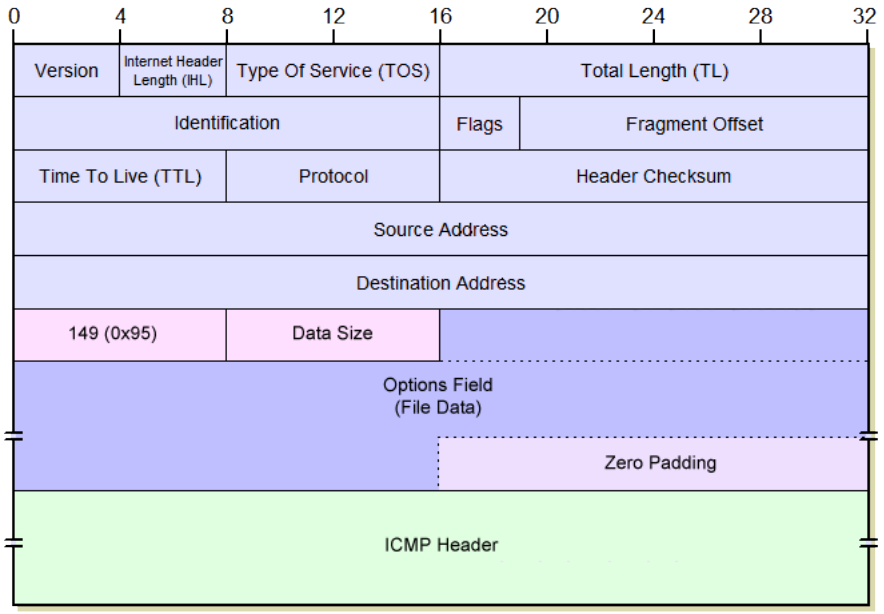
מטרת הפרויקט היא יישום העברת קבצים אמינה מעל פרוטוקול ICMP ותחת ערבול payload.
הפרויקט נכתב בשפת התסריט פייתון.

מבנה החבילות:

התקשורת בין השרת ללקוח מתבצעת באמצעות ping־ים (echo request & reply). הקובץ עצמו מועבר בשדה ה־options של תקורת (header) ה־IP, במסווה של Selective Directed Broadcast (הצעה לפרוטוקול שעלתה ב־RFC–1770). הבית הראשון בשדה הוא מספר האופציה 149 (0x95), הבית השני הוא אורך המידע, כולל זוג בתים אלה, והשאר הוא ריפוד אפסים לכפולה של 4.

אורך תקורת ה־IP מוגבלת ל־$2^4-1 = 15$ מלים בנות 32 ביט (שכן גודל שדה ה־IP Header Length הוא 4 ביטים), כלומר 60 בתים סה״כ. גודל השדות הבסיסיים הוא 20 בתים, מה שמשאיר עד 40 בתים לשדה האופציות. כאמור לעיל, זוג הבתים הראשונים בשדה משמשים לציון מספר האופציה וגודל המידע, ומכאן שכל חבילה שנשלח תעביר 38 בתי־מידע מהקובץ, מלבד החבילה האחרונה. ריפוד האפסים יתבצע למעשה רק בחבילה זו (אם גודל הקובץ מתחלק בדיוק ב־38, החבילה האחרונה תהיה ריקה).

תרשים: מבנה חבילה



בתקורת ה־ICMP נקבעו זוג הבתים הראשונים ל־8 ול־0 בהתאמה, לציין הודעות "בקשת־הד". שדה ה־ID נקבע לערך ה"שרירותי" 24102 (=0x5E26, שהוא בבדיחות ערך ה־checksum של המחרוזת "Baruch HaShakel"), וזאת על־מנת לברור את הודעות בקשות־ההד של התכנית מבין שאר בקשות ההד שעלולות להגיע למערכת. שדה ה־sequence number מסמל את מספרה הסידורי של החבילה. חבילות בקרה, של הקמת קשר וסגירתו, מקבלות את המספר הסידורי 0. גודל ה־payload בכל חבילה הוא 0 (כלומר שאין מידע בתוך ה־ping־ים, ומכיוון שכך, לנתב אין מה לערבל, ועל־כן אנו יכולים להסתמך על ה־cheksum של תקורת ה־ICMP בכדי לוודא שלא חלו בה שיבושים.

אופן הפעולה:

הקמת הקשר

הקמת הקשר מתבצעת באמצעות לחיצת יד משולשת (3-Way Handshake), השרת מאזין לבקשת קובץ מהלקוח, ברגע שהוא מקבל בקשה כזו (הודעת FILE_REQ_MESSAGE), הוא שולח חזרה הודעת אישור בקשה (REQ_ACK_MESSAGE) ומצפה ל־"תשובת־הד" ממערכת הלקוח. הלקוח מתחיל להמתין לחבילות־המידע לאחר אישור הבקשה, והשרת מתחיל לשלוח אותן לאחר קבלת תשובת ההד.

השרת (שליחת הקובץ)

לאחר שהקשר הוקם, השרת טוען את הקובץ לזיכרון (לחיסכון בזמן) ושומר אותו בחוצץ file_buffer שהוא למעשה רשימה, כאשר הקובץ מחולק בה למחרוזות בנות 38 בתים שמוכנות לשליחה. השרת משתמש בחלון שליחה בעל גודל דינאמי, שגדל בהתאם למצב בקרת העומס. כאשר גודל החלון קטן מה־Threshold שנקבע, הוא גדל באופן מעריכי (מצב Slow-start), ב־1 עבור כל אישור חבילה (ומכאן מוכפל בכל RTT), וכאשר גודלו של החלון מעל ה־Threshold, הוא גדל באופן לינארי (Congestion-avoidance), ב־1 עבור כל RTT. השרת שולח חבילות כל עוד גודל החלון לא נוצל במלואו, או שכבר הגיע לשליחת החבילה האחרונה.

השרת שומר שני משתנים שנוגעים למספרים הסידוריים של החבילות: curr_segment מסמל את מספרה הסידורי של החבילה שעומדת להישלח – מתקדם ב־1 עבור כל שליחה, ו־waiting_for מסמל את החבילה האחרונה שידוע בוודאות שהשרת קיבל את כל אלו שקודמות לה (ולכן היא הבאה שלה הוא מצפה), מתעדכן רק בהגיע אישורים שנוצרו בידי תכנית הלקוח (ACK_MESSAGE), ולא בתשובות ההד אוטומטיות של מערכתו. בכל שליחת חבילה מונה הזמן (Timer) מאותחל אם לא נעשה זאת לפני־כן, כך שהמונה הוא עבור החבילה הישנה ביותר שלא אושרה.

במקרה של Timeout, כלומר כאשר לא מתקבלת שום הודעה חדשה במשך זמן מסוים, השרת שולח שוב את החבילה לה הלקוח מצפה (waiting_for), ומאתחל את מידע בקרת העומס: ה־Threshold נקבע לחצי מהחלון העכשווי, והחלון נקבע ל־1. אינדקס החבילה הבאה להישלח (curr_segment) חוזר לאינדקס waiting_for.

בהתקבל הודעה חדשה, אם מספרה הסידורי קטן מהאינדקס waiting_for, השרת מתעלם ממנה, אחרת הוא בודק האם זו הודעת אישור של תכנית הלקוח, או תשובת־הד אוטומטית. האינדקס waiting_for מתעדכן, כאמור לעיל, רק באישורי תכנית הלקוח. אם מתקבלות שלוש הודעות כאלה בהן המספר הסידורי לא התעדכן (ע״פ המונה dup_acks), השרת מבצע Fast Retransmit בו הוא שולח שוב את החבילה המדוברת, וכמו־כן מאתחל את מידע בקרת העומס: ה־Threshold נקבע

לחצי מהחלון העכשווי, והחלון נקבע לאותו ערך ועוד 3 (Fast-recovery, במקום ל־1 כמו במקרה של Timeout). בכל מקרה של אישור (מתכנית הלקוח או מן המערכת), חלון השליחה גדל לפי מצב בקרת העומס, כנאמר לעיל, והטיימר מאופס.

אם מצב בקרת העומס אותחל (מצוין במשתנה הדגלון restarted), בין אם בידי Timeout ובין אם בידי Fast Retransmit, השרת מחכה להודעת אישור של תכנית הלקוח כדי להמשיך, ומתעלם מתשובות־הד של המערכת.

אם השרת מקבל הודעת סיום (FIN_MESSAGE), הוא מבין שהלקוח קיבל את כל הקובץ ועובר לשלב סגירת הקשר.

הלקוח (קבלת הקובץ)

הלקוח פועל בשיטת אישורים מושהים (Delayed ACK), עם זמן השהייה של 200ms, הלקוח מחכה לקבלת חבילה ולמעבר זמן ההשהיה, ורק לאחר־מכן שולח הודעת אישור (ACK_MESSAGE) עם מספרה הסידורי של החבילה הבאה שהוא ממתין לה (waiting_for).

בהגיע חבילה חדשה ללקוח, הוא תחילה בודק האם היא הודעת בקשת־הד, ושאינה הודעת בקרה (מספר סידורי 0). את כל מקטעי הקובץ (segments) שנתקבלו הלקוח שומר במילון (dictionary), שכן הם לא בהכרח מגיעים בסדר הנכון. כאשר הלקוח מקבל חבילה שמקטע המידע שבה קטן מ־38 בתים, הוא מזהה אותה כחבילת המידע האחרונה, ושומר את מספרה הסידורי במשתנה last_segment. הלקוח מקדם את המשתנה waiting_for, שמסמל לאיזו חבילה הוא מצפה, לחבילה הראשונה שאינה נמצאת במפתח מילון החבילות שנתקבלו. אם החבילה שהוא מצפה לה היא מעבר לחבילה האחרונה (last_segment), הלקוח מסיק שהוא קיבל את הקובץ במלואו, ולכן כותב אותו לדיסק, ועובר לשלב סגירת הקשר.

סגירת הקשר

סגירת הקשר מתבצעת באופן דומה להקמתו, באמצעות לחיצת יד משולשת (3-Way Handshake), הלקוח שולח בקשת סיום קשר (FIN_MESSAGE), ומצפה לאישור סיום מהשרת (FIN_ACK_MESSAGE). השרת, לעומת זאת, ברגע שמקבל את הבקשה לסיום הקשר, שולח חזרה את הודעת אישור הסיום, ומצפה לתשובת־הד ממערכת הלקוח (עם מספר סידורי 0). הלקוח לא צריך להמתין עוד זמן לאחר קבלת אישור הסגירה, שכן המערכת היא זו שעונה את תשובות־ההד להן ממתין השרת.

EOF