אוניברסיטת אריאל בשומרון, המחלקה למדעי המחשב רשתות תקשורת, סמסטר ב' תשפ"ג

PING (ICMP & TCP)

ליאור וינמן יועד תמר 213451818 213081763

2023 בינואר 3

תוכן עניינים

2													 	 	 											ת	5-	עו	הכ	ור	יא	n -	'n	١ آ	פרכ)	1
2																																זרצ		·	1.1		
3																																זקו			1.2	2	
4																																<u>.</u> מימו			1.3	3	
7																																מימו			1.4	ŀ	
9																																מימו			1.5	5	
12																																ר ת		ī	פרכ)	2
12																																פינג			2.1		
12				 	 	 											•	 	 			1	C	'A	IF	1	רח	בו	תע			2.1.	1				
13				 														 	 					ת	בני	נו	הו	v	פל			2.1.2	2				
14																															מו	פינג)		2.2	2	
14																													תע			2.2.					
15				 														 	 				. 7	Γ	CP	1	רח	בו	תע			2.2.2	2				
17				 														 	 					ת	בני	נו	הו	v	פל			2.2.3	3				
18			 	 		 													 				í	ה	ציו	ע		ע	בלל	שוכ	מו	פינג)		2.3	3	
18	•			 													•		 			1	C	'A	1 P	1	רח	בו	תע			2.3.	1				
19	•			 													•		 				. 7	Γ	CP	1	רח	בו	תע			2.3.2	2				
20				 												•			 					ת	בני	נו	הו	v	פל			2.3.	3				
21																																מנה	٦1,	רנו	7)-	1	3

1 פרק א' - תיאור המערכת

1.1 הרצת הפרויקט

(שאר הקבצים הם קבצי $new_ping.c$ ו ping.c הם הם הרצה הקבצים הניתנים להרצה הם ping.c

על מנת להריץ את הפרויקט - עלינו קודם כל להוריד את כל ארבעת הקבצים לתוך תיקיה משותפת על המחשב על מנת להריץ את הפעלה $Linux\ Ubuntu\ LTS\ 22.04$ בלבד).

 $"make \ all"$ הפקודה את הקבצים ונכתוב את התקיה הנתונה על התקיה תרמינל על התקיה הנחונה את הקבצים ונכתוב את הפקודה "clear" ההרצה של התוכניות הניתנות להרצה. (לאחר מכן מומלץ להריץ את הפקודה "clear" כדי לנקות את הפלט הסטנדרטי מהטרמינל).

כעת, אם ברצוננו להריץ את ping.c את הפקודה "sudo ./PartA < ip >" מכתוב בטרמינל את הפקודה "ping.c את הריץ את הפקודה אליה אנחנו מעוניינים לשלוח את הבקשות). אם אנו מעוניינים ואם ברצוננו להריץ את ip > ip > ip. ip > ip > ip > ip. נכתוב בטרמינל את הפקודה "ip > ip > ip > ip > ip > ip.

לאחר שכתבנו את אחת מהפקודות שמלעיל, התוכנית פשוט תתחיל לרוץ (כלומר ישר תתחיל לשלוח ולקבל תעבורת פינג).

Makefile - הקובץ היוצר 1.2

הקובץ הנ"ל הוא הקובץ שממיר את קבצי הקוד (הכתובים בשפת C) ומייצר מהם קבצי הרצה על מנת שנוכל להריץ בפועל את התוכניות שאנו כותבים. נעבור על הקוד:

```
CC = gcc
FLAGS = -Wall -g
.PHONY: all clean
all: ping new_ping watchdog
ping: ping.o
        $(CC) $(FLAGS) -o PartA ping.o
new ping: new ping.o
        $(CC) $(FLAGS) -o PartB new_ping.o
watchdog: watchdog.o
        $(CC) $(FLAGS) -o watchdog watchdog.o
ping.o: ping.c
        $(CC) $(FLAGS) -c ping.c
new ping.o: new ping.c
        $(CC) $(FLAGS) -c new_ping.c
watchdog.o: watchdog.c
        $(CC) $(FLAGS) -c watchdog.c
clean:
       rm -f *.o PartA PartB watchdog
```

בקובץ זה ראשית אנו מייצרים מקבצי הקוד קבצי אובייקטים בינאריים (שורות 24-17) תהליך זה הוא תהליך ההידור. לאחר מכן מהקבצים הבינאריים אנו מייצרים קבצי הרצה שבעזרתם אנו למעשה מריצים את הפרוייקט (שורות 15-8) תהליך זה הוא תהליך הלינקוג'.

1.3 מימוש פינג - 1.3

נעבור על הקובץ *ping.c*:

בכל חבילה הנשלחת של המיועד לביטים אשר מאמתים האם החבילה הגיעה בשלמותה ביטים אלו נקראים בכל חבילה הנשלחת של ה־checksum כאן כתבנו פונקציה לחישוב ולהשמה של ה-checksum

:ping אליה לבצע אכן הריץ את התוכנית כנדרש, ז"א נתן לתוכנית אכן הריץ אליה לבצע אליה לבצע אליה לבצע אליה אליה לבצע

```
if (argc != 2) // checking that the user has specified an IP address

for a printf("usage: ./partb <ip>\n");

exit(EXIT_FAILURE);

}
```

כאן סידרנו את החבילה שאנו מעוניינים לשלוח, אנו שולחים "...000" (אין סיבה מיוחדת לדבר, סתם חבילה דיפולטיבית):

```
for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent

for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we
```

כאן פתחנו שקע שדרכו תעבור תעבורת פינג (raw socket) כאן

```
sock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP); // creating an RAW socket for ICMP communication
if (sock <= 0) // checking if socket created

{
    perror("socket() failed");
    close(sock);
    exit(errno);
}

addr_ping.sin_family = AF_INET; // setting up the struct for ICMP communicationg
if(inet_aton(argv[1], &addr_ping.sin_addr) < 0)

{
    perror("inet_aton() failed");
    exit(errno);
}
</pre>
```

כאן אנו מסדרים את ההגדרות של החבילות הנשלחות:

```
memset(&icmp, 0, sizeof(icmp)); // setting up the struct and packet for ICMP communication
memset(&packet , 0 , sizeof(packet));
icmp.type = ICMP_ECHO;
icmp.un.echo.sequence = seq;
icmp.checksum = 0;

memcpy((packet), &icmp, ICMP_HDRLEN);
memcpy(packet + ICMP_HDRLEN, data, data_len);

icmp.checksum = checksum(&packet, sizeof(packet));
memcpy((packet), &icmp, ICMP_HDRLEN);
```

כאן אנו שולחים בקשת פינג לכתובת ולאחר מכן מקבלים תשובת פינג. בנוסף, אנו מחשבים את הזמנים של כל התהליך:

```
gettimeofday(&start , NULL); // starting counting ping-time
if (sendto(sock, &packet, ICMP_HDRLEN + data_len, 0, (struct sockaddr*)&addr_ping, sizeof(addr_ping)) < 0) // sending ICMP-ECHO-REQUEST

{
    perror("sendto() failed");
    close(sock);
    exit(errno);

}

addr_len = sizeof(addr_ping); // receiving ICMP-ECHO-REPLEY
bzero(buffer, IP_MXMPACKET);

len = recvfrom(sock, buffer, buffer_len, 0, (struct sockaddr *)&addr_ping, &addr_len);

if (len == -1)
{
    perror("recvfrom() failed");
    close(sock);
    exit(errno);

gettimeofday(&end , NULL); // ending counting ping-time

time = (end.tv_sec - start.tv_sec) * 1000.0 + (end.tv_usec - start.tv_usec) / 1000.0; // saving the time in mili-seconds</pre>
```

כאן אנו מקבלים את הכתובת ממנה החבילה נשלחה כתגובה ומדפיסים למשתמש, בנוסף מבצעים המתנה של שניה:

```
ip = (struct iphdr*)buffer;

ip = (struct iphdr*)buffer;

printf("%d bytes has been recv from %s to " , len, inet_ntoa(*(struct in_addr*)&ip->saddr));

printf("%s: icmp_seq=%d ttl=%d time=%.2f ms\n", inet_ntoa(*(struct in_addr*)&ip->daddr) , icmp.un.echo.sequence , ip->ttl, time);

seq++;

seq++;

sleep(1);
```

לבסוף אנו סוגרים את השקע שפתחנו ויוצאים בהצלחה מהתוכנית.

$new_ping.c$ מימוש פינג משוכלל 1.4

:new_ping.c נעבור על הקובץ

קובץ זה הוא הרחבה לקובץ הקודם, לכן רוב הדברים זהים לחלוטין למה שכבר ראינו (אזי לא נפרט עליהם שוב). נציג רק את התוספות החדשות של הקוד: watchdog (המתבצעת עם ה־watchdog בפורט 3000) ומגדירים את האובייקט שיחזיק את הכתובת של התקשורת:

```
sockfd = socket(AF_INET, SOCK_STREAM, 0); // creating socket for TCP communcation
if(sock <= 0) // checking if socket created

{
    perror("socket() failed");
    close(sock);
    close(sockfd);
    exit(errno);
}

memset(&addr_server, '\0', sizeof(addr_server)); // setting up the struct for TCP communication
addr_server.sin_family = AF_INET;
addr_server.sin_port = htons(PORT);
addr_server.sin_addr.s_addr = inet_addr(IP);</pre>
```

כאן אנו מריצים את ה־*watchdog* (מריצים כעוד תהליך במקביל):

```
pid = fork(); // starting the watchdog proccess
if (pid == 0)

{
    execvp(args[0], args);
}
```

כאן אנו ממתינים שניה עד שה־watchdog יהיה מוכן לקבלת תקשורת, לאחר מכן אנו מתחברים אליו ושולחים אליו את כתובת ה־IP שהמשתמש הזין:

כאן אנו אמורים לקבל הודעת Watchdog מה־Watchdog שיש תקשורת:

```
if(recv(sockfd , buffer , BUFSIZ , 0) <= 0) // receiving an OK message from watchdog

{
    perror("recv() failed");
    close(sock);
    close(sockfd);
    exit(errno);

}

if(strcmp(buffer, OK) != 0) // checking that OK received

printf("error occurred!");
    close(sock);
    close(sockfd);
    exit(EXIT_FAILURE);
}</pre>
```

כאן אנו שולחים בחזרה הודעת OK וכמו כן בודקים מה מצב התהליך שרץ ברקע:

```
if(send(sockfd, OK, strlen(OK) + 1 , 0) < 0) // sending OK message

{
    perror("send() failed");
    close(sock);
    close(sockfd);
    exit(errno);

}

wait(&status);

if(status == -1)

{
    printf("proccess status failed\n");
    close(sockfd);
    exit(EXIT_FAILURE);

}</pre>
```

חוץ מהדברים שפירטנו מלעיל, שאר הקובץ זהה לחלוטין (שכן הוא רק הרחבה לקובץ הקודם), לכן לא פירטנו על שאר הדברים.

watchdog.c מימוש טיימר 1.5

הקובץ הנ"ל הוא למעשה קובץ בדיקת תקינות כתובת IP שהמשתמש הזין. הבדיקה מתרחשת על ידי הפעלת טיימר של 10 שניות של המתנה לתשובה מהכתובת (אחרי ששלחנו פינג בקשה) ואם לא התקבלה תשובה, אזי התוכנית תיעצר.

:watchdog.c נעבור על הקובץ

כאן כתבנו פונקציה אשר בודקת האם עבר הזמן והאם צריך לעצור או לא:

```
void timer_callback()

{
    if (!received_echo_reply)

{
       printf("server <%s> cannot be reached.\n" , buffer); // printing unreach message

kill(0 , SIGKILL); // killing all proccesses and exiting
}

}
```

כאן פתחנו את השקע שדרכו תעבור התעבורה (TCP)

```
server_sock = socket(AF_INET, SOCK_STREAM, 0); // creating the listener socket
if(server_sock <= 0) // checking if socket created

{
    perror("socket() failed");
    close(server_sock);
    exit(errno);
}</pre>
```

כאן אנו מאפשרים שימוש חוזר בפורט ובכתובת ומגדירים מבנה שיחזיק את הכתובת והפורט לתקשורת:

```
if(setsockopt(server_sock, SOL_SOCKET, SO_REUSEADDR, &er, sizeof(er)) < 0) // checking if ip and por

feetsockopt() failed");

close(server_sock);

exit(errno);

}

server_addr.sin_family = AF_INET; // setting up the struct for TCP communication

server_addr.sin_port = htons(PORT);

server_addr.sin_addr.s_addr = inet_addr(IP);</pre>
```

כאן אנו מקשרים את הכתובת והפורט לשקע שפתחנו ומאזינים לתקשורת נכנסת, אם יש כזו מאשרים:

```
if(bind(server_sock, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) // binding socket with settings
{
    perror("bind() failed");
    close(server_sock);
    exit(errno);
}

if(listen(server_sock, CONNECTIONS) < 0) //listen to incoming connections
{
    perror("listen() failed");
    close(server_sock);
    exit(errno);
}

addr_size = sizeof(client_addr);
client_sock = accept(server_sock, (struct sockaddr*)&client_addr, &addr_size); //accept a connection
if(client_sock <= 0) // checking if accepted
{
    perror("accept() failed");
    close(client_sock);
    close(server_sock);
    exit(errno);
}</pre>
```

:OK אנו מקבלים מהמשתמש את כתובת ה־IP ושולחים הודעת

כאן אנו מפעילים טיימר ל-10 שניות, לאחר מכן בודקים האם יש צורך לעצור את התוכנית:

```
timer.it_value.tv_sec = 10; // setting up timer for 10 seconds
timer.it_value.tv_usec = 0;
timer.it_interval.tv_sec = 0;
timer.it_interval.tv_usec = 0;

printf("waiting for ping response...\n");

setitimer(ITIMER_REAL, &timer, NULL); // running the timer

signal(SIGALRM, timer_callback);
```

כמובן שאם כן, אנו חוזרים לפונקציה למעלה שם מדפיסים הודעת שגיאה ויוצאים. אבל אם אין צורך, אז אנו מקבלים הודעת OK ויוצאים מהתוכנית בהצלחה:

```
if(recv(client_sock , buff , BUFSIZ , 0) < 0) // receiving OK message
{
    perror("recv() failed");
    close(client_sock);
    close(server_sock);
    exit(errno);
}

if(strcmp(buff, OK) != 0) // checking that OK received

{
    printf("error occurred!");
    close(client_sock);
    close(server_sock);
    exit(EXIT_FAILURE);
}

close(client_sock);
close(server_sock);
exit(EXIT_SUCCESS);</pre>
```

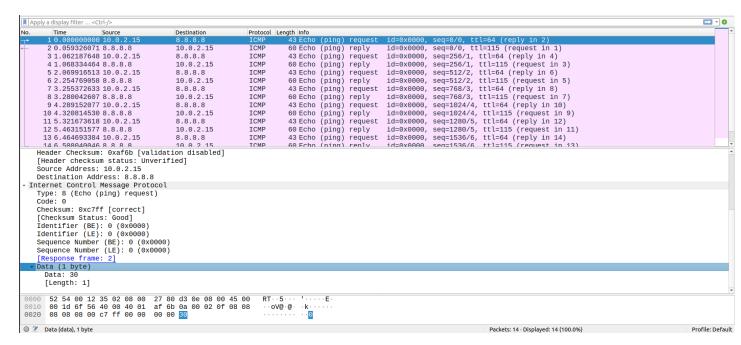
2 פרק ב' - תעבורה ופלט

2.1 פינג

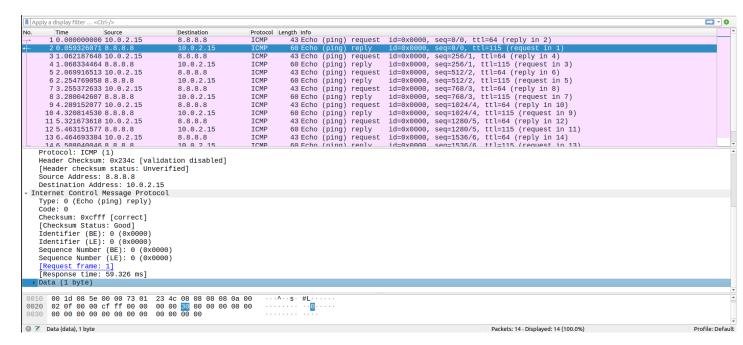
"ping.c" כאן אנו רואים את התעבורה והפלט של

ורת ICMP תעבורת **2.1.1**

כאן אנו יכולים לראות את התעבורה של הפינג, שליחה סטנדרטית של חבילות בקשה ולאחר מכן קבלה של חבילות תגובה. חבילת בקשה:



חבילת תגובה:



2.1.2 פלט התוכנית

פלט התוכנית המתקבל הוא פלט סטנדרטי כמו של התוכנית ping הסטנרטית של לינוקס. כאן הרצנו את התוכנית 7 פעמים:

```
vyoad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$ sudo ./PartA 8.8.8.8
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=0 ttl=115 time=59.44 ms
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=1 ttl=115 time=6.28 ms
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=2 ttl=115 time=185.08 ms
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=2 ttl=115 time=24.80 ms
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=4 ttl=115 time=31.81 ms
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=5 ttl=115 time=141.57 ms
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=6 ttl=115 time=43.48 ms
^c
voad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$
```

כמו כן, נשים לב כי בתעבורה שמלעיל יש בדיוק ($7\cdot 2$) אבילות, שכן בכל ריצה של התוכנית נשלחת חבילת בקשה ומתקבלת חבילת תגובה כך שהפלט הגיוני מאד.

2.2 פינג משוכלל ללא עצירה

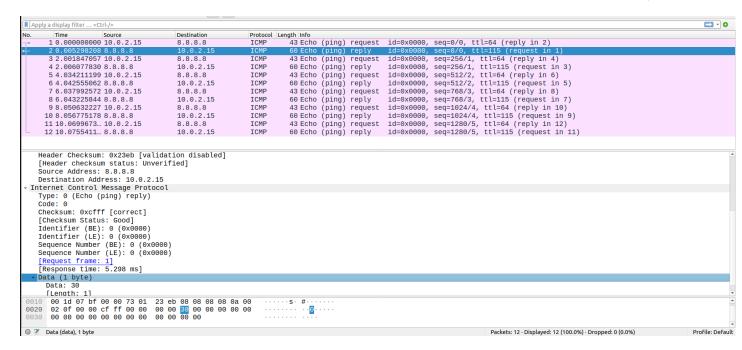
.watchdog ללא עצירת ה"better_ping.c" כאן אנו רואים את התעבורה והפלט של

וכMP תעבורת 2.2.1

כאן אנו רואים חבילה של בקשת פינג:

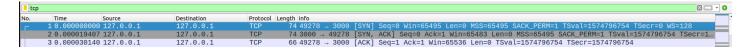
```
Apply a display filter ... <Ctrl-/
                                                                                                                                                                                                                                                                                 -
                                                                                           Protocol Length Info
                                                                                                         1 0.000000000 10.0.2.
2 0.005298208 8.8.8.8
                                                              10.0.2.15
                                                                                           ICMP
          3 2.001847057 10.0.2.15
                                                             8.8.8.8
10.0.2.15
                                                                                           ICMP
          4 2.006077830 8.8.8.8
5 4.034211199 10.0.2.15
                                                             8.8.8.8
                                                                                           ICMP
          6 4.042555062 8.8.8.8
7 6.037992572 10.0.2.15
8 6.043225844 8.8.8.8
                                                             10.0.2.15
                                                                                           TCMP
                                                             8.8.8.8
10.0.2.15
                                                                                           ICMP
ICMP
          9 8.050632227 10.0.2.15
                                                             8.8.8.8
                                                                                           ICMP
        10 8.056775178 8.8.8.8
11 10.0699673... 10.0.2.15
                                                             10.0.2.15
                                                                                           ICMP
ICMP
                                                             10.0.2.15
        12 10.0755411... 8.8.8.8
                                                                                           ICMP
  Frame 1: 43 bytes on wire (344 bits), 43 bytes captured (344 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_80:d3:0e (08:00:27:80:d3:0e), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 8.8.8.8
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
     Code: 0
Checksum: 0xc7ff [correct]
[Checksum Status: Good]
Identifier (BE): 0 (0x00000)
Identifier (LE): 0 (0x00000)
Sequence Number (BE): 0 (0x00000)
Sequence Number (LE): 0 (0x00000)
Sequence Number (LE): 0 (0x00000)
       [Response frame:
         Data: 30
         [Length: 1]
                                                                                               RT. 5... '...
 Data (data), 1 byte
                                                                                                                                                                                  Packets: 12 · Displayed: 12 (100.0%)
                                                                                                                                                                                                                                                                        Profile: Default
```

כאן אנו רואים חבילה של תגובת פינג:

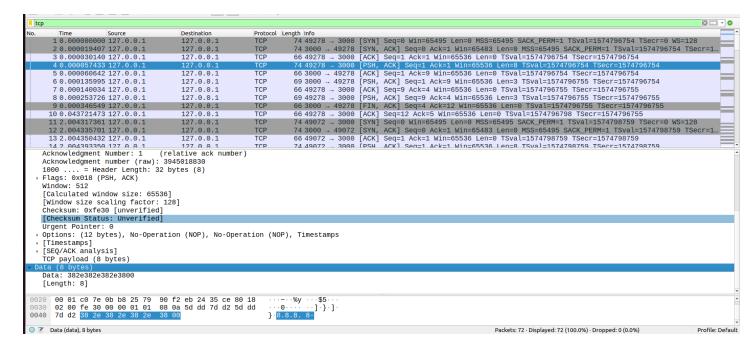


TCP תעבורת 2.2.2

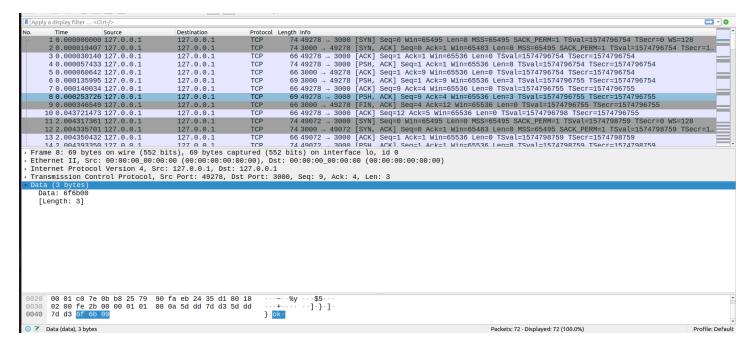
כעת בגלל שיש לנו watchdog ישנה גם תעבורת matchdog נתבונן בה. כאן אנו רואים פתיחת קשר:



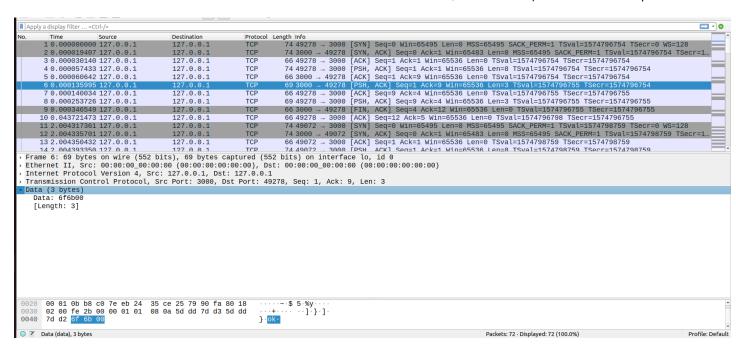
:IPכאו אנו רואים קבלה של כתובת ה



:OK כאן אנו רואים שיש שליחה של הודעת



:OK טיש קבלה של הודעת אנו רואים שיש קבלה



2.2.3 פלט התוכנית

פלט התוכנית שהתקבל:

```
    yoad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$ sudo ./PartB 8.8.8.8
    waiting for ping response...
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=0 ttl=115 time=0.81 ms
    waiting for ping response...
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=1 ttl=115 time=24.41 ms
    waiting for ping response...
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=2 ttl=115 time=2.44 ms
    waiting for ping response...
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=3 ttl=115 time=5.97 ms
    waiting for ping response...
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=4 ttl=115 time=0.87 ms
    waiting for ping response...
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=4 ttl=115 time=0.87 ms
    waiting for ping response...
29 bytes has been recv from 8.8.8.8 to 10.0.2.15: icmp_seq=5 ttl=115 time=0.62 ms
    ^C
    yoad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$
```

2.3 פינג משוכלל עם עצירה

.watchdogעם עצירת ה־ $"better_ping.c"$ עם עצירת התעבורה את אנו רואים את התעבורה והפלט של

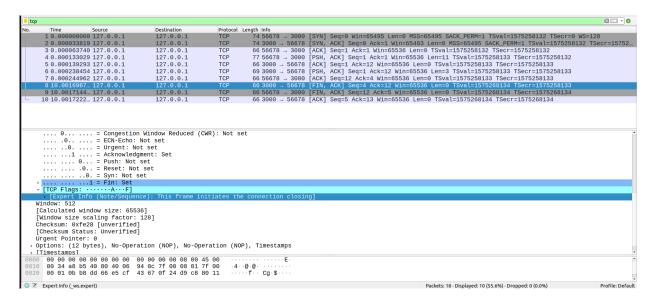
ורת 2.3.1 תעבורת 2.3.1

כאן אנו רואים חבילה בודדת של בקשת פינג, נשים לב שאין תגובה:

א כל התעבורה כיוון שלאחר החבילה הזו ה־watchdog עצר את התוכנית.

TCP תעבורת 2.3.2

כאן אנו רואים יצירת קשר וסגירת קשר אחת ורואים את כלל התהליכים שכבר ראינו קודם לכן:



2.3.3 פלט התוכנית

פלט התוכנית שהתקבל:

```
    yoad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$ sudo ./PartB 101.5.40.1
    waiting for ping response...
    server <101.5.40.1> cannot be reached.
    Killed
    yoad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$
```

3 ביבליוגרפיה

רשימת מקורות בהם השתמשנו בעת מימוש וכתיבת הפרוייקט:

- https://www.geeksforgeeks.org/fork-system-call/ .1
- https: //www.digitalocean.com/community/tutorials/execvp-function-c-plus-plus .2
 - https://www.techtarget.com/searchnetworking/definition/ICMP .3
 - 4. מצגות הקורס "תכנות מערכות 1" של ד"ר חוגי אסף.
 - 5. מצגות וחוברת הקורס "רשתות תקשורת".