

## PING (ICMP & TCP)

יועד תמר  
213451818

ליאור וינמן  
213081763

5 בינואר 2023

### תוכן עניינים

2	פרק א' - תיאור המערכת	1
2	1.1 הרצת הפרויקט	
3	1.2 הקובץ היוצר - <i>Makefile</i>	
4	1.3 מימוש פינג - <i>ping.c</i>	
7	1.4 מימוש פינג משוכלל - <i>new_ping.c</i>	
9	1.5 מימוש טיימר - <i>watchdog.c</i>	
12	פרק ב' - תעבורה ופלט	2
12	2.1 פינג	
12	2.1.1 תעבורת <i>ICMP</i>	
13	2.1.2 פלט התוכנית	
14	2.2 פינג משוכלל ללא עצירה	
14	2.2.1 תעבורת <i>ICMP</i>	
15	2.2.2 תעבורת <i>TCP</i>	
17	2.2.3 פלט התוכנית	
18	2.3 פינג משוכלל עם עצירה	
18	2.3.1 תעבורת <i>ICMP</i>	
19	2.3.2 תעבורת <i>TCP</i>	
20	2.3.3 פלט התוכנית	
21	3 ביבליוגרפיה	

# 1 פרק א' - תיאור המערכת

## 1.1 הרצת הפרויקט

הקבצים הניתנים להרצה הם `ping.c` ו-`new_ping.c` בלבד. (שאר הקבצים הם קבצי עזר).  
על מנת להריץ את הפרויקט - עלינו קודם כל להוריד את כל ארבעת הקבצים לתוך תיקיה משותפת על המחשב (יש להשתמש במערכת הפעלה `Linux Ubuntu LTS 22.04` בלבד).  
לאחר שהורדנו את הקבצים, נפתח תרמינל על התקיה הנתונה עם הקבצים ונכתוב את הפקודה "`make all`"  
הדבר יגרום ליצירת קבצי 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` - נכתוב בטרמינל את הפקודה "`sudo ./PartA < ip >`" (כאשר במקום  
`< ip >` יש לרשום כתובת אליה אנחנו מעוניינים לשלוח את הבקשות). אם אנו מעוניינים ואם ברצוננו להריץ את  
`new_ping.c` נכתוב בטרמינל את הפקודה "`sudo ./PartB < ip >`".  
לאחר שכתבנו את אחת מהפקודות שמלעיל, התוכנית פשוט תתחיל לרוץ (כלומר ישר תתחיל לשלוח ולקבל  
תעבורת פינג).

## 1.2 הקובץ היוצר - Makefile

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:

```
1 CC = gcc
2 FLAGS = -Wall -g
3
4 .PHONY: all clean
5
6 all: ping new_ping watchdog
7
8 ping: ping.o
9     $(CC) $(FLAGS) -o PartA ping.o
10
11 new_ping: new_ping.o
12     $(CC) $(FLAGS) -o PartB new_ping.o
13
14 watchdog: watchdog.o
15     $(CC) $(FLAGS) -o watchdog watchdog.o
16
17 ping.o: ping.c
18     $(CC) $(FLAGS) -c ping.c
19
20 new_ping.o: new_ping.c
21     $(CC) $(FLAGS) -c new_ping.c
22
23 watchdog.o: watchdog.c
24     $(CC) $(FLAGS) -c watchdog.c
25
26 clean:
27     rm -f *.o PartA PartB watchdog
```

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

### 1.3 מימוש פינג - *ping.c*

נעבור על הקובץ *ping.c*:  
בכל חבילה הנשלחת יש תא המיועד לביטים אשר מאמתים האם החבילה הגיעה בשלמותה ביטים אלו נקראים *checksum*. כאן כתבנו פונקציה לחישוב ולהשמה של ה-*checksum*:

```
26 unsigned short checksum(void *b, int len)
27 {
28     unsigned short *buf = b;
29     unsigned int sum=0;
30     unsigned short result = 0;
31
32     for ( sum = 0; len > 1; len -= 2 )
33         sum += *buf++;
34     if ( len == 1 )
35         sum += *(unsigned char*)buf;
36
37     sum = (sum >> 16) + (sum & 0xFFFF);
38     sum += (sum >> 16);
39     result = ~sum;
40     return result;
41 }
```

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

```
58 if (argc != 2) // checking that the user has specified an IP address
59 {
60     printf("usage: ./partb <ip>\n");
61     exit(EXIT_FAILURE);
62 }
```

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

```
69 for (size_t i = 0; i < MSG_LEN - 1; i++) //the message we sent
70 {
71     data[i] = '0';
72 }
73
74 data[MSG_LEN - 1] = '\0';
```

כאן פתחנו שקע שדרכו תעבור תעבורת פיינג (raw socket):

```
76     sock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP); // creating an RAW socket for ICMP communication
77     if (sock <= 0) // checking if socket created
78     {
79         perror("socket() failed");
80         close(sock);
81         exit(errno);
82     }
83
84     addr_ping.sin_family = AF_INET; // setting up the struct for ICMP communication
85     if(inet_aton(argv[1], &addr_ping.sin_addr) < 0)
86     {
87         perror("inet_aton() failed");
88         exit(errno);
89     }
```

כאן אנו מסדרים את ההגדרות של התבילות הנשלחות:

```
93     memset(&icmp, 0, sizeof(icmp)); // setting up the struct and packet for ICMP communication
94     memset(&packet, 0, sizeof(packet));
95     icmp.type = ICMP_ECHO;
96     icmp.un.echo.sequence = seq;
97     icmp.checksum = 0;
98
99     memcpy((packet), &icmp, ICMP_HDRLLEN);
100    memcpy(packet + ICMP_HDRLLEN, data, data_len);
101
102    icmp.checksum = checksum(&packet, sizeof(packet));
103    memcpy((packet), &icmp, ICMP_HDRLLEN);
```

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

```
105     gettimeofday(&start , NULL); // starting counting ping-time
106     if (sendto(sock, &packet, ICMP_HDRLen + data_len, 0, (struct sockaddr*)&addr_ping, sizeof(addr_ping)) < 0) // sending ICMP-ECHO-REQUEST
107     {
108         perror("sendto() failed");
109         close(sock);
110         exit(errno);
111     }
112
113     addr_len = sizeof(addr_ping); // receiving ICMP-ECHO-REPLY
114     bzero(buffer, IP_MAXPACKET);
115     len = recvfrom(sock, buffer, buffer_len, 0, (struct sockaddr *)&addr_ping, &addr_len);
116     if (len == -1)
117     {
118         perror("recvfrom() failed");
119         close(sock);
120         exit(errno);
121     }
122
123     gettimeofday(&end , NULL); // ending counting ping-time
124
125     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:

```
127     ip = (struct iphdr*)buffer;
128
129     printf("%d bytes has been recv from %s to " , len, inet_ntoa(*(struct in_addr*)&ip->saddr));
130     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);
131
132     seq++;
133
134     sleep(1);
```

לבסוף אנו סוגרים את השקע שפתחנו ויוצאים בהצלחה מהתוכנית.

## 1.4 מימוש פינג משוכלל - *new\_ping.c*

נעבור על הקובץ *new\_ping.c*:

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: כאן אנו פותחים שקע לתעבורת *TCP* (המתבצעת עם ה-*watchdog* בפורט 3000) ומגדירים את האובייקט שיחזיק את הכתובת של התקשורת:

```
105     sockfd = socket(AF_INET, SOCK_STREAM, 0); // creating socket for TCP communication
106     if(sock <= 0) // checking if socket created
107     {
108         perror("socket() failed");
109         close(sock);
110         close(sockfd);
111         exit(errno);
112     }
113
114     memset(&addr_server, '\0', sizeof(addr_server)); // setting up the struct for TCP communication
115     addr_server.sin_family = AF_INET;
116     addr_server.sin_port = htons(PORT);
117     addr_server.sin_addr.s_addr = inet_addr(IP);
```

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

```
140     pid = fork(); // starting the watchdog process
141     if (pid == 0)
142     {
143         execvp(args[0], args);
144     }
```

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

```
147     sleep(1); // delay until watchdog is set up
148
149     if(connect(sockfd, (struct sockaddr*)&addr_server, sizeof(addr_server)) < 0) // connecting to watchdog
150     {
151         perror("connect() failed");
152         close(sock);
153         close(sockfd);
154         exit(errno);
155     }
156
157     if(send(sockfd, argv[1], strlen(argv[1]) + 1, 0) <= 0) // sending IP to watchdog
158     {
159         perror("send() failed");
160         close(sock);
161         close(sockfd);
162         exit(errno);
163     }
```

כאן אנו אמורים לקבל הודעת OK מה-watchdog שזהו סימן שיש תקשורת:

```
165         if(recv(sockfd , buffer , BUFSIZ , 0) <= 0) // receiving an OK message from watchdog
166         {
167             perror("recv() failed");
168             close(sock);
169             close(sockfd);
170             exit(errno);
171         }
172         if(strcmp(buffer, OK) != 0) // checking that OK received
173         {
174             printf("error occurred!");
175             close(sock);
176             close(sockfd);
177             exit(EXIT_FAILURE);
178         }
```

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

```
193         if(send(sockfd, OK, strlen(OK) + 1 , 0) < 0) // sending OK message
194         {
195             perror("send() failed");
196             close(sock);
197             close(sockfd);
198             exit(errno);
199         }
200
201         wait(&status);
202         if(status == -1)
203         {
204             printf("proccess status failed\n");
205             close(sock);
206             close(sockfd);
207             exit(EXIT_FAILURE);
208         }
```

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



## 1.5 מימוש טיימר - *watchdog.c*

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.

נעבור על הקובץ *watchdog.c*:

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

```
24 void timer_callback()
25 {
26     if (!received_echo_reply)
27     {
28         printf("server <%s> cannot be reached.\n" , buffer); // printing unreachable message
29
30         kill(0 , SIGKILL); // killing all processes and exiting
31     }
32 }
```

כאן פתחנו את השקע שדרכו תעבור התעבורה (TCP):

```
46     server_sock = socket(AF_INET, SOCK_STREAM, 0); // creating the listener socket
47     if(server_sock <= 0) // checking if socket created
48     {
49         perror("socket() failed");
50         close(server_sock);
51         exit(errno);
52     }
```

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

```
54     if(setsockopt(server_sock, SOL_SOCKET, SO_REUSEADDR, &er, sizeof(er)) < 0) // checking if ip and port
55     {
56         perror("setsockopt() failed");
57         close(server_sock);
58         exit(errno);
59     }
60
61     server_addr.sin_family = AF_INET; // setting up the struct for TCP communication
62     server_addr.sin_port = htons(PORT);
63     server_addr.sin_addr.s_addr = inet_addr(IP);
```

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

```
65     if(bind(server_sock, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) // binding socket with settings
66     {
67         perror("bind() failed");
68         close(server_sock);
69         exit(errno);
70     }
71
72     if(listen(server_sock, CONNECTIONS) < 0) //listen to incoming connections
73     {
74         perror("listen() failed");
75         close(server_sock);
76         exit(errno);
77     }
78
79     addr_size = sizeof(client_addr);
80     client_sock = accept(server_sock, (struct sockaddr*)&client_addr, &addr_size); //accept a connection
81     if(client_sock <= 0) // checking if accepted
82     {
83         perror("accept() failed");
84         close(client_sock);
85         close(server_sock);
86         exit(errno);
87     }
```

כאן אנו מקבלים מהמשתמש את כתובת ה-IP ושולחים הודעת OK:

```
89     if(recv(client_sock , buffer , BUFSIZ , 0) < 0) // receiving IP
90     {
91         perror("recv() failed");
92         close(client_sock);
93         close(server_sock);
94         exit(errno);
95     }
96
97     if(send(client_sock, OK , strlen(OK) + 1 , 0) < 0) // sending OK message
98     {
99         perror("send() failed");
100         close(client_sock);
101         close(server_sock);
102         exit(errno);
103     }
```

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

```
105     timer.it_value.tv_sec = 10; // setting up timer for 10 seconds
106     timer.it_value.tv_usec = 0;
107     timer.it_interval.tv_sec = 0;
108     timer.it_interval.tv_usec = 0;
109
110     printf("waiting for ping response...\n");
111
112     setitimer(ITIMER_REAL, &timer, NULL); // running the timer
113
114     signal(SIGALRM, timer_callback);
```

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

```
116     if(recv(client_sock , buff , BUFSIZ , 0) < 0) // receiving OK message
117     {
118         perror("recv() failed");
119         close(client_sock);
120         close(server_sock);
121         exit(errno);
122     }
123     if(strcmp(buff, OK) != 0) // checking that OK received
124     {
125         printf("error occurred!");
126         close(client_sock);
127         close(server_sock);
128         exit(EXIT_FAILURE);
129     }
130
131
132     close(client_sock);
133     close(server_sock);
134     exit(EXIT_SUCCESS);
```

## 2 פרק ב' - תעבורה ופלט

### 2.1 פינג

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

#### 2.1.1 תעבורת ICMP

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

The screenshot shows a Wireshark packet capture of an ICMP Echo (ping) request. The packet list at the top shows a single packet (No. 1) of type ICMP Echo (ping) request, source 10.0.2.15, destination 8.8.8.8. The packet details pane shows the following information:

- Header Checksum: 0xaf6b [validation disabled]
- [Header checksum status: Unverified]
- Source Address: 10.0.2.15
- Destination Address: 8.8.8.8
- Internet Control Message Protocol
  - Type: 8 (Echo (ping) request)
  - Code: 0
  - Checksum: 0xc7ff [correct]
  - [Checksum Status: Good]
  - Identifier (BE): 0 (0x0000)
  - Identifier (LE): 0 (0x0000)
  - Sequence Number (BE): 0 (0x0000)
  - Sequence Number (LE): 0 (0x0000)
  - [Response frame: 2]
- Data (1 byte)
  - Data: 30
  - [Length: 1]

The packet bytes pane shows the raw data of the packet, including the ICMP header and the data byte 30.

חבילת תגובה:

The screenshot shows a Wireshark packet capture of an ICMP Echo (ping) reply. The packet list at the top shows a single packet (No. 1) of type ICMP Echo (ping) reply, source 8.8.8.8, destination 10.0.2.15. The packet details pane shows the following information:

- Protocol: ICMP (1)
- Header Checksum: 0x234c [validation disabled]
- [Header checksum status: Unverified]
- Source Address: 8.8.8.8
- Destination Address: 10.0.2.15
- Internet Control Message Protocol
  - Type: 0 (Echo (ping) reply)
  - Code: 0
  - Checksum: 0xcfff [correct]
  - [Checksum Status: Good]
  - Identifier (BE): 0 (0x0000)
  - Identifier (LE): 0 (0x0000)
  - Sequence Number (BE): 0 (0x0000)
  - Sequence Number (LE): 0 (0x0000)
  - [Request frame: 1]
  - [Response time: 59.326 ms]
- Data (1 byte)
  - Data: 50
  - [Length: 1]

The packet bytes pane shows the raw data of the packet, including the ICMP header and the data byte 50.

## 2.1.2 פלט התוכנית

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

```
◎ yoad@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=3 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
○ yoad@yoad-VirtualBox:~/Desktop/Ex4_cn/cnc_assignment4-1$ █
```

כמו כן, נשים לב כי בתעבורה שמלעיל יש בדיוק  $2 \cdot 7 = 14$  חבילות, שכן בכל ריצה של התוכנית נשלחת חבילת בקשה ומתקבלת חבילת תגובה כך שהפלט הגיוני מאוד.

## 2.2 פינג משוכלל ללא עצירה

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

### 2.2.1 תעבורת ICMP

כאן אנו רואים חבילה של בקשת פינג:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 2)
2	0.005298208	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=115 (request in 1)
3	2.001847057	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=256/1, ttl=64 (reply in 4)
4	2.006077830	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=256/1, ttl=115 (request in 3)
5	4.034211199	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=512/2, ttl=64 (reply in 6)
6	4.042555062	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=512/2, ttl=115 (request in 5)
7	6.037992572	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=768/3, ttl=64 (reply in 8)
8	6.043225844	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=768/3, ttl=115 (request in 7)
9	8.050632227	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=1024/4, ttl=64 (reply in 10)
10	8.056775178	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=1024/4, ttl=115 (request in 9)
11	10.0699673...	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=1280/5, ttl=64 (reply in 12)
12	10.0755411...	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=1280/5, ttl=115 (request in 11)

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  
Checksum: 0xc7ff [correct]  
[Checksum Status: Good]  
Identifier (BE): 0 (0x0000)  
Identifier (LE): 0 (0x0000)  
Sequence Number (BE): 0 (0x0000)  
Sequence Number (LE): 0 (0x0000)  
[Response frame: 2]  
Data (1 byte)  
Data: 30  
[Length: 1]

0000 52 54 00 12 35 02 08 00 27 80 d3 0e 00 00 45 00 RT 5 .....E  
0010 00 1d 49 45 40 00 40 01 d5 7c 0a 00 02 0f 08 08 ..IE@. .|.....  
0020 08 08 08 00 c7 ff 00 00 00 00 30 ..... ..0

Packets: 12 - Displayed: 12 (100.0%) Profile: Default

כאן אנו רואים חבילה של תגובת פינג:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 2)
2	0.005298208	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=115 (request in 1)
3	2.001847057	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=256/1, ttl=64 (reply in 4)
4	2.006077830	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=256/1, ttl=115 (request in 3)
5	4.034211199	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=512/2, ttl=64 (reply in 6)
6	4.042555062	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=512/2, ttl=115 (request in 5)
7	6.037992572	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=768/3, ttl=64 (reply in 8)
8	6.043225844	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=768/3, ttl=115 (request in 7)
9	8.050632227	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=1024/4, ttl=64 (reply in 10)
10	8.056775178	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=1024/4, ttl=115 (request in 9)
11	10.0699673...	10.0.2.15	8.8.8.8	ICMP	43	Echo (ping) request id=0x0000, seq=1280/5, ttl=64 (reply in 12)
12	10.0755411...	8.8.8.8	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=1280/5, ttl=115 (request in 11)

Header Checksum: 0x23eb [validation disabled]  
[Header checksum status: Unverified]  
Source Address: 8.8.8.8  
Destination Address: 10.0.2.15  
Internet Control Message Protocol  
Type: 0 (Echo (ping) reply)  
Code: 0  
Checksum: 0xcfff [correct]  
[Checksum Status: Good]  
Identifier (BE): 0 (0x0000)  
Identifier (LE): 0 (0x0000)  
Sequence Number (BE): 0 (0x0000)  
Sequence Number (LE): 0 (0x0000)  
[Request frame: 1]  
[Response time: 5.298 ms]  
Data (1 byte)  
Data: 30  
[Length: 1]

0010 00 1d 07 bf 00 00 73 01 23 eb 08 08 08 08 0a 00 .....S.#.....  
0020 02 0f 00 00 cf ff 00 00 00 00 00 00 00 00 00 00 ..... ..0.....  
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... ..

Packets: 12 - Displayed: 12 (100.0%) - Dropped: 0 (0.0%) Profile: Default



## 2.2.2 תעבורת TCP

קעת בגלל שיש לנו *watchdog* ישנה גם תעבורת TCP. נתבונן בה.  
כאן אנו רואים פתיחת קשר:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	49278 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1574796754 TSecr=0 WS=128
2	0.000019407	127.0.0.1	127.0.0.1	TCP	74	3000 → 49278 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1574796754 TSecr=1
3	0.000030140	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1574796754 TSecr=1574796754

כאן אנו רואים קבלה של כתובת ה-IP:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	49278 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1574796754 TSecr=0 WS=128
2	0.000019407	127.0.0.1	127.0.0.1	TCP	74	3000 → 49278 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1574796754 TSecr=1
3	0.000030140	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1574796754 TSecr=1574796754
4	0.000057433	127.0.0.1	127.0.0.1	TCP	74	49278 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TSval=1574796754 TSecr=1574796754
5	0.000060642	127.0.0.1	127.0.0.1	TCP	66	3000 → 49278 [ACK] Seq=1 Ack=9 Win=65536 Len=0 TSval=1574796754 TSecr=1574796754
6	0.000135995	127.0.0.1	127.0.0.1	TCP	69	3000 → 49278 [PSH, ACK] Seq=1 Ack=9 Win=65536 Len=3 TSval=1574796755 TSecr=1574796754
7	0.000140034	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=9 Ack=4 Win=65536 Len=0 TSval=1574796755 TSecr=1574796755
8	0.000253726	127.0.0.1	127.0.0.1	TCP	69	49278 → 3000 [PSH, ACK] Seq=9 Ack=4 Win=65536 Len=3 TSval=1574796755 TSecr=1574796755
9	0.000346549	127.0.0.1	127.0.0.1	TCP	66	3000 → 49278 [FIN, ACK] Seq=4 Ack=12 Win=65536 Len=0 TSval=1574796755 TSecr=1574796755
10	0.043721473	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=12 Ack=5 Win=65536 Len=0 TSval=1574796798 TSecr=1574796755
11	2.004317361	127.0.0.1	127.0.0.1	TCP	74	49072 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1574798759 TSecr=0 WS=128
12	2.004335701	127.0.0.1	127.0.0.1	TCP	74	3000 → 49072 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1574798759 TSecr=1
13	2.004350432	127.0.0.1	127.0.0.1	TCP	66	49072 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1574798759 TSecr=1574798759
14	2.004393350	127.0.0.1	127.0.0.1	TCP	74	49072 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TSval=1574798759 TSecr=1574798759

  

Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 3945018830
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xfe30 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[Timestamps]
[SEQ/ACK analysis]
TCP payload (8 bytes)
Data (8 bytes)
Data: 382e382e382e3800
[Length: 8]

  

0020	00 01 c0 7e 0b b8 25 79 90 f2 eb 24 35 ce 80 18	.....%y...\$5...
0030	02 00 fe 30 00 00 01 01 08 0a 5d dd 7d d2 5d dd	...0.....}]..
0040	7d d2 88 2e 38 2e 38 2e 38 00	}8.8.8.8

Packets: 72 - Displayed: 72 (100.0%) - Dropped: 0 (0.0%) Profile: Default

כאן אנו רואים שיש שליחה של הודעת OK:

The screenshot shows a Wireshark packet capture on interface lo. The packet list displays several TCP segments. The selected packet (No. 13) is a TCP segment from 127.0.0.1 to 127.0.0.1, Seq=9, Ack=4, Len=3. The packet details pane shows the data field containing three bytes: 6f 6b 09. The packet bytes pane shows the raw data in hexadecimal and ASCII, with the ASCII representation being 'OK'.

No.	Time	Source	Destination	Protocol	Length	Info
10	0.000000000	127.0.0.1	127.0.0.1	TCP	74	49278 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1574796754 TSecr=0 WS=128
20	0.000019407	127.0.0.1	127.0.0.1	TCP	74	3000 → 49278 [ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1574796754 TSecr=1...
30	0.000030140	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1574796754 TSecr=1574796754
40	0.000057433	127.0.0.1	127.0.0.1	TCP	74	49278 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TSval=1574796754 TSecr=1574796754
50	0.000060642	127.0.0.1	127.0.0.1	TCP	66	3000 → 49278 [ACK] Seq=1 Ack=9 Win=65536 Len=0 TSval=1574796754 TSecr=1574796754
60	0.000135995	127.0.0.1	127.0.0.1	TCP	69	3000 → 49278 [PSH, ACK] Seq=1 Ack=9 Win=65536 Len=3 TSval=1574796755 TSecr=1574796754
70	0.000140034	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=9 Ack=4 Win=65536 Len=0 TSval=1574796755 TSecr=1574796755
80	0.000253726	127.0.0.1	127.0.0.1	TCP	69	49278 → 3000 [PSH, ACK] Seq=9 Ack=4 Win=65536 Len=3 TSval=1574796755 TSecr=1574796755
90	0.000346549	127.0.0.1	127.0.0.1	TCP	66	3000 → 49278 [FIN, ACK] Seq=4 Ack=12 Win=65536 Len=0 TSval=1574796755 TSecr=1574796755
100	0.043721473	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=12 Ack=5 Win=65536 Len=0 TSval=1574796798 TSecr=1574796755
112	2.004317361	127.0.0.1	127.0.0.1	TCP	74	49672 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1574798759 TSecr=0 WS=128
122	2.004335701	127.0.0.1	127.0.0.1	TCP	74	3000 → 49672 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1574798759 TSecr=1...
132	2.004350432	127.0.0.1	127.0.0.1	TCP	66	49672 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1574798759 TSecr=1574798759
142	2.004393350	127.0.0.1	127.0.0.1	TCP	74	49672 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TSval=1574798759 TSecr=1574798759

Frame 8: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface lo, id 0  
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
Transmission Control Protocol, Src Port: 49278, Dst Port: 3000, Seq: 9, Ack: 4, Len: 3  
Data (3 bytes)  
Data: 6f6b09  
[Length: 3]

0020 00 01 c0 7e 0b b8 25 79 00 fa eb 24 35 d1 80 18 .....%y...\$5...  
0030 02 00 fe 2b 00 00 01 01 08 0a 5d dd 7d d3 5d dd .....+.....}.].  
0040 7d d3 6f 6b 09 .....OK..

Packets: 72 · Displayed: 72 (100.0%) Profile: Default

כאן אנו רואים שיש קבלה של הודעת OK:

The screenshot shows a Wireshark packet capture on interface lo. The packet list displays several TCP segments. The selected packet (No. 13) is a TCP segment from 127.0.0.1 to 127.0.0.1, Seq=9, Ack=9, Len=3. The packet details pane shows the data field containing three bytes: 6f 6b 09. The packet bytes pane shows the raw data in hexadecimal and ASCII, with the ASCII representation being 'OK'.

No.	Time	Source	Destination	Protocol	Length	Info
10	0.000000000	127.0.0.1	127.0.0.1	TCP	74	49278 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1574796754 TSecr=0 WS=128
20	0.000019407	127.0.0.1	127.0.0.1	TCP	74	3000 → 49278 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1574796754 TSecr=1...
30	0.000030140	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1574796754 TSecr=1574796754
40	0.000057433	127.0.0.1	127.0.0.1	TCP	74	49278 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TSval=1574796754 TSecr=1574796754
50	0.000060642	127.0.0.1	127.0.0.1	TCP	66	3000 → 49278 [ACK] Seq=1 Ack=9 Win=65536 Len=0 TSval=1574796754 TSecr=1574796754
60	0.000135995	127.0.0.1	127.0.0.1	TCP	69	3000 → 49278 [PSH, ACK] Seq=1 Ack=9 Win=65536 Len=3 TSval=1574796755 TSecr=1574796754
70	0.000140034	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=9 Ack=4 Win=65536 Len=0 TSval=1574796755 TSecr=1574796755
80	0.000253726	127.0.0.1	127.0.0.1	TCP	69	49278 → 3000 [PSH, ACK] Seq=9 Ack=4 Win=65536 Len=3 TSval=1574796755 TSecr=1574796755
90	0.000346549	127.0.0.1	127.0.0.1	TCP	66	3000 → 49278 [FIN, ACK] Seq=4 Ack=12 Win=65536 Len=0 TSval=1574796755 TSecr=1574796755
100	0.043721473	127.0.0.1	127.0.0.1	TCP	66	49278 → 3000 [ACK] Seq=12 Ack=5 Win=65536 Len=0 TSval=1574796798 TSecr=1574796755
112	2.004317361	127.0.0.1	127.0.0.1	TCP	74	49672 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1574798759 TSecr=0 WS=128
122	2.004335701	127.0.0.1	127.0.0.1	TCP	74	3000 → 49672 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1574798759 TSecr=1...
132	2.004350432	127.0.0.1	127.0.0.1	TCP	66	49672 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1574798759 TSecr=1574798759
142	2.004393350	127.0.0.1	127.0.0.1	TCP	74	49672 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TSval=1574798759 TSecr=1574798759

Frame 6: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface lo, id 0  
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
Transmission Control Protocol, Src Port: 3000, Dst Port: 49278, Seq: 1, Ack: 9, Len: 3  
Data (3 bytes)  
Data: 6f6b09  
[Length: 3]

0020 00 01 0b b8 c0 7e eb 24 35 ce 25 79 90 fa 80 18 .....\$ 5%y...  
0030 02 00 fe 2b 00 00 01 01 08 0a 5d dd 7d d3 5d dd .....+.....}.].  
0040 7d d2 6f 6b 09 .....OK..

Packets: 72 · Displayed: 72 (100.0%) Profile: Default



כאן אנו רואים סגירת קשר:

9	0.000346549	127.0.0.1	127.0.0.1	TCP	66 3000 → 49278 [FIN, ACK] Seq=4 Ack=12 Win=65536 Len=0 TSval=1574796755 TSecr=1574796755
10	0.043721473	127.0.0.1	127.0.0.1	TCP	66 49278 → 3000 [ACK] Seq=12 Ack=5 Win=65536 Len=0 TSval=1574796798 TSecr=1574796755

### 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 rcv 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 rcv 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 rcv 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 rcv 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 rcv 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 rcv 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

כאן אנו רואים את התעבורה והפלט של "better\_ping.c" עם עצירת ה-watchdog.

### 2.3.1 תעבורת ICMP

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.0.0.8	ICMP	42	Echo (ping) request id=0x3592, seq=0/0, ttl=64 (no response found!)
Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0						
Ethernet II, Src: PcsCompu_08: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: 8.0.0.8						
Internet Control Message Protocol						
Type: 8 (Echo (ping) request)						
Code: 0						
Checksum: 0xc26d [correct]						
[Checksum Status: Good]						
Identifier (BE): 13714 (0x3592)						
Identifier (LE): 37429 (0x9235)						
Sequence Number (BE): 0 (0x0000)						
Sequence Number (LE): 0 (0x0000)						
[No response seen]						
[Expert Info (warning/Sequence): No response seen to ICMP request]						
0000	52 54 00 12 35 02 08 00	27 00 d3 0e 08 00 45 00	RT: 5... ..E-			
0010	00 1c 83 60 40 00 01 a3	6a 00 02 0f 08 00	...@...j.....			
0020	00 08 08 00 c2 6d 35 92	00 00	.....m5..			

זו כל התעבורה כיוון שלאחר החבילה הזו ה-watchdog עצר את התוכנית.

## 2.3.2 תעבורת TCP

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	56678 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1575258132 TSecr=0 WS=128
2	0.000033819	127.0.0.1	127.0.0.1	TCP	74	3000 → 56678 [SYN, ACK] Seq=0 Ack=1 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1575258132 TSecr=15752...
3	0.000063740	127.0.0.1	127.0.0.1	TCP	66	56678 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1575258132 TSecr=1575258132
4	0.000133829	127.0.0.1	127.0.0.1	TCP	77	56678 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=11 TSval=1575258133 TSecr=1575258132
5	0.000139293	127.0.0.1	127.0.0.1	TCP	66	3000 → 56678 [ACK] Seq=1 Ack=12 Win=65536 Len=0 TSval=1575258133 TSecr=1575258133
6	0.000238454	127.0.0.1	127.0.0.1	TCP	69	3000 → 56678 [PSH, ACK] Seq=1 Ack=12 Win=65536 Len=3 TSval=1575258133 TSecr=1575258133
7	0.000244962	127.0.0.1	127.0.0.1	TCP	66	56678 → 3000 [ACK] Seq=12 Ack=4 Win=65536 Len=0 TSval=1575258133 TSecr=1575258133
8	10.0016967	127.0.0.1	127.0.0.1	TCP	66	3000 → 56678 [FIN, ACK] Seq=4 Ack=12 Win=65536 Len=0 TSval=1575268134 TSecr=1575258133
9	10.0017144	127.0.0.1	127.0.0.1	TCP	66	56678 → 3000 [FIN, ACK] Seq=12 Ack=5 Win=65536 Len=0 TSval=1575268134 TSecr=1575268134
10	10.0017222	127.0.0.1	127.0.0.1	TCP	66	3000 → 56678 [ACK] Seq=5 Ack=13 Win=65536 Len=0 TSval=1575268134 TSecr=1575268134

  

.... 0...	.... = Congestion Window Reduced (CWR): Not set
.... .0...	.... = ECN-Echo: Not set
.... ..0...	.... = Urgent: Not set
.... ...1...	.... = Acknowledgment: Set
.... ....0...	.... = Push: Not set
.... .....0...	.... = Reset: Not set
.... ....0...	.... = Syn: Not set
.... ....1...	.... = Fin: Set
[TCP Flags: .....A...F]	
[Expert Info (Note/Sequence): This frame initiates the connection closing]	
Window: 512	
[Calculated window size: 65536]	
[Window size scaling factor: 128]	
Checksum: 0xfe28 [unverified]	
[Checksum Status: Unverified]	
Urgent Pointer: 0	
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps	
[Timestamps]	

  

0000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....E.
0010	00 34 a8 b5 40 00 00 06 94 0c 7f 00 00 01 7f 00	4: 0 0 .....
0020	00 01 0b b8 dd 66 e5 cf 43 67 0f 24 d9 c8 80 11	.....f.. Cg \$....

Expert Info (.ws.expert)      Packets: 18 - Displayed: 10 (55.6%) - Dropped: 0 (0.0%)      Profile: Default

### 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 וכתובת הפרוייקט:

1. <https://www.geeksforgeeks.org/fork-system-call/>

2. <https://www.digitalocean.com/community/tutorials/execvp-function-c-plus-plus>

3. <https://www.techtarget.com/searchnetworking/definition/ICMP>

4. מצגות הקורס "תכנות מערכות 1" של ד"ר חוגי אסף.

5. מצגות וחוברת הקורס "רשתות תקשורת".