

תקשורת מחשבים - פרויקט גמר

Paper Soccer

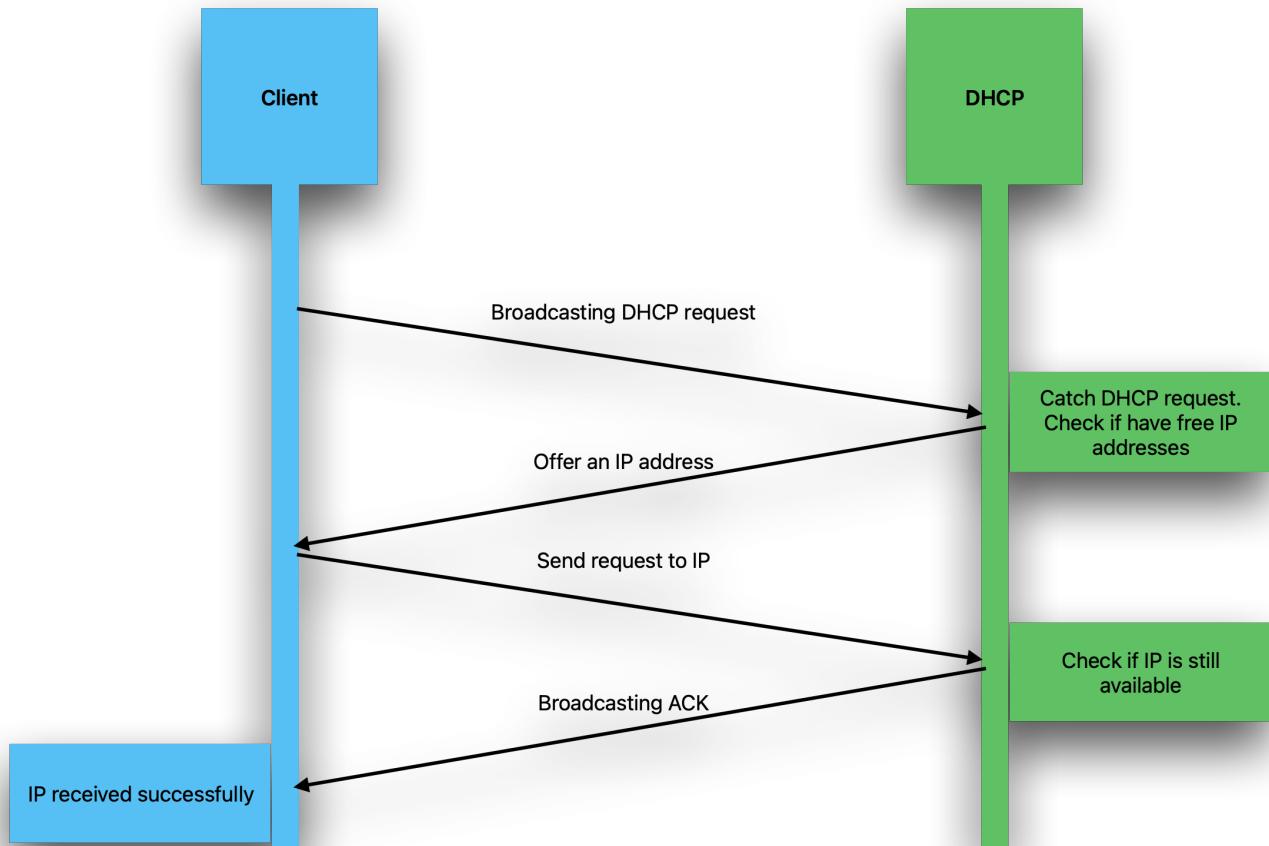
תיאור כללי

מטרת הפרויקט היא לבנות אפליקציה אשר מימושה תקשורתית בין קלינינט למספר שרתים שהם DNS, DHCP, Application Server. במקרה שלנו כתבנו משחקPaperSoccer שמתנהל על ידי השרת ושני שחקנים יכולים להתחבר אליו ולשחק יחד. התוכנה והשירותים נכתבו בשפת פיטון.

ניתן לראות סרטיון הדגמה [כאן](#).

DHCP

DHCP - זה שירות שמנהל את כתובת הIP ברשות המקומיית. מטרה שלו היא לתת כתובת ייחודית לכל לקוח חדש שמחבר לרשת. התקשורת מתנהלת ב프וטוקול UDP.



במקרה שלנו אין צורך ב DHCP אמתי מכיוון שנמצאנו כבר במצבם ברשות וקיים DHCP שמנהל אותה, לכן כתבנו שירות שמדמה את העבודה של DHCP. בצדקה בסיסית.

על מנת לכתוב את השירות השתמשנו בספרייה Scapy.

DHCP_IP, DHCP_PORT 127.0.0.2:67. ניתן לשנות בשדות כתובות השרת

בהרצתה של התוכנה היא תבקש להזין את שם המ捨יר עליו היא תאזור. עשינו זאת כיוון שאנו לא יכולים לדעת על איזה מ捨יר ייריצו אותה.

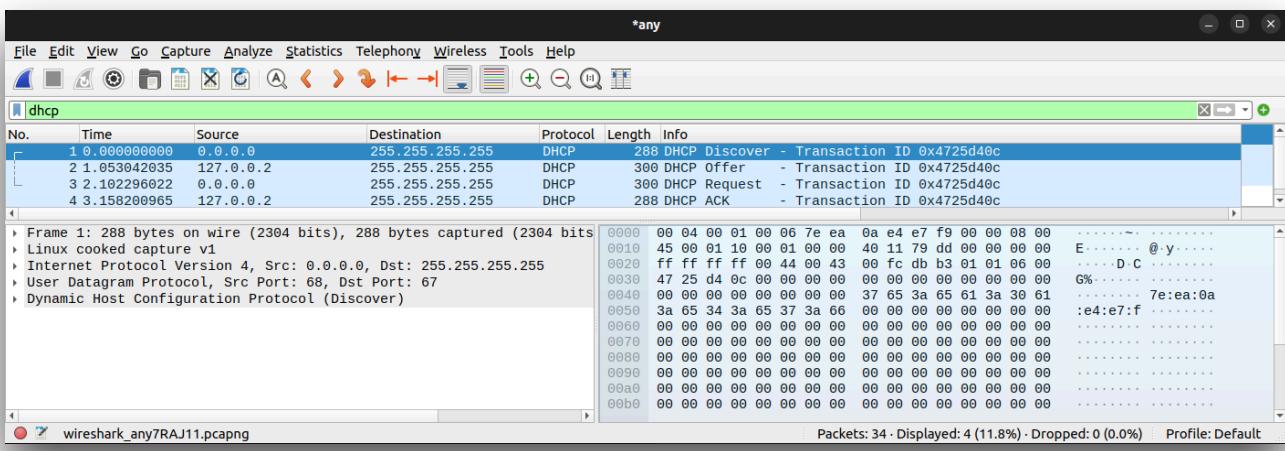
כאשר מריםים את התוכנה אז רצה בה לולה אינסופית אשר מבהה לקבל בקשות. כיוון שהתוכנה מסניפה לכל הפקחות ברשות עם פורט נתון ומסוג UDP, אז התוכנה מודדת שהפקטה מהויה בבקשת התחרבות. אחריו תפיסת הבקשה התוכנה תבודוק את המאגר שלה (במקרה שלנו רק עם שתי כתובות) ותשלח הצעה עם הכתובת הפנوية. במידה ולאחר מכן התוכנה תקבל אישור מצד הלוקה היא תשלח גם ACK.

חשוב לציין שהתוכנה שלנו מבצעת את העבודה בצורה הבסיסית ביותר. לדוגמה היא לא בודקת ומחררת כתובות במידה והלוקה התנטק או סירב לקבל כתובות. לכן כדי שהתוכנה הראשית תפעל כמו שמתוכנן אז כדאי להפעיל אותה לפני כל הריצה של הפרויקט. אחרת היא לא תוכל לספק כתובות.

בנוסח רצינו שהתוכנה לא רק תיתן את הכתובת ללוקה אלא גם תיתן לו כתובות של שרת DNS. מצאנו באינטרנט שניין לעשות זאת בשדה דינמי Options על ידי הוספה אופצייה חדשה. אך נתקלנו בבעיה. השדה נוצר כמו שצרכיך ואפילו היה אפשר לראות אותו דרך wireshark אבל בצד הלוקה לא הצלחנו להוציא אותו. לכן קיבלנו החלטה לשים את הכתובת של DNS בשדה siaddr בשכבת BOOTP כיוון שהיא פנויה במקרה שלנו.

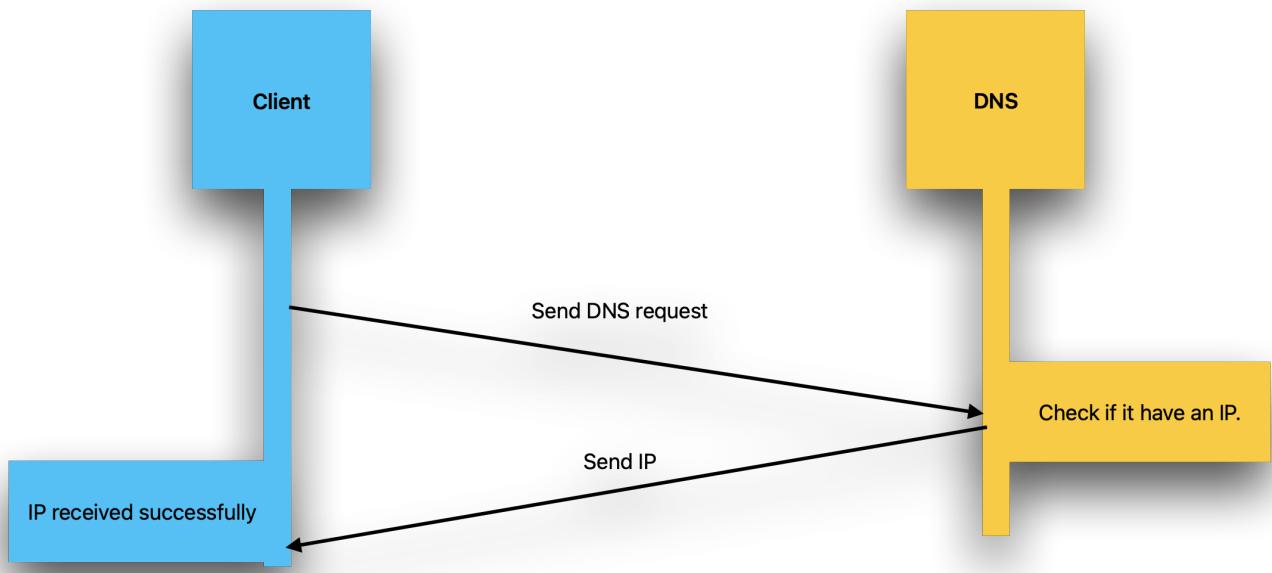
```
nikita@NikitaVM:~/Project/Other$ sudo python3 DHCP.py
[sudo] password for nikita:
Enter the name of the network interface manually:
enp0s1
Listening for DHCP requests...
DHCP request received
DHCP Discover received
Offering IP address: 127.0.0.5
.
Sent 1 packets.
Listening for DHCP requests...
DHCP request received
DHCP Request received
IP address: 127.0.0.5
.
Sent 1 packets.
DHCP ACK sent
Listening for DHCP requests...
nikita@NikitaVM: ~/Project$ sudo python3 Client.py
pygame 2.1.2 (SDL 2.0.20, Python 3.10.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter the name of the network interface manually:
enp0s1
Broadcasting DHCP request...
.
Sent 1 packets.
Waiting for DHCP offer...
DHCP offer received
IP address: 127.0.0.5
DNS server IP address: 127.0.0.3
.
Sent 1 packets.
DHCP ACK received
```

אם נריץ את wireshark כדי להסניף בקשות DHCP נראה如下:



DNS

DNS - זה שרת שמנהל כתובות IP ואת השמות שלפיהם ניתן למצוא אותם. כאשר לך רצוח להתחבר לשרת כלשהו ואני לו כתובות IP אלה יש שם, הוא שולח בקשה DNS וDNS יחזיר לך את הכתובת. התקשרות מתנהלת בפרוטוקול UDP.



אם כאן השתמשנו בספרייה Scapy.

כתובת השירות - 127.0.0.3:53. ניתן לשנות בשדות DNS_IP, DNS_PORT

כמו ב DHCP צריך להזין את שם המכשיר להאזנה.

בדומה לשרת DHCP התוכנה שלנו רק מדמה שרת DNS בצורה הconi בסיסית.

התוכנה מרים לולאה אינטגרית אשר מאפשרת לפקטוות מסוג UDP ופורט 53. מכיוון שפורט 53 הוא פорт סטנדרטי לשרת DNS אז התוכנה גם תתפוץ בקשות אחרות במקורה ויהי על המחשב אבל כיון שבנו מאגר רק עם כתובות אחת היא תענה רק עליה. תוכנה לא תחזיר שום תשובה לכל בקשה אחרת.

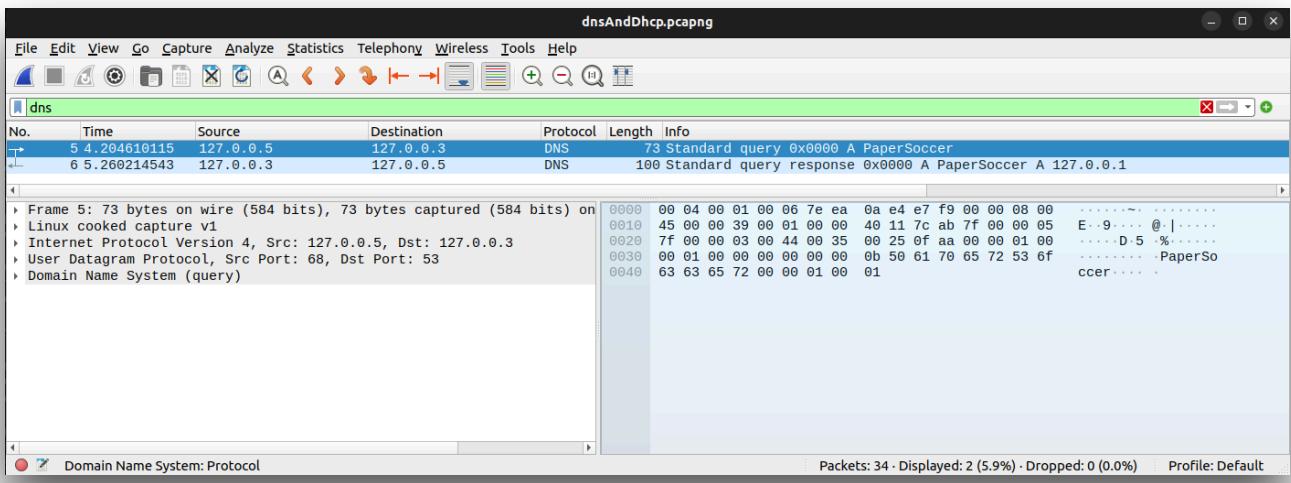
בהרצת wireshark ניתן לראות את התקשרות הרצוייה.

```

nikita@NikitaVM:~/Project/Other$ sudo python3 DNS.py
[sudo] password for nikita:
Enter the name of the network interface manually:
enp0s1
Listening for DNS requests...
DNS request received
Answering DNS request for PaperSoccer with 127.0.0.1:None
.
Sent 1 packets.
DNS request answered
  
```

```

nikita@NikitaVM:~/Project$ sudo python3 Client.py
pygame 2.1.2 (SDL 2.0.20, Python 3.10.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter the name of the network interface manually:
enp0s1
Broadcasting DHCP request...
.
Sent 1 packets.
Waiting for DHCP offer...
DHCP offer received
IP address: 127.0.0.5
DNS server IP address: 127.0.0.3
.
Sent 1 packets.
DHCP ACK received
Create DNS request...
.
Sent 1 packets.
Waiting for DNS response...
DNS response received
  
```



Client

- זו בעצם התוכנה של השחקן. מטרתה של התוכנה היא לדמות תקשורת עם שרתים DHCP וDNS ולאחר מכן להתחבר לשרת של המשחק כדי להתחיל לשחק. התוכנה מצירמת את הGUI של המשחק ושולחת פקודות לשרת על מנת לקבל מידע כדי לייצג אותו ב-GUI.

כדי לבנות את החלק הגרפי של המשחק השתמשנו בספריית PyGame. בהתחלה בנינו את המשחק נקי ללא שום קשר לשרתים. ניתן למצוא אותו ואת הפירוט [עליו כאן](#).

תקשורות מול השרת בנוייה בצורה הבא:
כל בקשה והודעה תשלוח ב프וטוקול TCP.
החלק הגדול הזה נתוני המשחק יתקבלו מהשרת בפרוטוקול RUDP.

התוכנה בנוייה מאربעה קבוצי פיתון MVC וממשת תבנית:
Client - הקובץ הראשי שמנוהל (Controller) את כל התוכנה כגון הפעלת GUI ותקשות עם השירותים.

View - קובץ שמנוהל את GUI באמצעות ספריית PyGame
Model - קובץ שמייצג את נתוני המשחק ושומר על חוקיו. מכיל שלושה קלאסים Point, Game, Field. כאשר לכל קלאס קיימת פונקציה שהופכת אותו לצורה בינרית וחזרה כדי להעביר את המידע בראשת.

Packet - קובץ שmagdir את הפקודות הנשלחות לשרת המשחק.

נפרט בקצרה על הפונקציות בכל קובץ

Client

get_ip_from_dhcp - פונקציה אחראית על דימוי של תקשורת עם שרת DHCP ומחזירה כתובותamen. משתמש בספריית Scapy.

get_game_ip - פונקציה אחראית על דימוי של תקשורת עם שרת DNS ומחזירה את הכתובתamen. משתמש בספריית Scapy.

connect_to_game - פונקציה שאחראית על התחברות לשרת המשחק ואיתחולו המשחק. פונקציה יוצרת סוקטים מסוג TCP ו-UDP. וגם יוצרת פקטת בקשת התחברות ושולחת אותה לשרת בפרוטוקול TCP. ולאחר מכן מאתחלת את GUI.

game_update - יוצרת פקטת בקשה לעדכון ושולחת אותה לשרת. לאחר מכן מפעילה את הפונקציה של קבלת נתוני המשחק. אם המידע והמשחק יתקבלו בהצלחה הפונקציה תעמעכן את הנתונים של התוכנה אחרת היא תפעיל עדכון של שרת על שגיאה ותפעיל בקשה חדשה לעדכון. חשוב לציין שפונקציה זו מופעלת רק אם השרת אישר עדכון לפני כדי לא להעיסס על השרת.

get_game_from_server - אחראית על קבלת נתוני המשחק בפרוטוקול TCP. (לא מופעלת)

- מעדכנת את השירות על כישלון בקבלת נתוני המשחק.
- אחראית על ייצור קשר עם השירות. מופעלת כאשר אנו רוצים לקבל נתוני המשחק מהשירות ב프וטוקול RUDP.

- אחראית על קבלת נתוני המשחק ב프וטוקול RUDP.
הfonkzia יוצרת קשר עם השירות על ידי ליחצת ידיים מושלמת. לאחר מכן מתחילה להאזין לפקודות שנשלחות מהשירות עד קבלת הودעה על סוף הפעולה. פקודות נראות כך:

Part	Sequence	First	Last
חלק נתוני המשחק	מספר פקטה ראשונה בחלון	מספר פקטה ראשונה בחלון	מספר פקטה אחרת בחלון

fonkzia תזעק אחריה בפקודות שהתקבלו בחלון ובמידע. כשתגלה על חוסר בפקטה תשלח NACK לשרת אחרי הסיום של החלון, אחרית היא תשלח ACK. במידה והfonkzia קיבל את כל החלקים של נתוני המשחק היא תחבר אותם ותשמר אותם ותחזיר True. אם fonkzia לא קיבל שום פקטה מסוימת במשך timeouut (בעמ"ס) אז היא תפסיק את הפעולה שלה ותחזיר False. מספר הפקודות המksamלי שfonkzia יכולה לשמור מוגדר להיות 64 בקרה דינית.

- אחראית על בקשת אישור לקבלת עדכון של המשחק. בכך היא מפחיתה את העומס על השירות. במידע ותקבל אישור היא תפעיל fonkzia בקשה לעדכון.

- שולחת את הגדרות המשחק לשרת. במקרה שלנו זה רק גודל הלוח (אורך ורוחב). אחרי זה מבקשת עדכון של המשחק. מופעלת רק אצל אחד המשתמשים.

- שולחת בקשה לשות מהלך במשחק ואחרי זה מבקשת עדכון. ורק שחקן שהתו שלו יכול להפעיל אותה.

- שולחת בקשה להתחיל משחק חדש לשרת ומתחילה את הGUI.

- מעדכנת את השירות על עזיבת הלוקו (כאשר משחק נסגר בצורה רגילה בליחיצה על א). לאחר מכן סגורת את הסוקטים.

Model , View

לא נפרט עליהם כיוון זהה פחות נגוע לקורס רשות תקשורת.

Packet

מכיל קלאס האב Packet שמכיל רק שדה message וfonkzia (serialize, deserialize) המmirות אותו לצורה בינארית וחזרה.
מכיל קלאסים PacketOptions,PacketMove,PacketGame שיורשים מקלאס האב ומרחיבים את היכולת שלו לשמר נתונים נוספים לפי הצורך.

- תוכנה אשר מייצגת את צד השירות. מטרת השירות הוא לנוהל את השחקנים, חוקי המשחק, לקבל ולשלוח עדכונים לשחקנים. כל שחקן מנוהל בתהליך נפרד.

כתובות של השירות:

מכיוון שבחרנו להשתמש בשני פרוטוקלים אז לשרת יש שני פורטים בהם משתמש כדי לתקשר עם השחקנים.

TCP 127.0.0.1:5000
RUDP 127.0.0.1:5001

השרת בנוי משלושה קבצים:

- קובץ ראשי אשר מנהל את כל העבודה של השירות.

- כמו אצל השחקן

- כמו אצל השחקן

גם כאן נפרט בקצרה על הפונקציות:

add_player - מוסיפה שחקן חדש לרשימה (רק שני אנשים יכולים להיות ברשימה). כדי לשמר את הכתובת (TCP) של השחקן ולקשר אותו לצבע.

client_handler - פונקציה שמאזינה לביקשות ועדכונים מצעד השחקנים. במידה ומקבלת בקשה מפעילה פונקציה לניהול בקשות.

proces_request - הפונקציה המכילה מפחידה בתוכנה מצטערים מראש. מטרתה היא לנוהל את הביקשות של השחקנים לפי הבקשות ולהפעיל פונקציות אחרות כדי לעדכן את נתוני המשחק או לשולח נתונים לשחקנים במידה ויש צורך.

- במידה ומשחק נגמר מתחילה אותו.

- שולחת את נתוני המשחק בפרוטוקול TCP (לא מופעלת).

three_way_handshake - מאזינה לניסיונות התחברות לשרת מצד RUDP. מופעלת ברגע שצריך לשולח את נתוני המשחק כדי ליצור קשר עם הלוקו. אם היא לא מקבלת שום בקשה מסpter פעומים היא תפסיק את הפעולה ולא תחזיר כלום. אחרת תחזיר את כתובות UDP של השחקן.

send_game_rudp - פונקציה שמנוהלת את השיליחה של נתונים המשחק בחלוקת קטנים על מנת לייצג עבודה של פרוטוקול RUDP. יוצרת קשר עם שחקן על ידי לחיצת ידיים משולשת במידה ולא נוצר קשר מפסקה לעבוד. אחרי יצירת קשר מפצלת את נתונים המשחק (בצורה ביןארית) לחלקים קטנים ומתחילה לשולח אותם.

השליחה מתבצעת בחלונות. חלון הוא כמהות הפקות שפונקציה שולחת עד קבלת אישור ACK. פונקציה מתחילה עם חלון בגודל 1. במידה וכל הפקות בחלון נשלחו בהצלחה (קיבלו אישור מצד הלוקו) גודל החלון יגדל פי 2 במידה זה אפשרי או 1. במידה ואי אפשר להגדיל את הגודל החלון אז הוא לא ישתנה. שדה threshold אחראי על הגבלת גודל החלון. בתחילת הריצה הוא שווה לכמות הפקות. במידה ושיליחה של החלון לא הצליחה כי נאבדה פקטה וקיבלו NACK מлокו, או שלא קיבלנו תשובה מלוקו כלל אז נקבעו את גודל החלון פי 2 ואת threshold פי 2 וננסה לשולח את החלוקי החלון הנכשל שוב. אם קרה מצב שמספר פעמים ניסינו לשולח את אותם הפקות ולא הצליחנו אז הפונקציה תפסיק את הפעולה ותצא.

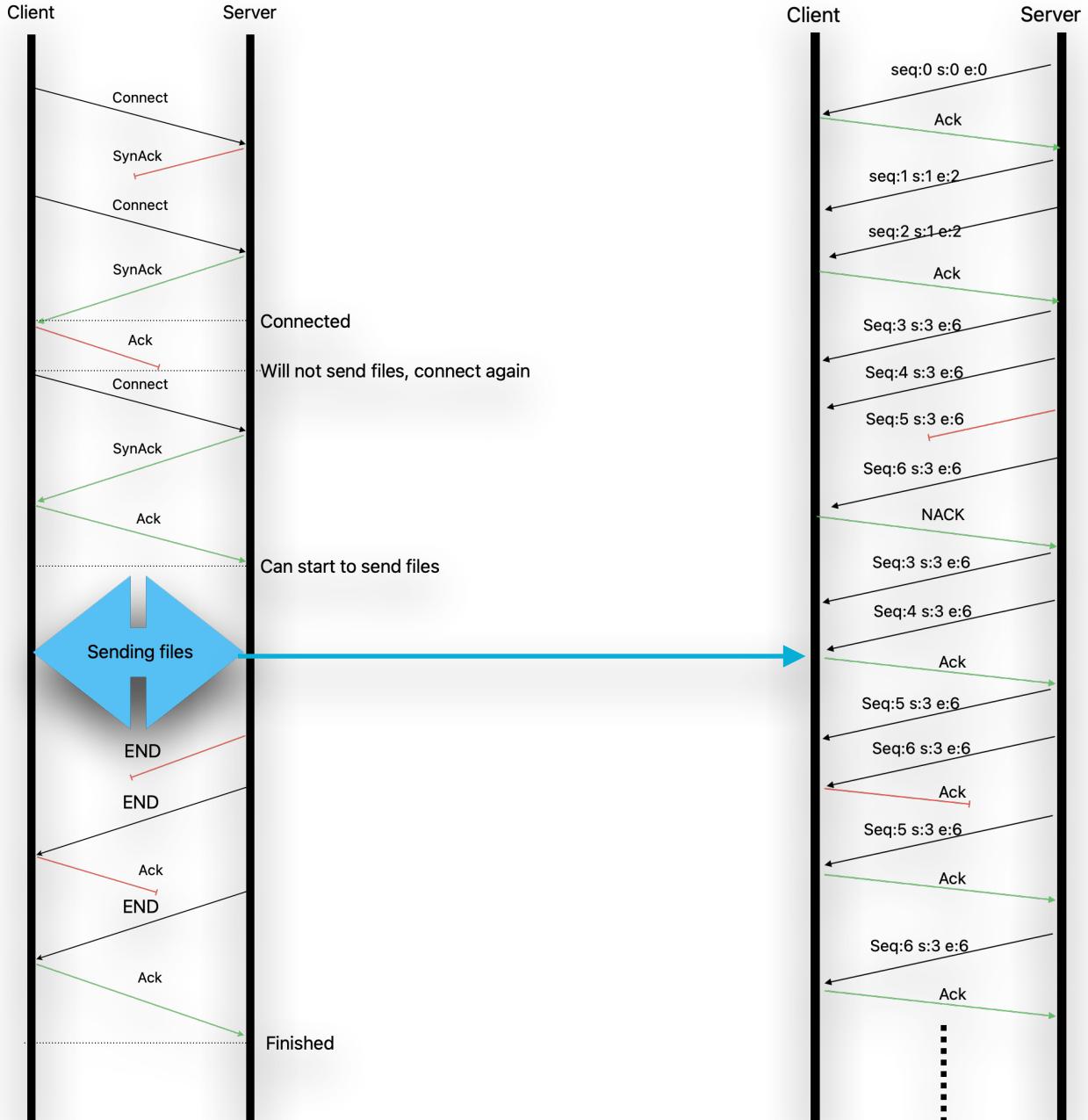
אם סיימנו לשולח את כל הפקות נשלח הודעה על כך ללוקו ונמתין לאישור. במידה ולא קיבלנו אישור ננסה עוד כמה פעמים. אם גם אחרי כמה פעמים לא קיבל נפסק את הריצה.

isNeedToUpdate - בודקת אם קיימים עדכונים לשחקן. במידה וכן משנה את זה ללא ומחזירה אמת אחרית תחזיר שקר.

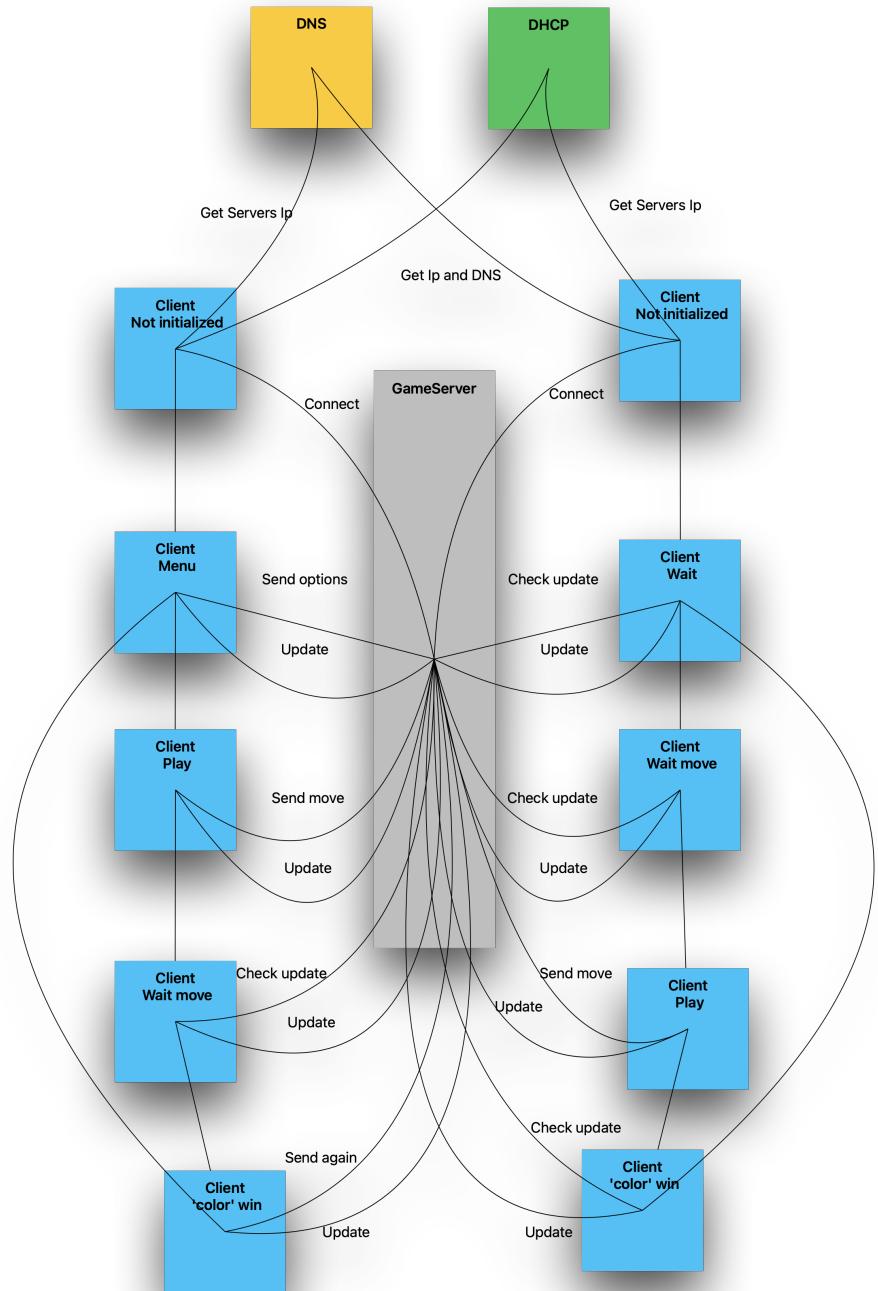
switchPlayer - מחליפה תור במידה ויש צורך

- בודקת אם מותר לחבר עוד שחקן. מחזירה אמת אם כן אחרת שקר

דיאגרמת שליחת נתונים המשחק בפרוטוקול RUDP



דיאגרמת המצבים של המשתמשים



הפעלת הפרויקט

לפניהם הפעלת התוכנה חשוב לוודא שעל המחשב קיימות הספריות הבאות:

Scapy 2.5.0 •
PyGame 2.1.2 •

קודם נפעיל את השירותים:

```
sudo python3 DHCP.py
```

DHCP •
• נמצא ב/
Project/Other
להריץ עם הרשות אדמין כדי לקבל גישה למכשיר
נכנייס את שם המכשיר בתחילת התוכנה
DNS •

```
sudo python3 DNS.py
```

אותו דבר
GameServer •

```
python3 GameServer.py
```

• נמצא ב/
Project
אין צורך בהרשות אדמין

```
sudo python3 Client.py
```

• Client
• נמצא ב/
Project
להריץ עם הרשות אדמין כדי לקבל גישה למכשיר
נכנייס את שם המכשיר בתחילת התוכנה
שרת יכול לקבל עד שני משתמשים בו זמן

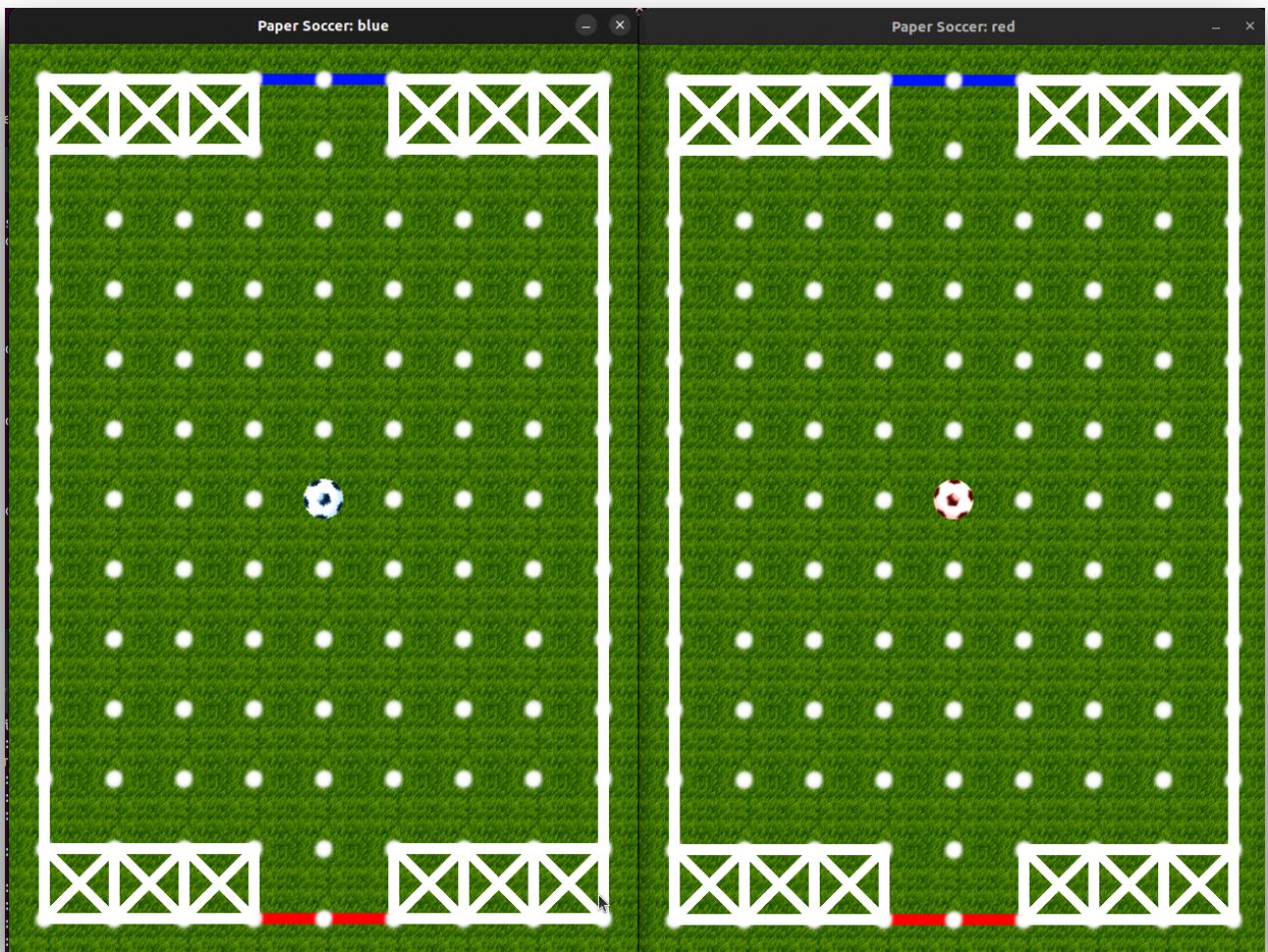
חשוב לציין שככל התוכנות פועלות על כתובת 127.0.0.1 . לכן אם נריץ הכל במכשורים שונים לא יוצר חיבור. ניתן לשנות זאת רק עם הזנה ידנית בשדות המתאים בתוכנות.

במידה והכל רץ תקין לכל משתמש אמרור להפתח חלון GUI.



אחד המשתמשים יהיה אחראי על להזין את נתוני המשחק והשני ימתין כאשר הצד שלו אוטומטי יבקש עדכונים מהשרת.

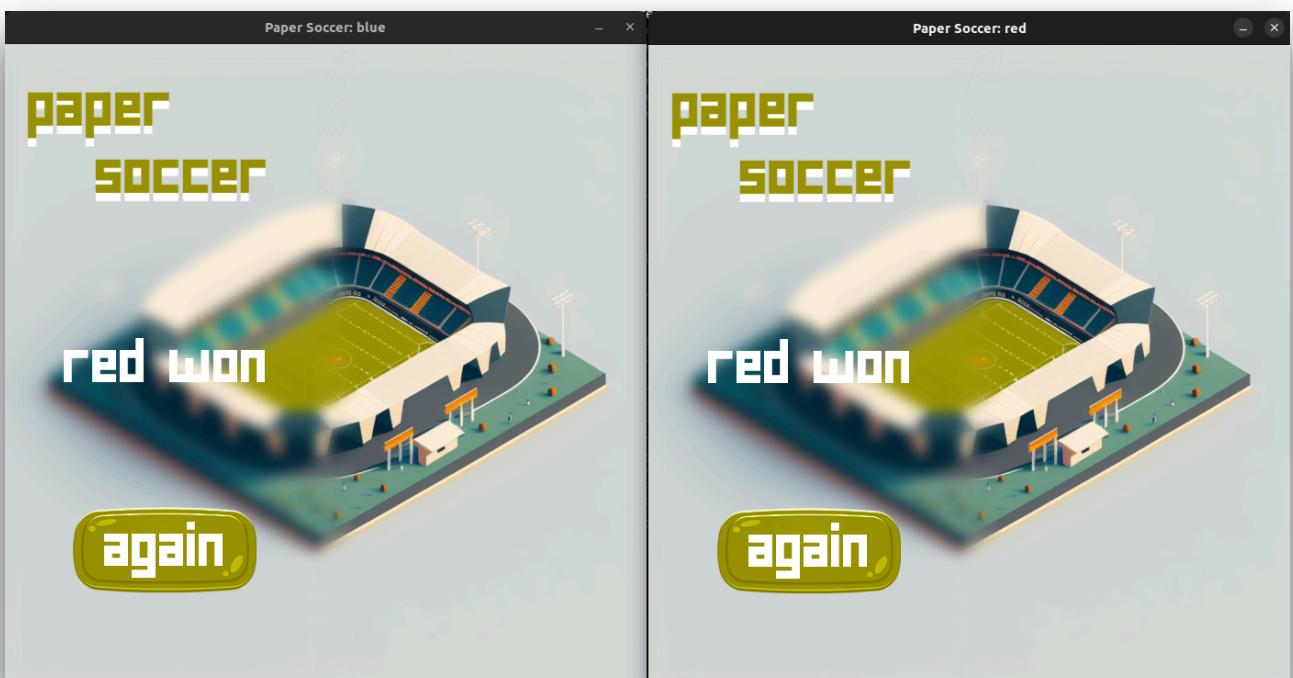
לאחר ההזנת נתונים משחק יפתח חלון המשחק לשני המשתמשים.



במשחק יכול להתקדם רק אותו שחקן שהטור שלו. השחקן השני ימתין ויבקש עדכונים מהשרת בצוורה אוטומטית. ניתן ליזהות שהטור שלך אם כדור התחיל להסתובב.

במידה ואחד השחקנים יחליט להתנתק מהמשחק. השני ייחכה לו וכאשר שחקן יתחבר שוב הוא ימשיך מאותו מצב. אם שני השחקנים יתנתקו המשחק ית/apps. זה יקרה רק אם הניתוק היה חוקי כלומר דרך כפטור X בפינה, לאחרת תהיה תקלה.

כאשר המשחק יסתום השחקנים יכולים לבחור על המשחק החוזר ומשחק י חוזר לפרטיו ההתחלתי.



המעקב אחרי ריצת התוכנה דרך Wireshark

נՐץ את התוכנה וננסה להגיע לכל מצבים המשחק תורן כדי נקליט את התוצאות.

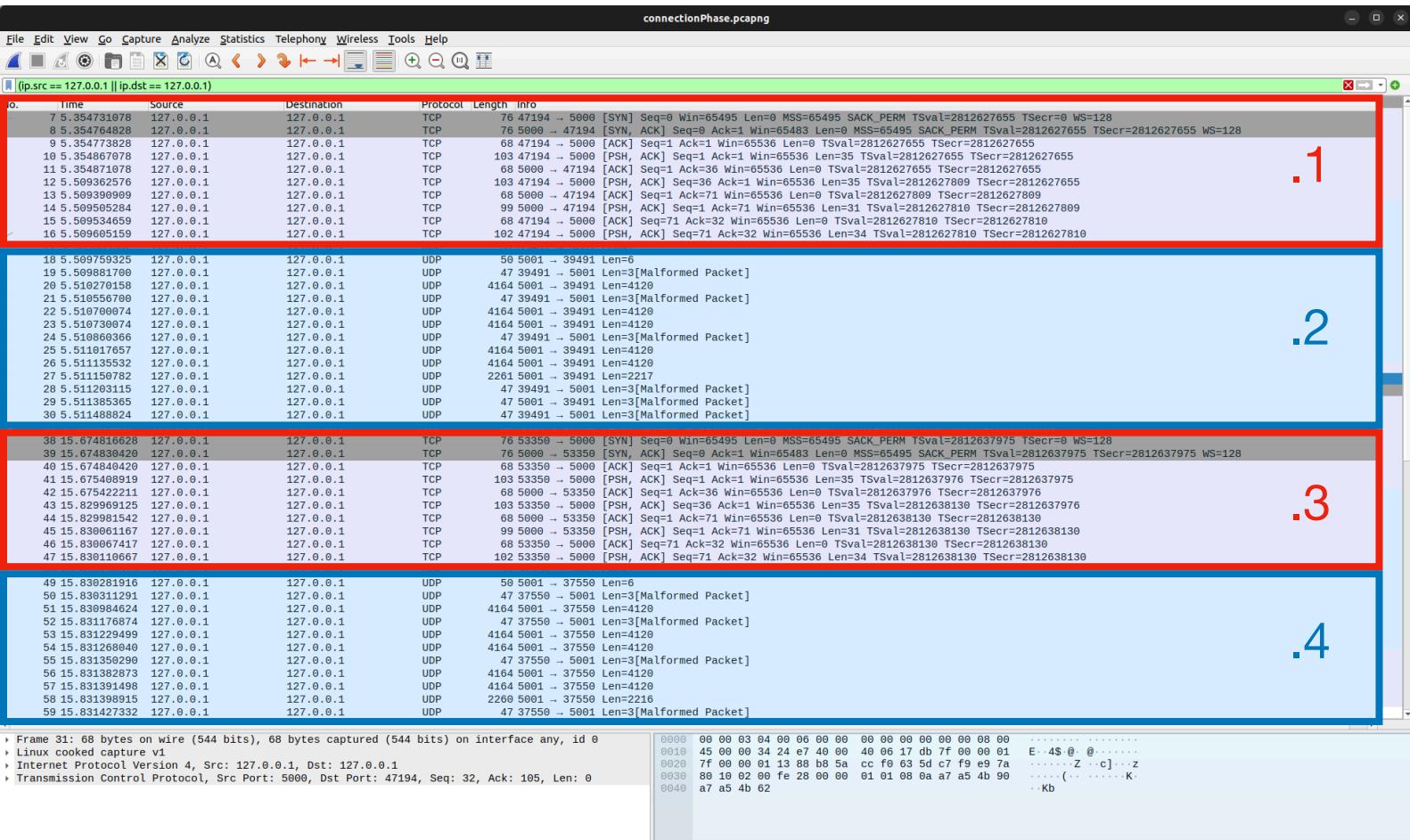
הסינון בWireshark

or dns or dhcp (ip.src == 127.0.0.1 || ip.dst == 127.0.0.1)

את הצלומים של שלב DHCP DNS ניתן היה לראות קודם.

שלב ההתחברות של השחקנים

ניתן לראות דרך wireshark את הפקודות שעוברות בין השרת לשחקנים. כמו שתכננו כל הבקשות והודעות מתבצעות ב프וטוקול TCP. וכאשר השרת מעביר את נתוני המשחק הוא עושה זאת בפרוטוקול RUDP.



1 - התחברות של השחקן הראשון לשרת

מעביר לשרת בקשת ההתחברות
לאחר מכן מעביר שאלה אם ניתן לקבל עדכון
מעביר בקשה לעדכון

2 - שחקן ראשון מקבל את נתוני המשחק (משחק עד לא מוחלט אך צריך להעביר את התפריט)

3 - פקודות ראשונות זו יוצרת קשר (לחיצת יד משולשת)

אחרי זה העברת פקודות יחד עם ACKs. כיוון שאין איבוד אז ניתן לראות שהחalon גדול. סה"כ 6 פקודות הועברו עם 3 אקים.

ואחריו זה קיבל עדכון על סיום העברת ושלח ACK

3 - התחברות של השחקן השני

4 - שחקן שני מקבל את נתוני המשחק

כל השלבים האחרים של המשחק עובדים בדיק באוותה הצורה لكن לא נפרט עליהם כאן. אך נעשו הקלטה אחת של משחק שלם עם כל המជבים.

הרצה של התוכנה עם איבוד פקודות

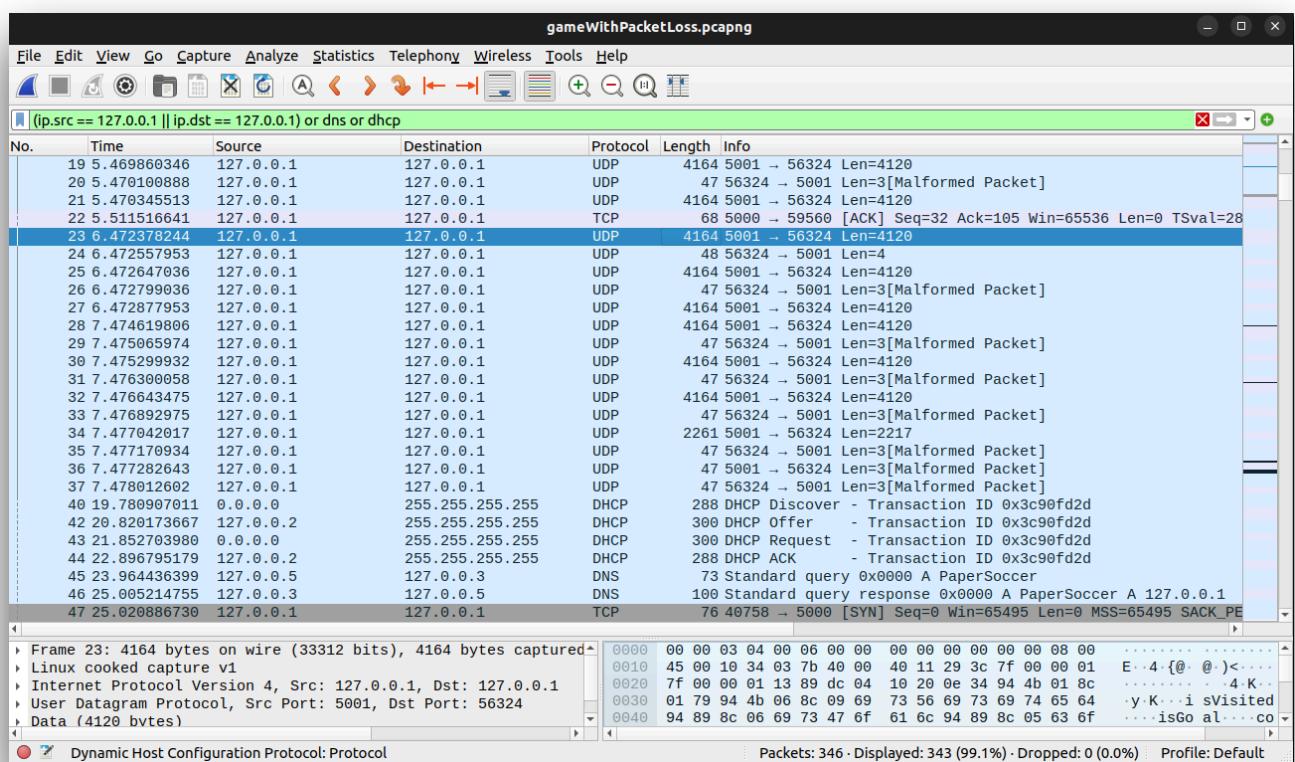
נ裏ץ את התוכנה שלנו עם איבוד פקודות ונראה אם ה프וטוקול RUDP שכתבנו יכול לשמר על המשחק.

על מנת להפעיל איבוד נ裏ץ את הפקודה הבא:

```
sudo tc qdisc add dev lo root netem loss 5%
```

התוכנה רצה בהצלחה ורק העברת פקודות לקחה קצר יותר זמן.

ניתן להזיהות שהייה איבוד בהקלטה של wireshark. כאשר הודעות tcp שנאבדו יסומנו. את ההודעות RUDP ניתן רק להזיהות לפי ספירת הפקודות ושינוי גודל החalon (גם ניתן לראות הודעות בשני הצדדים בקונסולה).



ניסינו לה裏ץ גם איבודים יותר חזקים. נראה שההעברת הצלילה בהם אך עקב שיבושים בזמן ועיקובים נוצרו מצבים לא צפויים במהלך המשחק.

לדוגמא שני השחקנים ביקשו עדכון. אחד קיבל והשני לא הספיק לקבל זאת מצב המשחק השתנה, שכן

כאשר הוא קיבל את המשחק שוב, הוא קיבל אותו במצב הלא נכון ולא הצליח המשיך.

תקלות אלה נובעות מחוסר ניסיון שלנו במבנה פרויקטים גדולים ומשחקי רשות במילוי.

תשובות לשאלות בסוף ההניות:

א) הבדלים בין פרוטוקולים tcp לcubic:
1. יצרת חיבור בין לקובו לשרת:

פרוטוקול tcp דורש לחיצת יד משולשת על מנת ליצור חיבור. תהליך זה כולל שלוש אקטואות שנשלחות בין הלקובו לשרת. הלקובו שולח לשרת אקטת syn המצביעת על כך שהוא מעוניין ביצירת קשר. לאחר מכן השרת מוחזר כתשובה אקטת ack המצביעת על כך כי השרת מוכן ליצור חיבור. ולבסוף הלקובו שולח לשרת אקטת ack המצביעת על כך שהוא קיבל את תגונת השרת והחיבור נוצר. לעומת זאת, ב프וטוקול cubic משתמשים בපתקה אחת על מנת ליצור את החיבור בין הלקובו לשרת, ובכך מופחתים עיכובים והחיבור נוצר הרבה יותר מהר.

2. הזרמת מידע:

פרוטוקול tcp מאפשר רק שליחה של סוג מידע אחד לכל חיבור. בניגוד לפרטוקול cubic המאפשר לשולח כמה סוגים של מידע על כל חיבור אחד בין הלקובו לשרת. זה בא לידי ביטוי בכך שלדוגמא הלקובו ב��יש לאתר מסוימת, בחיבור tcp כל פעם ישלח ללקח חלק אחר של אתר בעודו בבחירה cubic יכולים להישלח כמה חלקים של האתר בפעם אחת.

3. איבוד אקטואות:

כשאנבדת אקטת בחיבור tcp, מעדכנים את הצד השני שאבדה אקטת ומוחכים לאקטת זו עד שתישלח שוב. בתוצאה לכך מוחכים גם לשאר הפקטאות שצרכות להישלח לאחר האקטת שלא הגיעו ליעדה. לעומת זאת בפרטוקול cubic, השולח מוסיף מידע לאקטואות שנשלחות כך שאם תאבד אקטת בדרך הנמען יוכל להשתמש במידע הנוסף שנשלח כדי להתגבר על איבוד האקטת. בדרך זו מוחכים את זמן המתנה לשילחה נוספת של אותה אקטת שאבדה כמו בפרטוקול tcp.

4. אבטחת מידע:

לכט tcp אין אבטחה מידע מובנית בתוכו וצריך פרוטוקולי הגנה חיצונית על מנת להגן על המידע הנשלח. זאת בשונה מtcp שמכיל בתוכו הצפנה ששומרת על המידע.

ב)cubic וvegas הם אלגוריתמים שתפקידם הוא לשלוט על העומס של התעבורה ברשות על מנת למנוע עיכוב ואיבוד אקטואות בדרך. שני הבדלים מרכזים ביניהם:

1. בסיס האלגוריתם:

אלגוריתם cubic מבוסס על פונקציית cubic.

אלגוריתם vegas מבוסס על RTT (round trip time) של אקטת.

2. אופן הפעולה:

ב)cubic שולחים אקטואות בקצב איטי ומגבירים את קצב השילחה מהר. כאשר מזוהה איבוד אקטואות, קצב שלילה הפקטאות יורד והתהליך חוזר חלילה. לעומת זאת, בvegas משוימים את זמן שלילה הפקטת (RTT) בזמן השילחה המצופה של אקטת (זה נקבע על ידי מדידות קודמות של RTT). אם יש עלייה בזמן שלילת הפקטת, כלומר זמן השילחה גדול יותר מאשר השילחה המצופה מניהים כי יש עיכוב ומפחיתים את קצב שלילה הפקטאות. בvegas מורדים את העומס לפני שהאיבוד קורה ואילו בcubic האיבוד קורה וזה מורדים את העומס.

ג) פרוטוקול BGP (border gateway protocol) הוא פרוטוקול שאחראי על ניתוב בין AS (Autonomous Systems - חלוקת הרשת הגדולה לתתי רשתות). BGP הוא פרוטוקול Distance Vector, שזה אומר כי כל נתב מכיר את הנתבים השכנים לו וכל חישוב של מסלול קצר מתבצע על ידי אלגוריתם בלמן פורד.

נתבים מחליפים ביניהם מידע באמצעות TCP בפורט 179. כל AS מהוZA צומת ניתוב שמשתמש בפרטוקול BGP כדי לבנות מסלולי ניתוב דינמיים כדי להתחבר מול AS אחרים. בשונה מmeno OSPF (Open Shortest Path First) הוא פרוטוקול ניתוב היררכי להעברת נתונים בין רואטרים שונים הנמצאים באותו AS. הפרוטוקול הוא link state, זאת אומרת נתב מכיר את כל הרשת. מבוססת הפרטוקול עומדת אלגוריתם דיקסטרה. OSPF משתמש בעלות העברת הנתונים (מספר המציג את גודל רוחב הפס) לצורך חישוב המרחק, ותמיד יבחר את הנתיב הזרול ביותר להעברת חבילת מהמקורה אל היעד.

(ה) הבדלים בין פרוטוקול arp לבין dns:

1. מטרה:

פרוטוקול arp משמש כדי לקשר כתובות mac של תחנות ברשת המקומיות לכתובות ip. לעומת dns המשמש לקישור בין שמות יותר נוחים לאדם (כמו כתובת האתר) לכטבות ip (של אותו האתר).

2. לאיזה שכבה כל אחד שייך:

פרוטוקול arp שייך לשכבות הeth0. בשונה מזהם שייך לשכבת האפליקציה.

3. תחום הפרוטוקול:

בკארפ מפענחים כתובות mac של מכשירים הנמצאים רק ברשת המקומיית. לעומת זאת בזאפ מפענחים כתובות של אתרים בכל האינטרנט.

4. אופן פעולה:

בקארפ שלוחים הודעה broadcast לכל המכשירים באותו הרשת המכילה את כתובת הקו של המכשיר שרצו להגיע אליו. אותו מכשיר עם הקו המבוקש יענה להודעה זו בכך שישלח את כתובת mac שלו. כל מכשיר ישמר בתוך טבלה את כתובות mac והקו של המכשירים האחרים באותו הרשת שכבר מצא. זאת על מנת לאפשר שליחת הודעות בין המכשירים ללא הצורך שוב לאתר את כתובת הקו לכתובת mac הנכונה. וכך התקשורת בין המכשירים תהיה מהירה יותר. לעומת זאת, dns היא שיטת רישום היררכית. תהליך הבירור לגבי שם של אתר מסוים נעשה על ידי פניה לשרת וקבלת הפניה לשרת אשר כל הנראה יוכל לספק את התשובה, או לפחות לקרב אותו לשרת אשר ידע להסביר ובו התהליך יჩזור על עצמו. התהליך מסתיים כאשר מתאפשרת תשובה מדויקת משרת אשר אחראי לשם האתר המבוקש, או משרת אשר אינו אחראי אבל זוכרו שלו מכיל את התשובה.