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

PING (ICMP & TCP)

ליאור וינמן יועד תמר 213451818 213081763

2023 בינואר 2

תוכן עניינים

2								 					 							 				1	כח	ער	זמי	רו	או	תי	- '	X	ァ	פו	1
2																														צת				L.1	
3																														קוב			1	l.2	
4																														מוע			1	L.3	
6																														מוע			1	.4	
8																														מוע			1	l.5	
11																														תע		ב'	ァ	פו	2
11																														2				2.1	
11														 	 						Ι(C_{s}	M	P	ת	ור	נעב	П		2.	1.1				
12														 	 							1	ניר	וכ	זרנ	י ר	לט	<u>១</u>		2.	1.2				
13																													צש	נג כ	פיו		2	2.2	
13																											נעב				2.1				
14														 	 							T	C	P	ת	ור	נעב	П		2.	2.2				
16														 	 							1	ניר	וכ	זת	י ר	לט	೨		2.	2.3				
17															 							ī	ירו	עצ) t	עכ	ל	וכל	צש	נג נ	פיו		2	2.3	
17	•													 	 						Ι	$C_{\mathcal{A}}$	M	P	ת	ור	נעב	Π		2.	3.1				
18	•													 	 							T	C	P	ת	ור	נעב	Π		2.	3.2				
19														 	 							1	ניר	וכ	זרנ	י ר	לט	อ		2.	3.3				
20																														מה	97	าาว	ירל	רי	3

1 פרק א' - תיאור המערכת

1.1 הרצת הפרויקט

הקבצים הניתנים להרצה הם ping.c וping.c בלבד. (שאר הקבצים הם קבצי עזר). על מנת להריץ את הפרויקט בעלינו קודם כל להוריד את כל ארבעת הקבצים לתוך תיקיה משותפת על המחשב (יש להשתמש במערכת הפעלה $Linux\ Ubuntu\ LTS\ 22.04$).

 $"make \ all"$ הפקודה את הקבצים ונכתוב את התחינה על התקיה הנתונה על התרצה, נפתח תרמינל על התקיה הניתנות להרצה. (לאחר מכן מומלץ להריץ את הפקודה "clear" הדבר יגרום ליצירת קבצי ההרצה של התוכניות הניתנות להרצה. (לאחר מכן מומלץ להריץ את הפקודה "clear" כדי לנקות את הפלט הסטנדרטי מהטרמינל).

יש < ip> כעת, אם ברצוננו להריץ את ping.c ר נכתוב בטרמינל את הפקודה "ping.c את החוננו להריץ את לרשום כתובת אליה אנחנו מעוניינים לשלוח את הבקשות). אם אנו מעוניינים ואם ברצוננו להריץ את ping.c את אליה אנחנו מעוניינים לשלוח את הבקשות). אם אנו מעוניינים ואם ברצוננו להריץ את ping.c את הפקודה "ping.c את הפקודה". "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה "ping.c את הפקודה "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה "ping.c את הפקודה "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה "ping.c את הפקודה" וליינים לשלוח את הפקודה "ping.c את הפקודה" וליינים ליינים ליי

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 better ping watchdog
ping: ping.o
        $(CC) $(FLAGS) -o parta ping.o
better_ping: better_ping.o
        $(CC) $(FLAGS) -o partb better_ping.o
watchdog: watchdog.o
        $(CC) $(FLAGS) -o watchdog watchdog.o
ping.o: ping.c
        $(CC) $(FLAGS) -c ping.c
better_ping.o: better_ping.c
        $(CC) $(FLAGS) -c better 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

```
void calculate_checksum(struct icmphdr *icmp)

{
    unsigned long sum = 0;
    unsigned short *ptr = (unsigned short *)icmp, chksum = 0;

for (int i = 0; i < sizeof(struct icmphdr) / 2; i++)

{
    sum += *ptr++;
}

}

sum = (sum >> 16) + (sum & 0xfffff);

sum += (sum >> 16);
    chksum = ~sum;

icmp->checksum = chksum;

icmp->checksum = chksum;

}
```

:ping אליה לבצע וידאנו שהמשתמש אכן הריץ את התוכנית כנדרש, ז"א נתן לתוכנית כתובת IP אליה לבצע

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

for printf("usage: ./partb <ip>\n");
exit(EXIT_FAILURE);
}
```

כאן פתחנו שקע שדרכו תעבור תעבורת פינג (rawsocket)

כאן אנו מסדרים את ההגדרות של החבילות הנשלחות:

```
memset(&icmp, 0, sizeof(icmp)); // setting up the struct for ICMP communication
icmp.type = ICMP_ECHO;
icmp.code = 0;
icmp.un.echo.sequence = seq;
icmp.un.echo.id = getpid();
calculate_checksum(&icmp);
```

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, &icmp, sizeof(icmp), 0, (struct sockaddr*)&addr_ping, sizeof(addr_ping)) < 0) // sending ICMP-ECHO-REQUEST

f (sendto(sock, &icmp, sizeof(icmp), 0, (struct sockaddr*)&addr_ping, sizeof(addr_ping)) < 0) // sending ICMP-ECHO-REQUEST

f (sendto(sock, &icmp, sizeof(icmp), 0, (struct sockaddr*)&addr_ping, sizeof(addr_ping)) < 0) // sending ICMP-ECHO-REQUEST

close(sock);

addr_len = sizeof(addr_ping); // receiving ICMP-ECHO-REPLEY

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

if (len < 0)

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

close(sock);
    exit(errno);

d }

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

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

לבסוף אנו סוגרים את השקע שפתחנו ויוצאים בהצלחה מהתוכנית.

$better_ping.c$ - מימוש פינג משוכלל 1.4

:better_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(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>
```

(מריצים לעוד תהליך במקביל): watchdog את היvatchdog

```
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 הIP שהמשתמש הזין:

יש תקשורת: watchdog מה־ watchdog שיש תקשורת לקבל הודעת אנו אמורים לקבל הודעת

```
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 בחדקים 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>
```

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 הודעת היחשתמש את כתובת היIPים מהמשתמש את מקבלים מהמשתמש את כתובת ו

כאן אנו מפעילים טיימר ל־01 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);

close(server_sock);

exit(EXIT_SUCCESS);
</pre>
```

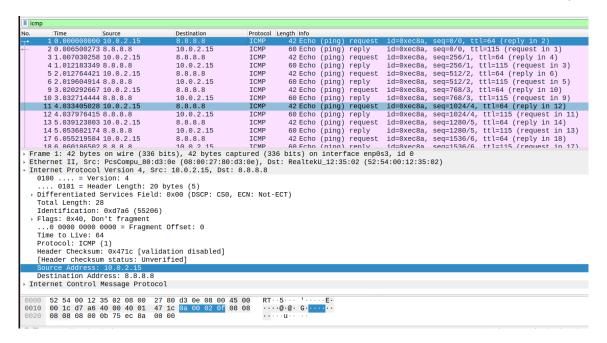
פרק ב' - תעבורה ופלט

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 21 פעמים:

```
voad@yoad-VirtualBox:~/Desktop/Ex4 cn/cnc assignment4-1$ sudo ./parta 8.8.8.8
28 bytes from 8.8.8.8: icmp_seq=0 ttl=115 time=6.61 ms
28 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=5.27 ms
28 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=7.03 ms
28 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=12.56 ms
28 bytes from 8.8.8.8: icmp_seq=4 ttl=115 time=12.56 ms
28 bytes from 8.8.8.8: icmp_seq=5 ttl=115 time=4.66 ms
28 bytes from 8.8.8.8: icmp_seq=5 ttl=115 time=5.07 ms
28 bytes from 8.8.8.8: icmp_seq=6 ttl=115 time=5.07 ms
28 bytes from 8.8.8.8: icmp_seq=7 ttl=115 time=5.04 ms
28 bytes from 8.8.8.8: icmp_seq=7 ttl=115 time=57.47 ms
28 bytes from 8.8.8.8: icmp_seq=9 ttl=115 time=157.47 ms
28 bytes from 8.8.8.8: icmp_seq=10 ttl=115 time=8.93 ms
28 bytes from 8.8.8.8: icmp_seq=11 ttl=115 time=8.93 ms
28 bytes from 8.8.8.8: icmp_seq=11 ttl=115 time=8.43 ms
28 bytes from 8.8.8.8: icmp_seq=13 ttl=115 time=6.12 ms
28 bytes from 8.8.8.8: icmp_seq=15 ttl=115 time=6.12 ms
28 bytes from 8.8.8.8: icmp_seq=15 ttl=115 time=6.12 ms
28 bytes from 8.8.8.8: icmp_seq=14 ttl=115 time=6.12 ms
28 bytes from 8.8.8.8: icmp_seq=13 ttl=115 time=6.14 ms
28 bytes from 8.8.8.8: icmp_seq=17 ttl=115 time=6.7.39 ms
28 bytes from 8.8.8.8: icmp_seq=17 ttl=115 time=4.54 ms
28 bytes from 8.8.8.8: icmp_seq=18 ttl=115 time=4.94 48 ms
28 bytes from 8.8.8.8: icmp_seq=18 ttl=115 time=5.10 ms
28 bytes from 8.8.8.8: icmp_seq=18 ttl=115 time=6.04 ms
70

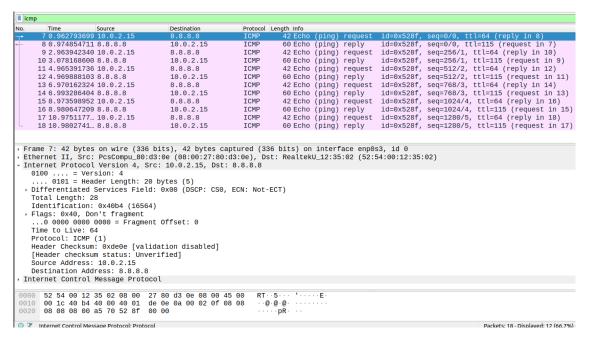
yoad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$
```

2.2 פינג משוכלל ללא עצירה

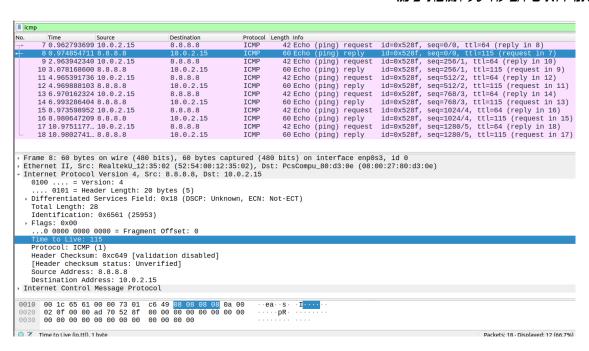
.watchdog ללא עצירת ה"better_ping.c" לא עצירת ההעבורה והפלט של

וכMP תעבורת 2.2.1

כאן אנו רואים חבילה של בקשת פינג:



כאן אנו רואים חבילה של תגובת פינג:

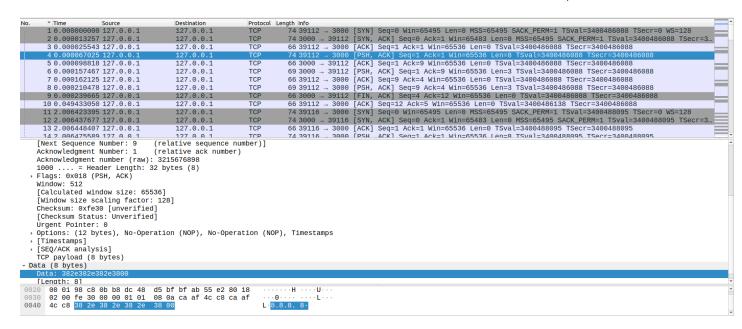


TCP תעבורת 2.2.2

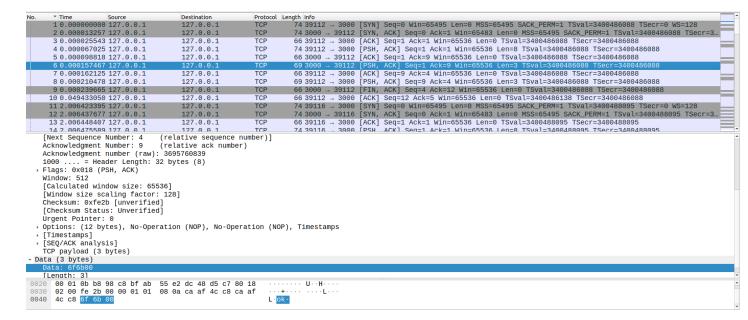
כעת בגלל שיש לנו watchdog ישנה גם תעבורת matchdog נתבונן בה. כאן אנו רואים פתיחת קשר:

No.	* Time	Source	Destination	Protocol I	Length Info
	10.000000000	127.0.0.1	127.0.0.1	TCP	74 39112 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSVal=3400486088 TSecr=0 WS=128
	2 0.000013257	127.0.0.1	127.0.0.1	TCP	74 3000 → 39112 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=3400486088 TSecr=3
	3 0.000025543	127.0.0.1	127.0.0.1	TCP	66 39112 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3400486088 TSecr=3400486088
	4 0.000067025	127.0.0.1	127.0.0.1	TCP	74 39112 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TSval=3400486088 TSecr=3400486088
	5 0.000098818	127.0.0.1	127.0.0.1	TCP	66 3000 → 39112 [ACK] Seq=1 Ack=9 Win=65536 Len=0 TSval=3400486088 TSecr=3400486088
	6 0.000157467	127.0.0.1	127.0.0.1	TCP	69 3000 → 39112 [PSH, ACK] Seq=1 Ack=9 Win=65536 Len=3 TSval=3400486088 TSecr=3400486088
	7 0.000162125	127.0.0.1	127.0.0.1	TCP	66 39112 → 3000 [ACK] Seq=9 Ack=4 Win=65536 Len=0 TSval=3400486088 TSecr=3400486088
	8 0.000210478	127.0.0.1	127.0.0.1	TCP	69 39112 → 3000 [PSH, ACK] Seq=9 Ack=4 Win=65536 Len=3 TSval=3400486088 TSecr=3400486088
	9 0.000239665	127.0.0.1	127.0.0.1	TCP	66 3000 → 39112 [FIN, ACK] Seq=4 Ack=12 Win=65536 Len=0 TSval=3400486088 TSecr=3400486088
	10 0.049433050	127.0.0.1	127.0.0.1	TCP	66 39112 → 3000 [ACK] Seq=12 Ack=5 Win=65536 Len=0 TSval=3400486138 TSecr=3400486088
	11 2.006423395	127.0.0.1	127.0.0.1	TCP	74 39116 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=3400488095 TSecr=0 WS=128
	12 2.006437677	127.0.0.1	127.0.0.1	TCP	74 3000 → 39116 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=3400488095 TSecr=3
	13 2.006448407	127.0.0.1	127.0.0.1	TCP	66 39116 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3400488095 TSecr=3400488095
	14 2 006475589	127 0 0 1	127 0 0 1	TCP	74 39116 . 3000 [PSH ACK] Seg=1 Ack=1 Win=65536 Len=8 TSval=3400488095 TSecr=3400488095

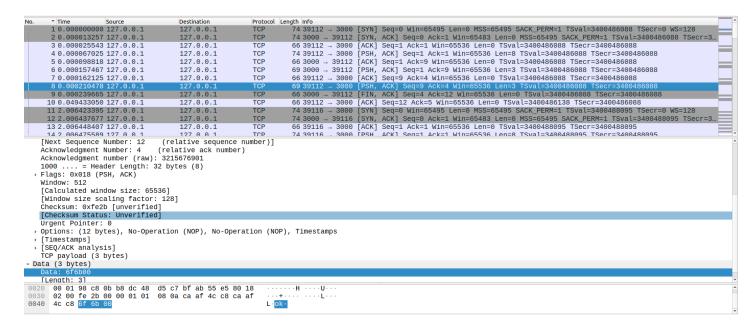
:IPכאו אנו רואים קבלה של כתובת ה



:OK כאן אנו רואים שיש שליחה של הודעת



:OK שיש קבלה של הודעת כאן



2.2.3 פלט התוכנית

פלט התוכנית שהתקבל:

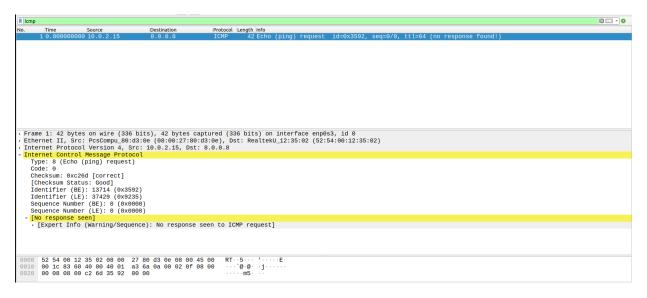
```
vyoad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$ sudo ./partb 8.8.8.8
waiting for ping response...
28 bytes from 8.8.8.8: icmp_seq=0 ttl=115 time=0.61 ms
waiting for ping response...
28 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=0.90 ms
waiting for ping response...
28 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=4.25 ms
waiting for ping response...
28 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=1.88 ms
waiting for ping response...
28 bytes from 8.8.8.8: icmp_seq=4 ttl=115 time=0.60 ms
waiting for ping response...
28 bytes from 8.8.8.8: icmp_seq=4 ttl=115 time=0.45 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

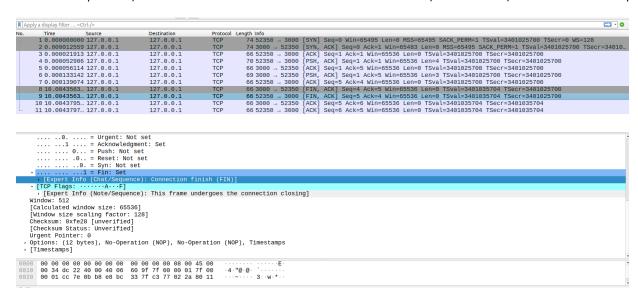
כאן אנו רואים חבילה בודדת של בקשת פינג, נשים לב שאין תגובה:



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 8.8
  waiting for ping response...
  server <8.8> 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".