תקשורת ומחשוב – תשפ"א - סמס' א' - מטלה שלישית

אין להעתיק אחד מהשני. המטלה צריכה להיות במילים שלכם. העתקת "קופי-פייסט" ממקומות כמו ויקיפדיה ודומיו תיחשב כהעתקה. את המטלה יש להגיש עד התאריך המצויין בתיבת ההגשה כאשר כל הקבצי המטלה דחוסים לקובץ ZIP ששמו הוא מס' ת.ז. של המגיש\ה. הגשה מאוחרת תתאפשר עד ארבעה ימים לאחר מועד ההגשה הנקוב, כאשר על כל יום איחור, ירדו 4 נקודות מציון המטלה.

במטלה זו תעמיקו ב-DNS וב-TCP, תרחיבו על העקרונות שלהם ומימושם. כמו כן, תשתמשו בסוקטים למימוש תוכנת מחקר משלכם שתחקור עקרון של TCP (אולי החשוב ביותר שלו), Congestion Control.

חלק א' - DoH

DNS over HTTPS

הינה שיטה חדשה יותר המוצעת לשימוש. כמו שמרמז השם, זוהי שיטת תשאול DNS בחיבור HTTPS מאובטח.

1. הציגו יתרון אחד לשימוש ב-DoH והסבירו אותו (כמובן, מעבר לעובדה שהוא מאובטח ומוצפן)
2. הציגו והסבירו על שני חסרונות לשימוש בשיטת DoH לעומת DNS הרגיל.
3. בחרו אחד מהחסרונות משאלה (2), הציעו דרך למתן\לעקוף\לפתור חיסרון זה והסבירו אותה.
4. ישנן 4 דרכים בהן ניתן לשלב את שיטת ה-DoH באינטרנט שלנו:
   a. מימוש DoH ברמת האפליקציות (לדוגמא: לעדכן את קוד הדפדפן כך שישלחו שאילתות דרך HTTPS)
   b. מימוש DoH ברמת שרת proxy* ברשת (מהמחשב לשרת נשלח לפורט 53 והלאה, כבר 443)
   c. מימוש DoH ברמת שרת proxy מקומי (על המכונה רץ שרת proxy)
   d. התקנת plugin הממש DoH ברמת הגדרות המחשב ("מעכשיו, אתה שולח רק DoH")

   כתבו השוואה בין כל ארבעת השיטות, בהשוואתכם הראו יתרונות וחסרונות לכל שיטה והציגו מהי, לדעתכם, השיטה המועדפת מבין הארבעה. כלומר, הציגו את השיטה בה, לדעתכם, היתרונות הגדולים ביותר לעומת החסרונות הקטנים ביותר.

5. נניח שאנו ברשת שקיים בה איבוד פקטות (packet loss) באחוז לא ידוע ואנו רוצים לטעון דף שצריך 25 שאילתות כדי לבקש את כל המשאבים שבו. הציגו יתרון ברור שיש ל-DoH לעומת 53Do. (רמז: מנגנון הקיים ב-TCP)

* שרת פרוקסי(proxy), במילים פשוטות, הינו שרת שאנו בוטחים בו שתפקידו לטפל, במקומנו, באינטראקציות עם שרתים חיצוניים.

חלק ב' - Congestion Control

בחלק זה נחקור את ההבדלים בין אלגוריתמי Congestion Control (להלן: "CC"), ספציפית, בין "cubic", שהוא הדפולטיבי, לבין "reno" .

אנו נצטרך קובץ גדול לכן נייצר אחד. מצורף למטלה קובץ בשם create_large_file.py, הריצו אותו והוא ייצר קובץ. זה יהיה הקובץ אותו נשלח ונקבל במהלך התרגיל (להלן: "הקובץ" \ "קובץ").

בשאלה זו עליכם לרשום שני קבצים: sender.c ו- measure.c. הראשונה תשלח את הקובץ שלנו והשניה תקבל אותו ותמדוד כמה זמן לקח לקבל את כולו. לאחר מכן, תבצעו שינוי באלגוריתם ה-CC שלהן ויחזרו על השליחה וחישוב . כדי לדמות איבוד פקטות, אנו נשתמש **בכלי של לינוקס** שנקרא tc.
אם הכלי אינו קיים אצלכם תוכלו להוריד אותו בעזרת הפקודה הבאה:

```
sudo apt install iproute
```

לאחר מכן נייצר איבוד פאקטות ראנדומאלי:

```
sudo tc qdisc add dev lo root netem loss 10%
```

אנו נרצה לשנות את אחוז איבוד הפאקטות בכל סיבוב דגימות:

```
sudo tc qdisc change dev lo root netem loss XX%
```

כאשר נרצה למדוד 10,15,20,25,30 אחוזי איבוד.

כשנסיים את התרגיל, נריץ את הפקודה הבאה על מנת לבטל את איבוד הפקטות:

```
sudo tc qdisc del dev lo root netem
```

<u>הקבצים:</u>

sender.c

בתכנית זאת תממשו שליחה של קובץ דרך סוקט.
אופן פעולת התכנית:

1. פתיחת סוקט TCP.
2. יצירת חיבור עם measure
3. שליחת הקובץ חמש פעמים.
4. שינוי אלגוריתם CC.
5. שליחת הקובץ חמש פעמים.
6. סגירת חיבור.

Measure.c

בתכנית זאת תקבלו את הקובץ הנשלח מהתכנית sender דרך סוקט.

אופן פעולת התכנית:

1. פתיחת סוקט TCP.
2. האזנה לחיבורים נכנסים.
3. קבלת חיבור מ-sender.
4. קבלת הקובץ חמש פעמים תוך כדי מדידת זמן קבלה. (אין צורך לשמור את הקובץ בקבלתו)
5. ביצוע ממוצע לזמנים שנדגמו.
6. שינוי אלגוריתם CC.
7. קבלת הקובץ חמש פעמים תוך כדי מדידת זמן קבלה .(אין צורך לשמור את הקובץ בקבלתו)
8. ביצוע ממוצע לזמנים שנדגמו.
9. הדפסת הזמנים שנמדדו .
10. סגירת חיבור.

לאחר הרצת התכנית ומדידת ממוצע זמני ההגעה לכל אחד מאחוזי האיבוד המצויינים מעלה, יש לאגד את הזמנים בטבלה (זמן הגעה ממוצע \ איבוד פאקטות באחוזים).

- את התכניות יש לממש בסביבת לינוקס
- יש לצרף צילום מסך של ריצת כל פקודה.

דוגמא לתכנית המשנה CC בזמן ריצה: אתם תשתמשו בחלק ממנו בקוד שלכם

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main(int argc, char **argv) {
	char buf[256];
	socklen_t len;
	int sock = socket(AF_INET, SOCK_STREAM, 0);
	if (sock == -1) {
		perror("socket");
		 return -1;
	 }

	len = sizeof(buf);
	if (getsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, buf, &len) != 0) {
		perror("getsockopt");
		return -1;
	}

	printf("Current: %s\n", buf);



	strcpy(buf, "reno");
	len = strlen(buf);
	if (setsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, buf, len) != 0) {
		perror("setsockopt");
		return -1;
	}
	len = sizeof(buf);
	if (getsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, buf, &len) != 0) {
		perror("getsockopt");
		return -1;
	}
	printf("New: %s\n", buf);
	close(sock);
	return 0;
}
```

<u>בהצלחה!</u>