מטלה 4- מגישות: אגסי נועה- , נדב רוני- 325730190 ואיזביצקי בר - 212138457

חלק א-

בחלק זה השתמשנו בקוד הבסיס של ping שראינו בהרצאה,
הקוד בנוי מהקבצים הבאים- ping.h, ping.c, makefile
הרצת הקוד -
make
sudo ./ping –t<> -c<> -a <> -f

הסבר של הקוד: (הוספות ושינויים לקוד שראינו בתרגול)

בקובץ Ping.h יש – את כל ההצהרות של הפונקציות שאנו משתמשות בהם בקובץ ping.c , המבנים שנתמש (stracr) והשמה של קבצי h נוספים שקיימים במערכת.

המבנים שהגדרנו -

מבנה של חבילות ה ICMP-וה ICMPv6-שישלחו ויתקבלו
במהלך פעולת הPing.

```
struct icmp_packet {
    struct icmphdr icmp_hdr;
    char data[PACKET_SIZE - sizeof(struct icmphdr)];
};

struct icmp6_packet {
    struct icmp6_hdr icmp6_hdr;
    char data[PACKET_SIZE - sizeof(struct icmp6_hdr)];
};
```

מימוש הפעולות בקובץ ping.c-

פונקציית send_ping-

תחילה, הגדרת המשתנים הבאים-
start ו-end הם משתנים מסוג timeval,
שמשמשים כדי לשמור זמן.
packet ו-packet6 הם מבני נתונים שמייצגים
את החבילות שישלחו ב-IPv4 ו-IPv6.
gettimeofday(&start, NULL)- שומר את הזמן
הנוכחי (זמן שליחת הבקשה) במשתנה start.

```
void send_ping(int sockfd, struct sockaddr_in *dest_addr, struct sockaddr_in6 *dest_addr6, int ttl, int seq_num,
               double *total_time, int *received, double *min_time, double *max_time, int is_ipv6) {
    struct timeval start, end; // To measure RTT
    struct icmp_packet packet; // ICMP packet for IPv4
    struct icmp6_packet packet6; // ICMPv6 packet for IPv6

    gettimeofday(&start, NULL); // Record the start time
```

אם המשתנה is_ipv6 הוא true, הפונקציה שולחת בקשה ב-IPv6.
כל חבילה ICMP ב-IPv6 מכילה מידע כמו סוג הבקשה (ECHO_REQUEST), מזהה החבילה (icmp6_id), ורצף החבילה (icmp6_seq).
memset(packet6.data, 0, sizeof(packet6.data))
מאפס את הנתונים בתוך החבילה.
מחשב את ה-checksum של החבילה.
שולח את החבילה לכתובת היעד ב-IPv6 באמצעות הפונקציה sendto.

```
if (is_ipv6) {
    // Prepare the ICMPv6 packet
    packet6.icmp6_hdr.icmp6_type = ICMP6_ECHO_REQUEST;
    packet6.icmp6_hdr.icmp6_id = getpid();
    packet6.icmp6_hdr.icmp6_seq = seq_num;
    memset(packet6.data, 0, sizeof(packet6.data));
    packet6.icmp6_hdr.icmp6_cksum = 0;
    packet6.icmp6_hdr.icmp6_cksum = checksum(&packet6, sizeof(packet6));

    // Send the ICMPv6 packet
    if (sendto(sockfd, &packet6, sizeof(packet6), 0, (struct sockaddr *)dest_addr6, sizeof(*dest_addr6)) <= 0) {
        perror("sendto failed");
        return;
    }
```

אם המשתנה is_ipv6 הוא false, הפונקציה שולחת בקשה ב-IPv4.
כמו ב-IPv6, גם כאן יש להגדיר את סוג הבקשה (ECHO), את מזהה החבילה (id), ואת רצף החבילה (sequence).
- memset(packet.data, 0, sizeof(packet.data)) מאפס את הנתונים בחבילה.
מחשב את ה-checksum של החבילה, כמו ב-IPv6.
שולח את החבילה לכתובת היעד ב-IPv4.

```
} else {
    // Prepare the ICMP packet
    packet.icmp_hdr.type = ICMP_ECHO;
    packet.icmp_hdr.code = 0;
    packet.icmp_hdr.un.echo.id = getpid();
    packet.icmp_hdr.un.echo.sequence = seq_num;
    memset(packet.data, 0, sizeof(packet.data));
    packet.icmp_hdr.checksum = 0;
    packet.icmp_hdr.checksum = checksum(&packet, sizeof(packet));

    // Send the ICMP packet
    if (sendto(sockfd, &packet, sizeof(packet), 0, (struct sockaddr *)dest_addr, sizeof(*dest_addr)) <= 0) {
        perror("sendto failed");
        return;
    }
}
```

pollfd הוא מבנה שמכיל מידע על הסוקט .
מחכה לקריאה (POLLIN) מהסוקט (fd).
הפונקציה poll מחכה לתשובה מהסוקט במשך

PING_TIMEOUT, ואם לא התקבלה תשובה תוך הזמן הזה, הפונקציה מחזירה 0.
אם poll נכשל, זורק שגיאה.
אם לא התקבלה תשובה בזמן שנקבע , מדפיס הודעת timeout ומפסיק את הפעולה.

```
struct pollfd pfd = {
    .fd = sockfd,
    .events = POLLIN
};

// Wait for a reply with a timeout
int poll_result = poll(&pfd, 1, PING_TIMEOUT * 1000);

if (poll_result < 0) {
    perror("poll failed");
    return;
} else if (poll_result == 0) {
    printf("Request timeout for icmp_seq=%d\n", seq_num);
    return;
}
```

קובע את גודל הכתובת של היעד (addr_len), תלוי בהאם זו כתובת IPv4 או IPv6.
ממתין לקבלת תשובה מהסוקט עם recvfrom. אם התשובה מתקבלת, היא מאוחסנת ב-buffer.

```
// Receive the reply
socklen_t addr_len = is_ipv6 ? sizeof(*dest_addr6) : sizeof(*dest_addr);
char buffer[1024];
int n = recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr *)(is_ipv6 ? (struct sockaddr *)dest_addr6 : (struct sockaddr *)dest_addr), &addr_len);
```

אם התקבלה תשובה, שומר את הזמן הנוכחי (זמן סיום) ב-end.
מחשב את הזמן שעבר בין שליחת הבקשה לקבלת התשובה.
אם הזמן שעבר קצר יותר מהזמן המינימלי הקודם, מעדכן את הזמן המינימלי.
אם הזמן שעבר ארוך יותר מהזמן המקסימלי הקודם, מעדכן את הזמן המקסימלי.
מוסיף את הזמן הכולל של כל הבקשות ומעדכן את מספר התשובות שהתקבלו.

```
if (n > 0) {
    gettimeofday(&end, NULL); // Record the end time

    double elapsed_time = (end.tv_sec - start.tv_sec) * 1000.0;
    elapsed_time += (end.tv_usec - start.tv_usec) / 1000.0;

    // Update RTT statistics
    if (elapsed_time < *min_time) {
        *min_time = elapsed_time;
    }
    if (elapsed_time > *max_time) {
        *max_time = elapsed_time;
    }

    *total_time += elapsed_time;
    (*received)++;
```

אם התקבלה תשובה ב-IPv6, מדפיס את כתובת ה-IP6 והזמן של החבילה.
אם התקבלה תשובה ב-IPv4, מדפיס את כתובת ה-IP4, את ה-TTL של החבילה ואת הזמן שלקח עד שהתשובה חזרה.

```
    // Print reply information
    if (is_ipv6) {
        char ip6_str[INET6_ADDRSTRLEN];
        inet_ntop(AF_INET6, &dest_addr6->sin6_addr, ip6_str, INET6_ADDRSTRLEN);
        printf("%d bytes from %s: icmp_seq=%d time=%.3fms\n",
               PACKET_SIZE,
               ip6_str,
               seq_num,
               elapsed_time);

    } else {
        struct iphdr *ip_hdr = (struct iphdr *)buffer;
        printf("%d bytes from %s: icmp_seq=%d ttl=%d time=%.3fms\n",
               PACKET_SIZE,
               inet_ntoa(dest_addr->sin_addr),
               seq_num,
               ip_hdr->ttl,
               elapsed_time);
    }
}
```

פונקציית המain-

אתחול המשתנים-
min_time מוגדר כ-DBL_MAX כדי להבטיח שהזמן המינימלי יישתנה במהלך הריצה.
count הוא מספר החבילות שיישלחו, ברירת המחדל היא 4.
ttl הוא ערך ה-(Time-to-Live (TTL של החבילות.
is_ipv6 מגדיר אם הכתובת היא IPv6 או IPv4.
flood מציין אם צריך לשלוח בקשות ללא השהייה.

```
int main(int argc, char *argv[]) {

    double total_rtt = 0, min_time = DBL_MAX, max_time = 0;
    int transmitted = 0, received = 0;
    struct sockaddr_in dest_addr;
    struct sockaddr_in6 dest_addr6;
    int sockfd, count = 4, ttl = 64;
    int is_ipv6 = 0, flood = 0;

    // Set up a timeout handler
    signal(SIGALRM, handle_timeout);
    alarm(PING_TIMEOUT);
```

לפי הקלט מעדכן את הנתונים –לדוגמא האם דגל f דולק או לאת מה מספר החבילות , האם ipv6 או ipv4 וכדומה..

```
for (int i = 1; i < argc; i++) {
    if (strcmp(argv[i], "-a") == 0 && i + 1 < argc) {
        if (strchr(argv[i + 1], ':')) {
            is_ipv6 = 1;
            inet_pton(AF_INET6, argv[i + 1], &dest_addr6.sin6_addr);
            dest_addr6.sin6_family = AF_INET6;
            i++;
        } else {
            inet_pton(AF_INET, argv[i + 1], &dest_addr.sin_addr);
            dest_addr.sin_family = AF_INET;
            i++;
        }
    } else if (strcmp(argv[i], "-t") == 0 && i + 1 < argc) {
        if (atoi(argv[i + 1]) == 6) {
            is_ipv6 = 1;
        }
        i++;
    } else if (strcmp(argv[i], "-c") == 0 && i + 1 < argc) {
        count = atoi(argv[i + 1]);
        i++;
    } else if (strcmp(argv[i], "-f") == 0) {
        flood = 1;
    }
}
```

יצירת סוקט RAW על פי סוג ה-IP שנבחר (IPv4 או IPv6). ובוחר פרוטוקול בהתאם ICMP או ICMPV6.
אם הסוקט לא נוצר בהצלחה, מדפיס הודעת שגיאה ומסיים את התוכנית.

```
sockfd = socket(is_ipv6 ? AF_INET6 : AF_INET, SOCK_RAW, is_ipv6 ? IPPROTO_ICMPV6 : IPPROTO_ICMP);
if (sockfd < 0) {
    perror("socket");
    exit(1);
}
```

שולח כמספר החבילות שנבחרו (count).
כל פעם שנשלחת חבילה, העדכון של מספר החבילות שנשלחו (transmitted) מתעדכן.
אם דגל f לא דולק, יש השהייה של שנייה בין שליחה לשליחה.

```
for (int i = 0; i < count; i++) {
    send_ping(sockfd, &dest_addr, &dest_addr6, ttl, i, &total_rtt, &received, &min_time, &max_time, is_ipv6);
    transmitted++;
    if (!flood) {
        sleep(1);
    }
}
```

אם התקבלו תשובות, מחשב את הזמן הממוצע (mean), הזמן המינימלי (min_time), המקסימלי (max_time), והחציון (mdev).
אם לא התקבלה אף תשובה, מדפיס " No reply received".

```
if (received > 0) {
    double mean = total_rtt / received;
    double mdev = 0.0;
    if (received > 1) {
        mdev = sqrt((total_rtt * total_rtt) / received - (mean * mean));
    }

    printf("--- ping statistics ---\n");
    printf("%d packets transmitted, %d received\n", transmitted, received);
    printf("rtt min/avg/max/mdev = %.3f/%.3f/%.3f/%.3fms\n", min_time, mean, max_time, mdev);
} else {
    printf("No reply received\n");
}
```

ובסיום, סוגר את הסוקט שנפתח ומחזיר 0 = הצלחה

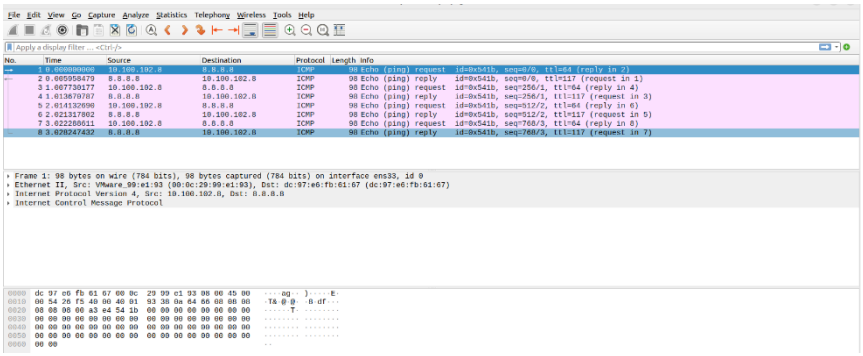
```
close(sockfd);
return 0;
```

הקלטות wireshark ותוצאות-

Ipv4- הרצה ללא דגלים מיוחדים (- רק עם t– ai-)-

```
bar@bar-virtual-machine:~/m$ sudo ./ping -t 4   -a 8.8.8.8
64 bytes from 8.8.8.8: icmp_seq=0 ttl=117 time=7.272ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=6.108ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=7.421ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=6.156ms
--- ping statistics ---
4 packets transmitted, 4 received
rtt min/avg/max/mdev = 6.108/6.739/7.421/11.673ms
```

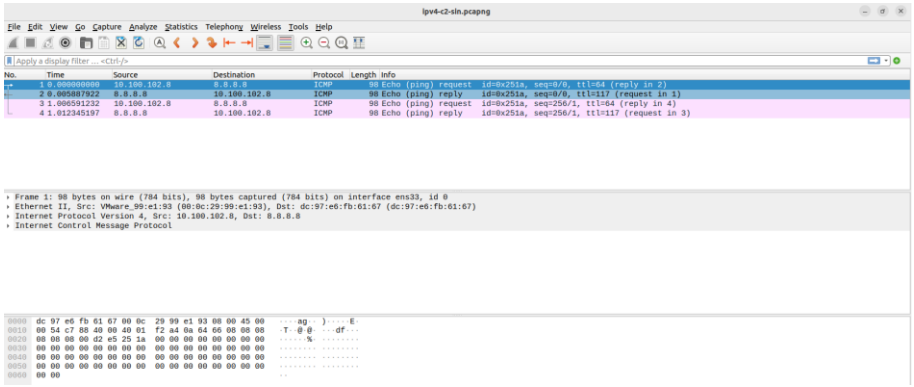
כאשר דגל C לא דולק הדיפולט הוא 4 חבילות.
ניתן לראות שעבור כל חבילה יש echo reply וecho request .



Ipv4- הרצה עם דגל c– דולק (=2)-

```
bar@bar-virtual-machine:~/m$ sudo ./ping -t 4 -c 2 -a 8.8.8.8
64 bytes from 8.8.8.8: icmp_seq=0 ttl=117 time=6.059ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=5.929ms
--- ping statistics ---
2 packets transmitted, 2 received
```

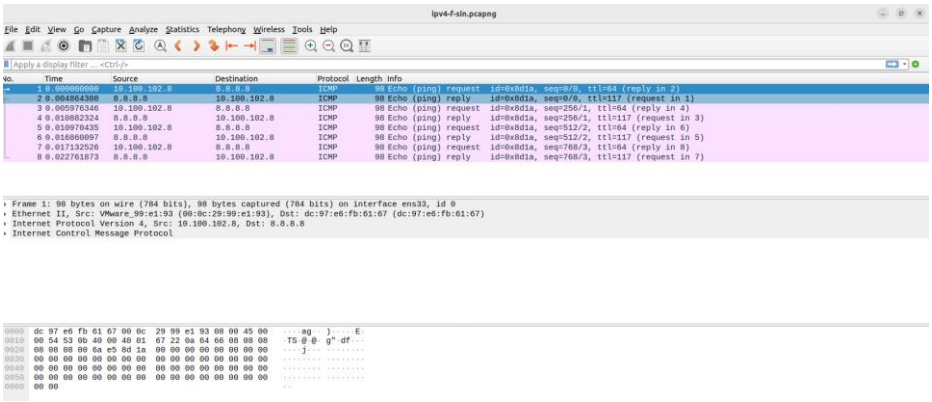
ניתן לראות שהוגדר c 2– ולכן נשלחו רק שתי חבילות.



Ipv4- הרצה עם דגל f– דולק-

```
bar@bar-virtual-machine:~/m$ sudo ./ping -t 4  -a 8.8.8.8 -f
64 bytes from 8.8.8.8: icmp_seq=0 ttl=117 time=5.066ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=4.979ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=6.004ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=6.534ms
--- ping statistics ---
4 packets transmitted, 4 received
rtt min/avg/max/mdev = 4.979/5.646/6.534/9.779ms
```

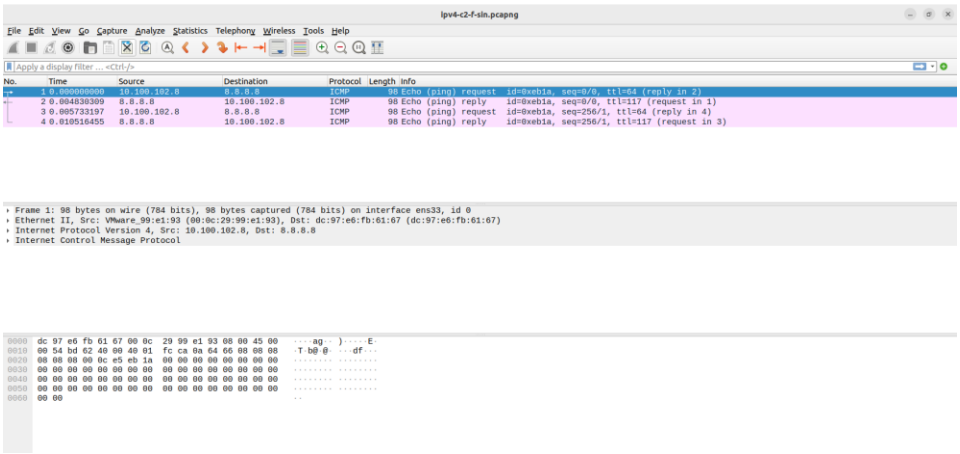
ניתן לראות שכשדגל c– לא פועל הדיפולט הוא 4 חבילות ובנוסף כאשר דגל f– דולק הזמנים מתקצרים.



Ipv4- הרצה עם שני הדגלים דולקים- (f-, c-(=2))-

```
bar@bar-virtual-machine:~/m$ sudo ./ping -t 4 -c 2  -a 8.8.8.8 -f
64 bytes from 8.8.8.8: icmp_seq=0 ttl=117 time=4.930ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=4.887ms
--- ping statistics ---
2 packets transmitted, 2 received
rtt min/avg/max/mdev = 4.887/4.909/4.930/4.909ms
```

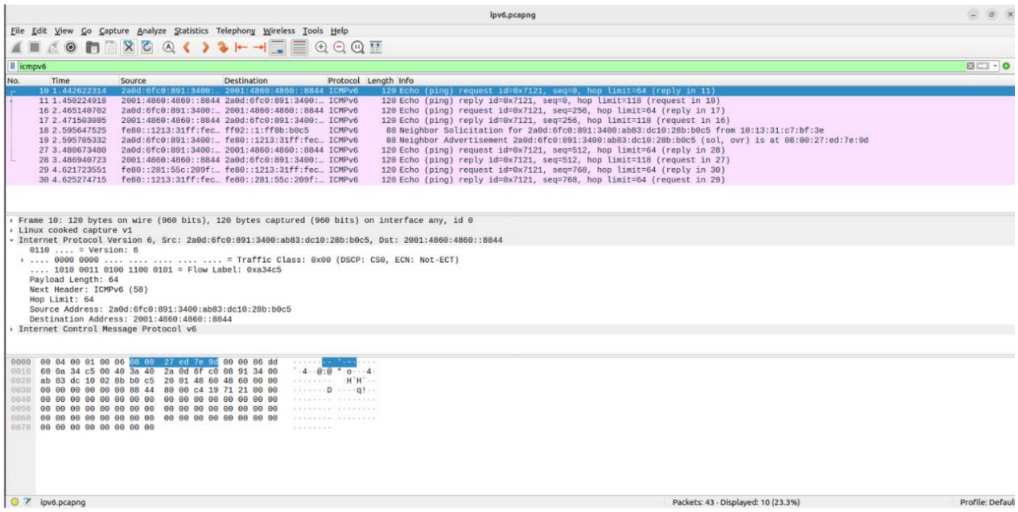
ניתן לראות שהזמנים קצרים יותר וכמות החבילות מוגבלת (-2).



Ipv6- הרצה ללא דגלים (-f, -c)-

```
dotan@Ubunto22:~/Downloads/partA$ sudo ./ping -a 2001:4860:4860::8844 -t 6
64 bytes from 2001:4860:4860::8844: icmp_seq=0 time=11.304ms
64 bytes from 2001:4860:4860::8844: icmp_seq=1 time=7.797ms
64 bytes from 2001:4860:4860::8844: icmp_seq=2 time=6.698ms
64 bytes from 2001:4860:4860::8844: icmp_seq=3 time=7.894ms
--- ping statistics ---
4 packets transmitted, 4 received
rtt min/avg/max/mdev = 6.698/8.423/11.304/14.589ms
```

ניתן לקרות בהקלטה שסיננו לפי icmp6 וגם בנתוני החבילה רשום שאנחנו על 6.
הקלטה זו היא ללא הדגלים ניתן לקרות שהדיפולט של c– נשאר 4.



Ipv6- הרצה עם דגלים-

```
dotan@Ubunto22:~/Downloads/partA$ sudo ./ping -a 2001:4860:4860::8844 -t 6 -c 8 -f
64 bytes from 2001:4860:4860::8844: icmp_seq=0 time=12.715ms
64 bytes from 2001:4860:4860::8844: icmp_seq=1 time=7.803ms
64 bytes from 2001:4860:4860::8844: icmp_seq=2 time=9.338ms
64 bytes from 2001:4860:4860::8844: icmp_seq=3 time=8.750ms
64 bytes from 2001:4860:4860::8844: icmp_seq=4 time=7.197ms
64 bytes from 2001:4860:4860::8844: icmp_seq=5 time=8.409ms
64 bytes from 2001:4860:4860::8844: icmp_seq=6 time=7.807ms
64 bytes from 2001:4860:4860::8844: icmp_seq=7 time=10.920ms
--- ping statistics ---
8 packets transmitted, 8 received
rtt min/avg/max/mdev = 7.197/9.117/12.715/24.122ms
```

ניתן לקרות בהקלטה שסיננו לפי icmp6 וגם בנתוני החבילה רשום שאנחנו על 6.
הקלטה זו היא עם הדגלים ולכן ניתן לראות שהגדרנו את c– להיות 8 (שונה מהדיפולט) והפרש הזמנים בין החבילות יותר קטן (בעמודת הזמן)