

# מטלה 4 ברשתות תקשורת-

מגישים: יסמין כהן- 212733836  
שלומי זכריה- 315141242

הקדמה-

במטלה הזאת למדנו איך להשתמש בsocket Raw וגם בפרוטוקול ICMP. בהתחלה בנינו תוכנית ping.c בעזרת הקוד שהביאו לנו, מטרת ping.c היא לתקשר עם מכשיר מסוים שנתון לנו רק ה IP שלו, והתקשורת איתו נעשית בשליחת וקבלת ping שזה בעצם חבילות ICMP, ולכן בחיבור נשתמש בsocket raw. בנינו בנוסף תוכנית better\_ping.c שיש בה תוספת ל ping.c, התוספת היא watchdog שתפקידו להקציב זמן לקבלת ה ping, ולכן אם נכשל החיבור מאיזה שהיא סיבה או שלקח יותר מהזמן המותר לקבלת ה ping החיבור נסגר בצורה יפה. והתוכנית לא מחכה לנצח עד שיתקבל החבילה. גם בתוכנית הזאת בכדי להתחבר ל IP יעד (היעד שנרצה לקבל ממנו את ping) נשתמש ב Raw Sockets וכדי להתחבר ל watchdog (שמקציב לנו את הזמן) נשתמש ב TCP socket.

## :ping.c

דבר ראשון אנחנו פותחים Raw Socket מכון שאנחנו עובדים עם פרוטוקול ICMP.

עבור כל פעולה, אם אירעה שגיאה, התוכנית תודיע מיד למשתמש ותיסגר. בנינו חבילת ICMP:

חבילת ECHO REQUEST עם כל הפרמטרים והמידע שצריך, לדוגמא המספר הסידורי, ההודעה שנרצה לשלוח, והגודל של החבילה.

נשלח את החבילה בעזרת פונקציית sendto() אל כתובת IP של יעד.

אם מתקבלת תגובה, אז התוכנית תדפיס את חבילת ה-ICMP ECHO REPLAY שהתקבלה, היא תדפיס את: גודל החבילה, מספר סידורי שלה, ואת הזמן שלקח לה להתקבל.

אם לא התקבלה תגובה היא תחכה עד שיגיע, ואם לא יגיע היא תחכה לנצח. אם התקבלה תגובה, התוכנית "תישן" שנייה אחת ואז שוב תחזור על אותה פעולה ותשלח עוד חבילה, עד שהמשתמש יחליט להפסיק (בעזרת control+c).

חשוב להבין שאם התוכנית לא מקבלת ping בחזרה היא תחכה לנצח.

## Better Ping.c

תוכנית Better Ping דומה ל-ping.c רק שיש בה הוספת שדרוג- השדרוג הוא watchdog שמודד את הזמן שלוקח לתגובה להגיע ומחליט מתי לסגור את התוכנית לפי זמן מוקצב מסוים, אצלנו זה 10 שניות, כדי שלא נחכה לנצח.

זה קורה באמצעות התוכנית "watchdog".

ב-better\_ping דבר ראשון פותחים RAW socket שאנחנו רוצים שהוא יהיה במצב non-blocking, שזה אומר שבפונקציות שבדרך כלל מחכות זמן בלתי מוגבל עד שתהיה תגובה, לדוג' recv, הם כבר יצטרכו לחכות את כל הזמן, כך שפשוט אם לא התקבל התגובה אליה ציפו, הם יעברו לפקודה הבאה בקוד. דבר שני נפתח גם TCP socket, כדי לתקשר עם watchdog.

בכדי ליצור ריצה בו זמנית של שתי התוכניות ה-better\_ping וה-watchdog נרצה להשתמש ב-thread, ולכן נשתמש בפונקציה fork כדי ליצור תהליך נוסף שהוא watchdog. ולכן כאשר אנחנו מריצים את better\_ping זה מפעיל את תוכנית ה-Watchdog בזמן ריצה.

כאשר נפתח תוכנית watchdog,

דבר ראשון נפתח אצלו חיבור TCP socket והוא מאזין מעל פורט 3000.

(אנחנו כאילו מתייחסים בתוכנית ל-watchdog כאל server ואל better\_ping כאל client).

ב-watchdog, אנו נמתין לחיבור TCP מהתוכנית ה-Better Ping.

לאחר שנוצר חיבור, watchdog פשוט ממתין לאות אישור מה-Better Ping. שאכן הגיע אליו תגובת הפינג.

אם watchdog לא קיבל מיד שום אות, הוא לא יחכה לקבל, הוא מוגדר כ-MSG\_DONTWAIT שזו שיטה של פרוטוקול tcp לעשות את הפעולה שהסברנו של non-blocking.

לכן אם הוא לא קיבל אות הוא מיד יגדיל באחד את האינדקס time שמכיל את השניות שעברו וילך לישון שנייה אחת. לאחר מכן יצפה לקבל אות מה-better\_Ping שוב.

ה-time סופר כמה שניות בעצם **לא התקבל אות** מה-better\_ping שמייצג שהתקבלה תגובה מהפינג.

בעצם time ממש סופר את השניות לקבלת החבילת חזרה של הפינג.

(ומכון שכל פעם אחרי שמעלים את האינדקס ישנים שנייה זה יוצא בסוף בערך 10 שניות)

אם הוא watchdog יקבל לפתע אות אישור '1', שזה אומר שה-better\_ping קיבל פינג בחזרה.

הוא מיד יאפס את הטיימר שלו, וילך לישון שנייה אחת.  
ויחכה שוב לקבל אות מחדש. על קבלת פינג של השליחה הבאה.  
ואם קורה מצב שהאינדקס timer (שמכיל את השניות שעברו) עבר את המספר 10, הוא ישלח אות '0'  
better\_ping שזה אות שמייצג **נכשל** בתוכנית Better Ping וזה יצא מהלולאת קבלה של  
better\_ping ויגרום לסגירת התוכנית better\_ping.

נרצה להסביר שוב יותר בפירוט איך כל זה נראה מהצד של better\_ping,  
לאחר ש better\_ping הפעיל את התוכנית watchdog, התוכנית תנסה להתחבר לתוכנית Watchdog  
דרך שקע ה-TCP.

לאחר יצירת חיבור (אם הצליח),  
התוכנית Better Ping תכין חבילת ECHO REQUEST עם כל הפרמטרים שצריך.  
ותשלח את החבילת icmp אל הכתובת IP של היעד.  
לאחר מכן, אנחנו נרצה לקבל חבילת ICMP ECHO REPLAY מה IP יעד שלנו.  
מכיוון שהsocket RAW נמצא במצב של nonBlocking, אז אם קורה מצב שלא נקבל תגובה מיד, אז  
התוכנית פשוט תמשיך לפקודה הבאה ולא תחכה לקבלת החבילה.  
ואז נבדוק את המצבים השונים:

- אם לא קיבלנו חבילה במייד, ננסה לקבל שוב את החבילה.  
חשוב להבין, שאם עבר 10 שניות watchdog ישלח לנו אות ויעצור אותנו, אבל כל עוד לא עבר  
10 שניות, הוא לא שולח לנו אות לעצירה ואנחנו ממשיכים לנסות לקבל.  
- אם התקבלה החבילה מיד נצא מהלולאת קבלה ונדפיס את הנתונים.  
לאחר מכן נשלח לwatchdog אות שמראה שאכן התקבלה החבילה ואז יתאפס אצלו הtimer.  
חשוב להבין שסוף סוף הוא יקבל לפונקציית recv שלו את האות אליו הוא חיכה, כל שאר  
הפעמים הוא פשוט לא יקבל כלום.  
לאחר שליחת האות שולחים פינג נוסף... באותה דרך.  
- אם עבר יותר מ10 שניות- שזה קורה בגלל שאנחנו לא עונים (לא שולחים שום אות שאכן  
התקבל) בכל פעם שאנחנו לא עונים watchdog במקביל יספור את מספר הפעמים שלא ענינו.  
וכאשר זה יגיע ל10 הוא ישלח אות סיום-  
watchdog ידאג לשלוח אות '0' שזה אומר לbetter\_ping לצאת מהתוכנית ולסגור סוקטים.

התוכנית תמשיך לשלוח חבילות עד שהמשתמש יחליט לסגור את Watchdog או עד שהwatchdog יבחין  
שלקח יותר מידי זמן לקבל את החבילה ויסגור מעצמו את התוכנית.

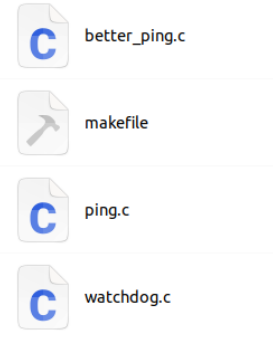
חשוב להבהיר שהמצב non blocking עוזר לכך שבפונקציית recv אנחנו לא נחכה לנצח אלא נצא אחרי  
זמן מסוים.  
וישנה שיטה דומה בtcp שנקראת MSG\_DONTWAIT.

## איך מריצים את התוכנית-

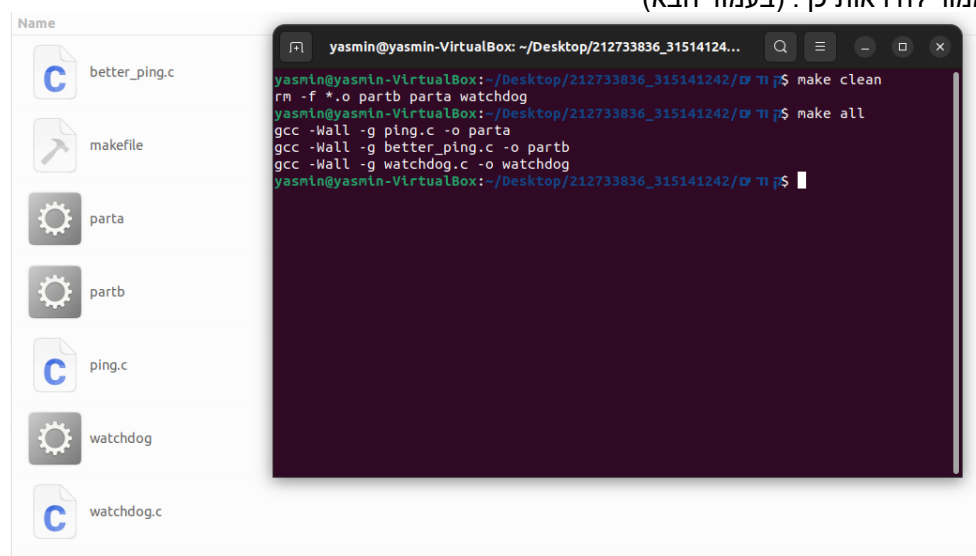
בזיפ שהעלנו יש שתי תיקיות, אחד של הקודים והשני של ההקלטות wireshark,  
צריך לחלץ את התקיה הכללית, ובתקיה של הקוד ישנם 3 קבצים להרצה שלבנתיים הם בסיומת c. ובנוסף  
יש קובץ readme שזה הקובץ הזה.

1. ping.c - תוכנית הפינג הפשוטה - חלק א'.
2. better\_ping.c - תוכנית הפינג שמיישמת את ה watchdog - חלק ב'.
3. watchdog.c - תוכנית שמריצים אותה בתוך butter\_ping.c לא ידנית, רק צריך לקמפל אותה.
4. makefile - משתמשים בו כדי שיהיה נוח לקמפל את התוכנית כולה בבת אחת כדי שיהיה מוכן  
להרצה.

## הנה הקבצים-



צריך דבר ראשון להשתמש ב- `makefile`.  
פונתחים טרמינל בתקיה שנמצאים בה כל הקבצי C, בתקיית הקודים,  
וכותבים בה את הפקודה- `make all`.  
לאחר מכן, יוצרו קבצים נוספים, מה שחשובים הם:  
parta - קובץ ההפעלה של C Ping חלק א' במטלה.  
partb - קובץ ההפעלה Better Ping C חלק ב' של המטלה.  
Watchdog - קובץ ההפעלה, שאותו רק partb מפעילה מתוכה, ולא אנחנו.  
אמור להיראות כך: (בעמוד הבא)



לאחר שיצרנו את הקבצי הרצה.  
בשביל להפעיל את חלק א- שזה ping הפשוט, צריך להקליד בטרמינל: `sudo ./parta "IP"`  
כאשר "IP" היא כתובת ה-IPv4 היעד שאליו נרצה לשלוח את הפינג.  
חשוב להקליד `sudo` לפני הפקודה, ומכאן שזה פקודה כזו צריך להזין את סיסמת המשתמש.  
הIP הדיפולטיבי של התוכנית הוא 8.8.8.8. אם לא נקליד את הIP זה ישלח פינגים לIP של גוגל.  
תמונה של הרצה פה למטה...

```

12733836_315141242/000000$ sudo ./parta 8.8.8.8
[sudo] password for yasmin:
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 1

RTT: 4.762000 milliseconds (4762 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 2

RTT: 4.685000 milliseconds (4685 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 3

RTT: 4.932000 milliseconds (4932 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8

```

כדי להפסיק את הריצה האינסופית הזאת של חלק א' נצטרך ללחוץ על `control+c`.

```

yasmin@yasmin-VirtualBox: ~/Desktop/final
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 20
RTT: 104.057999 milliseconds (104058 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 21
RTT: 100.427002 milliseconds (100427 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 22
RTT: 102.494003 milliseconds (102494 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 23
RTT: 102.439003 milliseconds (102439 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 24
RTT: 101.903000 milliseconds (101903 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 25
RTT: 101.832001 milliseconds (101832 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 26
RTT: 21.775999 milliseconds (21776 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
Successfully received one packet with 47 bytes : data length : 19 , icmp header : 8 , ip header : 20 seq: 27
RTT: 81.028999 milliseconds (81029 microseconds)
Successfully sent one packet : ICMP HEADER : 27 bytes, data length : 19 , icmp header : 8
^C
yasmin@yasmin-VirtualBox:~/Desktop/final$

```

כדי להפעיל את תוכנית `butter_ping` נקליד את הפקודה בטרמינל:

`sudo ./partb IP`

כאשר IP היא כתובת ה-IPv4 של מחשב היעד שנרצה לשלוח אליו את הפינג.

צריך לשים לב שלא צריך לאתחל את תוכנית Watchdog, מכון שהיא מופעלת לבד על ידי Better Ping (אם נריץ לבד זה פשוט לא יעבוד).

גם פה אם לא מקלידים קי בפקודת הרצה, אז ה `ip` הדיפולטיבי הוא 8.8.8.8.

דוגמא להרצה של חלק ב'- פה למטה.. IP יכול להיות כל דבר לא דווקא 8.8.8.8 כמו בדוגמא.

```
yasmin@yasmin-VirtualBox: ~/Downloads/212733836_315141...
yasmin@yasmin-VirtualBox:~/Downloads/212733836_315141242/212733836_315141242/cod
e$ make all
gcc -Wall -g ping.c -o parta
gcc -Wall -g better_ping.c -o partb
gcc -Wall -g watchdog.c -o watchdog
yasmin@yasmin-VirtualBox:~/Downloads/212733836_315141242/212733836_315141242/cod
e$ sudo ./partb 8.8.8.8
[sudo] password for yasmin:
conecct to IP: 8.8.8.8
Successfully received one packet from IP : 8.8.8.8 received one packet with 47
bytes , seq: 1 ,time: 33.061001 milliseconds and 33061 microseconds
Successfully received one packet from IP : 8.8.8.8 received one packet with 47
bytes , seq: 2 ,time: 28.118000 milliseconds and 28118 microseconds
Successfully received one packet from IP : 8.8.8.8 received one packet with 47
bytes , seq: 3 ,time: 24.931000 milliseconds and 24931 microseconds
Successfully received one packet from IP : 8.8.8.8 received one packet with 47
bytes , seq: 4 ,time: 22.481001 milliseconds and 22481 microseconds
^C
yasmin@yasmin-VirtualBox:~/Downloads/212733836_315141242/212733836_315141242/cod
e$
```

אם החיבור נכשל או לקח יותר מידי זמן לחבילה להגיע watchdog תשלח אות של נכשל לתוכנית Better Ping ותוכנית תסתיים .

```
yasmin@yasmin-VirtualBox: ~/Downloads/212733836_315141...
yasmin@yasmin-VirtualBox:~/Downloads/212733836_315141242/212733836_315141242/cod
e$ sudo ./partb 8.8.8.127
[sudo] password for yasmin:
conecct to IP: 8.8.8.127
Server 8.8.8.127 cannot be reached.
yasmin@yasmin-VirtualBox:~/Downloads/212733836_315141242/212733836_315141242/cod
e$
```

וכמו שהסברנו לא צריך להריץ את watchdog בנפרד זה לא יעבוד.

```
yasmin@yasmin-VirtualBox:~/Desktop/final$ sudo ./watchdog
^C
yasmin@yasmin-VirtualBox:~/Desktop/final$
```

# -Wireshark

נראה קצת איך התעבורה של התוכניות שכתבנו עוברות ברשת:

צילמנו מסך כאן למטה, כאשר מריצים את ה better\_ping :

ניתן לשים לב שבחבילות שכתוב בפרוטוקול ICMP אלו חבילות שמתקבלות כאשר שולחים או מקבלים הודעת ping.

פרוטוקול TCP זה כאשר watchdog מתקשר עם ה better\_ping , ניתן לשים לב שבפאקטות האלו גם ה source וגם ה destination הם 127.0.0.1 , כיוון ששני התוכניות מדברות מאותו מחשב.

החבילות שנשלחות בניהם מכילות את אותות אישור מוסכמים בניהם.

ב- destination כאשר כתוב 8.8.8.8 זה בעצם כאשר אנחנו שולחים לגוגל פינג, והמחשב שלנו נמצא ב-IP 10.0.2.15.

כאשר זה הפוך (ה src הופך ל dest ) זה מחזיר תגובת פינג מגוגל אלינו.

ניתן לראות את חבילות ה ICMP שהם או request או reply , אלו חבילות שליחה וקבלה פינג בהתאמה.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	0.0.0.0	ICMP	63	Echo (ping) request id=0x1200, seq=2560/10, ttl=64 (reply in 2)
2	0.025796946	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=2560/10, ttl=113 (request in 1)
3	0.051221114	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=0 TSval=3207463842 TSecr=3207463842
4	0.055039912	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=1 Ack=2 Win=512 Len=0 TSval=3207463842 TSecr=3207463842
5	1.056062491	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=2816/11, ttl=64 (reply in 6)
6	1.087028996	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=2816/11, ttl=113 (request in 5)
7	1.087709446	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=2 Ack=1 Win=512 Len=0 TSval=3207464904 TSecr=3207463842
8	1.087804489	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=1 Ack=3 Win=512 Len=0 TSval=3207464904 TSecr=3207464904
9	2.102452877	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=3072/12, ttl=64 (reply in 10)
10	2.132429893	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=3072/12, ttl=113 (request in 9)
11	2.132628006	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=3 Ack=1 Win=512 Len=0 TSval=3207465949 TSecr=3207464904
12	2.132645501	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=1 Ack=4 Win=512 Len=0 TSval=3207465949 TSecr=3207465949
13	3.342718127	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=3328/13, ttl=64 (reply in 14)
14	3.375995522	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=3328/13, ttl=113 (request in 13)
15	3.376342208	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=4 Ack=1 Win=512 Len=0 TSval=3207467192 TSecr=3207465949
16	3.376353195	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=1 Ack=5 Win=512 Len=0 TSval=3207467192 TSecr=3207467192
17	4.385818046	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=3584/14, ttl=64 (reply in 20)
18	4.410956976	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=3584/14, ttl=113 (request in 19)
19	4.416137400	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=5 Ack=1 Win=512 Len=0 TSval=3207468232 TSecr=3207467192
20	4.416145337	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=1 Ack=6 Win=512 Len=0 TSval=3207468232 TSecr=3207468232
21	5.438426629	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=3840/15, ttl=64 (reply in 24)
22	5.468827002	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=3840/15, ttl=113 (request in 23)
23	5.468950787	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=6 Ack=1 Win=512 Len=0 TSval=3207469285 TSecr=3207468232
24	5.468963107	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=1 Ack=7 Win=512 Len=0 TSval=3207469285 TSecr=3207469285
25	6.539106590	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=4096/16, ttl=64 (reply in 28)
26	6.562314797	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=4096/16, ttl=113 (request in 27)
27	6.572171577	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=7 Ack=1 Win=512 Len=0 TSval=3207470388 TSecr=3207469285
28	6.572185705	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=1 Ack=8 Win=512 Len=0 TSval=3207470388 TSecr=3207470388
29	7.576717946	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=4352/17, ttl=64 (reply in 34)
30	7.627066787	127.0.0.1	127.0.0.1	TCP	69	3000 → 52736 [PSH, ACK] Seq=1 Ack=8 Win=512 Len=0 TSval=3207471443 TSecr=3207470388
31	7.670847451	127.0.0.1	127.0.0.1	TCP	68	52736 → 3000 [ACK] Seq=8 Ack=2 Win=512 Len=0 TSval=3207471407 TSecr=3207471443
32	7.689551789	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=4352/17, ttl=113 (request in 31)
33	7.689600031	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=8 Ack=2 Win=512 Len=0 TSval=3207471506 TSecr=3207471443
34	7.689617170	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=2 Ack=9 Win=512 Len=0 TSval=3207471506 TSecr=3207471506
35	8.693365208	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=4608/18, ttl=64 (reply in 38)
36	8.725815260	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=4608/18, ttl=113 (request in 37)
37	8.725938339	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=9 Ack=2 Win=512 Len=0 TSval=3207472542 TSecr=3207471506
38	8.725956005	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=2 Ack=10 Win=512 Len=0 TSval=3207472542 TSecr=3207472542
39	9.733943475	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=4864/19, ttl=64 (reply in 42)
40	9.757446800	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=4864/19, ttl=113 (request in 41)
41	9.757610404	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=10 Ack=1 Win=512 Len=0 TSval=3207473574 TSecr=3207472542
42	9.757674697	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=2 Ack=11 Win=512 Len=0 TSval=3207473574 TSecr=3207473574
43	10.759256457	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=5120/20, ttl=64 (reply in 46)

כאן ניתן לראות את פעילות ה ping\_better כאשר הוא מצליח לשלוח ולקבל פינג, הוא שולח ומקבל

חבילות icmp ושולח ומקבל מה watchdog אות אישור.

9	2.102452877	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=3072/12, ttl=64 (reply in 10)
10	2.132429893	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=3072/12, ttl=113 (request in 9)
11	2.132628006	127.0.0.1	127.0.0.1	TCP	69	52736 → 3000 [PSH, ACK] Seq=3 Ack=1 Win=512 Len=0 TSval=3207465949 TSecr=3207464904
12	2.132645501	127.0.0.1	127.0.0.1	TCP	68	3000 → 52736 [ACK] Seq=1 Ack=4 Win=512 Len=0 TSval=3207465949 TSecr=3207465949

```

[Ip.src==127.0.0.1 && Ip.dst==127.0.0.1] || [Ip.src==8.8.8.8 && Ip.dst==10.0.2.15] || [Ip.src==10.0.2.15 && Ip.dst==8.8.8.8]
No.    Time           Source            Destination      Protocol  Length  Info
--
1    0.000000000      10.0.2.15        8.8.8.8         ICMP      63      Echo (ping) request  id=0x1200, seq=2560/10, ttl=64 (reply in 2)
2    0.025796946      8.8.8.8          10.0.2.15      ICMP      63      Echo (ping) reply    id=0x1200, seq=2560/10, ttl=113 (request in 1)
3    0.025922114      127.0.0.1        127.0.0.1      TCP       69      52736 -> 3000 [PSH, ACK] Seq=1 Ack=1 Min=512 Len=1 TSval=3207463842 TSecr=3207462801
4    0.025939912      127.0.0.1        127.0.0.1      TCP       68      3000 -> 52736 [ACK] Seq=1 Ack=2 Min=512 Len=0 TSval=3207463842 TSecr=3207463842
5    0.1056062491     10.0.2.15        8.8.8.8         ICMP      63      Echo (ping) request  id=0x1200, seq=2816/11, ttl=64 (reply in 0)
6    0.1163289596     10.0.2.15        10.0.2.15      ICMP      63      Echo (ping) reply    id=0x1200, seq=2816/11, ttl=113 (request in 5)
7    0.1087789446     127.0.0.1        127.0.0.1      TCP       69      52736 -> 3000 [PSH, ACK] Seq=2 Ack=1 Min=512 Len=1 TSval=3207464994 TSecr=3207463842
8    0.1087884489     127.0.0.1        127.0.0.1      TCP       68      3000 -> 52736 [ACK] Seq=1 Ack=3 Min=512 Len=0 TSval=3207464994 TSecr=3207464994
9    0.1624523771     10.0.2.15        8.8.8.8         ICMP      63      Echo (ping) request  id=0x1200, seq=3072/12, ttl=64 (reply in 10)
10   0.1624523771     10.0.2.15        8.8.8.8         ICMP      63      Echo (ping) reply    id=0x1200, seq=3072/12, ttl=113 (request in 9)
11   0.1624523771     10.0.2.15        8.8.8.8         ICMP      63      Echo (ping) request  id=0x1200, seq=3072/12, ttl=64 (reply in 10)
12   0.136262806     127.0.0.1        127.0.0.1      TCP       69      52736 -> 3000 [PSH, ACK] Seq=3 Ack=1 Min=512 Len=1 TSval=3207465049 TSecr=3207464994

```

Wireshark - Packet 1 - any

```

- Frame 1: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface any, id 0
- Linux cooked capture v1
  Packet type: Sent by us (4)
  Link-layer address type: Ethernet (1)
  Link-layer address length: 0
  Link-layer address length: 0
  Source: PcsComp_u6-41:5d (08:00:27:7a:06:41:5d)
  Unused: 0000
  Protocol: IPv4 (0x8000)
- Internet Protocol Version 4, Src: 10.0.2.15, Dst: 8.8.8.8
- Internet Control Message Protocol
  Type: 0 (Echo (ping) request)
  Code: 0
  Checksum: 0xa436 [correct]
  [Checksum Status: Good]
  Identifier (BE): 4608 (0x1200)
  Identifier (LE): 18 (0x0012)
  Sequence Number (BE): 2560 (0xa000)
  Sequence Number (LE): 10 (0x000a)
  [Response frame: 2]
- Data (18 bytes)
  Data: 54666973206973207466652070096e672e0a00
  [Length: 19]

```

0000 00 04 00 01 00 00 00 00 27 a6 41 5d 00 00 00 00 .....A.....  
0010 45 00 00 27 7f e0 40 00 40 01 9e cf 0a 00 02 0f E...@...  
0020 00 00 00 00 00 a4 30 12 00 0a 00 54 68 69 73 .....0...[N]  
0030 54 66 72 20 74 66 65 20 70 09 6e 67 2e 0a 00 15 Dr ping

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

[[ ip.src=127.0.0.1 & ip.dst=127.0.0.1 ]][[ ip.src=8.8.8.8 & ip.dst=10.0.2.15 ]][[ ip.src=127.0.0.1 & ip.dst=8.8.8.8 ]]

No.	Time	Source	Destination	Protocol	Length	Info
1	4.2.872967188	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) request seq=256/71, ttl=64 (request in 4)
2	4.2.872967313	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=256/71, ttl=64 (reply in 4)
3	4.2.872967348	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=512/72, ttl=64 (request in 5)
4	4.2.872967405	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=512/72, ttl=64 (reply in 5)
5	4.2.879709311	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=768/3, ttl=64 (request in 6)
6	4.2.884596701	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=768/3, ttl=64 (reply in 6)
7	4.2.892469954	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=1024/4, ttl=64 (request in 7)
8	4.2.889295572	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=1024/4, ttl=64 (reply in 7)
9	11.2.889380246	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=1280/5, ttl=64 (request in 12)
10	11.2.88453227	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=1280/5, ttl=64 (reply in 12)
11	13.2.894721446	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=1536/6, ttl=64 (request in 14)
12	14.2.900357395	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=1536/6, ttl=64 (reply in 14)
13	14.2.901498921	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=1792/7, ttl=64 (request in 15)
14	14.2.900604455	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=1792/7, ttl=64 (reply in 15)
15	17.2.906077998	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=2048/8, ttl=64 (request in 18)
16	17.2.90177553	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=2048/8, ttl=64 (reply in 18)
17	17.2.912107536	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=2304/9, ttl=64 (request in 17)
18	20.2.917232875	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=2304/9, ttl=64 (reply in 20)
19	21.2.9348591	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=2560/10, ttl=64 (request in 22)
20	22.2.922770358	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=2560/10, ttl=64 (reply in 22)
21	23.2.922848915	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=2816/11, ttl=64 (request in 21)
22	24.2.928148114	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=2816/11, ttl=64 (reply in 24)
23	25.2.928242409	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=3072/12, ttl=64 (request in 23)
24	26.2.933156276	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=3072/12, ttl=64 (reply in 26)
25	27.2.933237731	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=3328/13, ttl=64 (request in 28)
26	28.2.938033433	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=3328/13, ttl=64 (reply in 27)
27	29.2.938710441	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=3584/14, ttl=64 (request in 30)
28	30.2.943040958	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=3584/14, ttl=64 (reply in 29)
29	31.2.944040417	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=3840/15, ttl=64 (request in 31)
30	32.2.949198149	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=3840/15, ttl=64 (reply in 32)
31	34.2.949203793	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=4096/16, ttl=64 (request in 34)
32	34.2.945553334	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=4096/16, ttl=64 (reply in 33)
33	35.2.954664688	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=4352/17, ttl=64 (request in 35)
34	36.2.959847942	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=4352/17, ttl=64 (reply in 36)
35	37.2.959292143	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=4608/18, ttl=64 (request in 37)
36	38.2.964913014	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=4608/18, ttl=64 (reply in 38)
37	39.2.964971624	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=4864/19, ttl=64 (request in 40)
38	40.2.970452087	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply seq=4864/19, ttl=64 (reply in 39)
39	41.2.970581998	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request seq=5120/20, ttl=64 (request in 42)
40						



כאשר לא מתקבלת תגובת פינג ב better ping , אז התוכנית watchdog לאחר 10 שניות תסגור את החיבור.  
ניתן לראות שיש רק בקשת פינג מקו 8.8.8.127 (בוורוד) ולכן ולא מתקבלת תגובה ועוברים העשר שניות.  
כל העשר שניות האלו הwatchdog בודק האם better ping שלחה לו את האות המוסכם הפינג התקבל, לבסוף מכוון שלא התקבל האות, watchdog שולח לbetter\_ping סיום שגורמת לסגור את התוכנית.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	53396 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=514420101 TSecr=0 WS=128
2	0.000027388	127.0.0.1	127.0.0.1	TCP	76	3000 → 53396 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=514420101 TSecr=514420101
3	0.000059303	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=514420101 TSecr=514420101
4	0.000121667	10.0.2.15	8.8.8.127	ICMP	63	Echo (ping) request id=0x1200, seq=256/1, ttl=64 (no response found)
5	0.000183514	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1 TSval=514420101 TSecr=514420101
6	0.000196378	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=514420101 TSecr=514420101
7	1.000369535	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=2 Ack=1 Win=65536 Len=1 TSval=514421101 TSecr=514420101
8	1.000381053	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=3 Win=65536 Len=0 TSval=514421101 TSecr=514421101
9	2.000022414	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=3 Ack=1 Win=65536 Len=1 TSval=514422102 TSecr=514421101
10	2.000032368	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=4 Win=65536 Len=0 TSval=514422102 TSecr=514421102
11	3.001605600	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=4 Ack=1 Win=65536 Len=1 TSval=514423103 TSecr=514422102
12	3.001623857	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=5 Win=65536 Len=0 TSval=514423103 TSecr=514423103
13	4.002034053	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=5 Ack=1 Win=65536 Len=1 TSval=514424103 TSecr=514423103
14	4.002045650	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=6 Win=65536 Len=0 TSval=514424103 TSecr=514424103
15	5.002207457	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=6 Ack=1 Win=65536 Len=1 TSval=514425103 TSecr=514424103
16	5.002219619	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=7 Win=65536 Len=0 TSval=514425103 TSecr=514425103
17	6.017848300	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=7 Ack=1 Win=65536 Len=1 TSval=514426119 TSecr=514425103
18	6.017850007	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=8 Win=65536 Len=0 TSval=514426119 TSecr=514426119
19	7.017996726	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=8 Ack=1 Win=65536 Len=1 TSval=514427119 TSecr=514426119
20	7.018008272	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=9 Win=65536 Len=0 TSval=514427119 TSecr=514427119
21	8.021082116	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=9 Ack=1 Win=65536 Len=1 TSval=514428122 TSecr=514427119
22	8.021091797	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=10 Win=65536 Len=0 TSval=514428122 TSecr=514428122
23	9.020359961	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=10 Ack=1 Win=65536 Len=1 TSval=514429129 TSecr=514428122
24	9.020372167	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=11 Win=65536 Len=0 TSval=514429129 TSecr=514429129
25	10.0205616987	127.0.0.1	127.0.0.1	TCP	69	3000 → 53396 [PSH, ACK] Seq=11 Ack=1 Win=65536 Len=1 TSval=514430130 TSecr=514429129
26	10.020556048	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [ACK] Seq=1 Ack=12 Win=65536 Len=0 TSval=514430130 TSecr=514430130
27	10.0205700574	127.0.0.1	127.0.0.1	TCP	68	3000 → 53396 [FIN, ACK] Seq=1 Ack=13 Win=65536 Len=0 TSval=514430130 TSecr=514430130
28	10.0207800574	127.0.0.1	127.0.0.1	TCP	68	53396 → 3000 [FIN, ACK] Seq=1 Ack=13 Win=65536 Len=0 TSval=514430130 TSecr=514430130
29	10.0207804913	127.0.0.1	127.0.0.1	TCP	68	3000 → 53396 [ACK] Seq=13 Ack=2 Win=65536 Len=0 TSval=514430130 TSecr=514430130

כאשר אנחנו עוצרים את better ping באמצע, (על ידי C+control) אנחנו בעצם יכולים לפגוע בפאקטות שבעת העצירה היו באמצע להישלח. ולכן הפקטה האחרונה פה צבועה באדום כי היא נפגעה מהסגירה הפתאומית.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	56988 → 3000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=514910048 TSecr=0 WS=128
2	0.000022024	127.0.0.1	127.0.0.1	TCP	76	3000 → 56988 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=514910048 TSecr=514910048 WS=128
3	0.000044967	127.0.0.1	127.0.0.1	TCP	68	56988 → 3000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=514910048 TSecr=514910048
4	0.000090049	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=256/1, ttl=64 (reply in 7)
5	0.00019326	127.0.0.1	127.0.0.1	TCP	69	3000 → 56988 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1 TSval=514910049 TSecr=514910048
6	0.000534697	127.0.0.1	127.0.0.1	TCP	68	56988 → 3000 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=514910049 TSecr=514910049
7	0.012579319	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=256/1, ttl=118 (request in 4)
8	0.012827930	127.0.0.1	127.0.0.1	TCP	69	56988 → 3000 [PSH, ACK] Seq=1 Ack=2 Win=65536 Len=1 TSval=514910061 TSecr=514910049
9	0.012843963	127.0.0.1	127.0.0.1	TCP	68	3000 → 56988 [ACK] Seq=2 Ack=2 Win=65536 Len=0 TSval=514910061 TSecr=514910061
10	1.016262336	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=512/2, ttl=64 (reply in 11)
11	1.016262336	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) reply id=0x1200, seq=512/2, ttl=118 (request in 10)
12	1.020807087	127.0.0.1	127.0.0.1	TCP	69	56988 → 3000 [PSH, ACK] Seq=2 Ack=2 Win=65536 Len=1 TSval=514911077 TSecr=514910061
13	1.020826946	127.0.0.1	127.0.0.1	TCP	68	3000 → 56988 [ACK] Seq=2 Ack=3 Win=65536 Len=0 TSval=514911077 TSecr=514911077
14	2.040177821	10.0.2.15	8.8.8.8	ICMP	63	Echo (ping) request id=0x1200, seq=768/3, ttl=64 (reply in 15)
15	2.050782285	8.8.8.8	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=768/3, ttl=118 (request in 14)
16	2.050792538	127.0.0.1	127.0.0.1	TCP	69	56988 → 3000 [PSH, ACK] Seq=3 Ack=2 Win=65536 Len=1 TSval=514912107 TSecr=514911077
17	2.050800739	127.0.0.1	127.0.0.1	TCP	68	3000 → 56988 [ACK] Seq=2 Ack=4 Win=65536 Len=0 TSval=514912107 TSecr=514912107
18	2.0508074072	127.0.0.1	127.0.0.1	TCP	68	3000 → 56988 [RST, ACK] Seq=3 Ack=4 Win=65536 Len=0 TSval=514912107 TSecr=514912107

סוף.