תקשורת ומחושב – מטלה רביעית
אופיר רובין
ת.ז: 212831879

חלק א' – DoH

1. אחד היתרונות של הפרוטוקול הוא פרטיות. פרוטוקול ה-DoH הינו מאובטח ומוצפן ולמעשה הבקשות מועברות על גבי HTTPS באמצעות TCP. לכן כל בקשה שנשלחת נראית באינטרנט כמו חבילת HTTPS, זאת בנוסף לכך שתוכן ההודעה מוצפן ולכן מקשה על הצופים ברשת להבין כי הבקשה למעשה הינה DNS ולא HTTPS. (על ידי שימוש בTCP אנו מרוויחים גם את יתרונותיו כמו אמינות ועוד).
2. שני חסרונות בשימוש בפרוטוקול DoH הם:
מהירות – מאחר והפרוטוקול מוצפן ומתבסס בין היתר על TCP תהליך הבקשה והתגובה בהכרח איטי יותר מהליך המבוסס UDP.
אבטחה (ביטחון המשתמש) – חוסר האפשרות לחסימת חיבורים לא רצויים באופן דיפולטיבי שכן הבקשות מוצפנות הרשת המקומית (-הgetway ליתר דיוק, כמו הראוטר הביתי) שלנו שבד"כ מגנה עלינו על ידי חסימה וסינון חיבורים לא רצויים (כמו אתרים לא מאובטחים, כתובות חשודות וכו') לא יכולה לעשות זאת כאשר ההודעות מוצפנות – היא לא יודעת את התוכן שלהם ולא יכולה לשנות אותו.
3. ניתן למתן את איטיות הפרוטוקול באמצעות שימוש בCaching, איחוד בקשות, פיצול לשרתים שונים, ייעול הבקשות וכו'. בנוסף, ניתן לעשות את סינון הכתובות ברמת שכבת האפליקציה או לחילופין ניתן יהיה ליצור חלופה בה כאשר מתבקשת כתובת משרת, שאינה מומלצת או רצויה והוא יודע זאת הוא יחזיר את התשובה עם אזהרה.
4. השוואה בין המימושים:

מימוש	שוני פעולה מDNS	אבטחה	פרטיות
ברמת האפליקציה	האפליקציה אינה מבקשת כתובות על גבי DNS אלא בעצמה שולחת בקשות DoH על מנת להשיג את הכתובות	ההודעות נשלחות כבר מהמחשב באופן מאובטח	גורם חיצוני שאינו היעד אינו יכול לקרוא את תוכן ההודעה
ברמת שרת proxy ברשת	הבקשות נשלחות על גבי DNS לשרת ייעודי שבעצמו שולח על גבי DoH.	ההודעות אינן מאובטחות עד לשרת הפרוקסי, רק ממנו והלאה ההודעות מאובטחות.	ההודעות אינן באמת שומרות על פרטיות הלקוח שכן ניתן לראות את בקשתו.
ברמת שרת proxy מקומי	הבקשות נשלחות על גבי שרת DNS לשרת ייעודי מקומי שבעצמו שולח את הבקשות על גבי DoH.	ההודעות אינן מאובטחות ברשת המקומית אך כן מאובטחות ברשת החיצונית.	ההודעות אינן מאובטחות וניתן לקרוא אותם ברשת הפנימית אך מאובטחות כלפי הרשת החיצונית
ברמת הגדרות המחשב	המחשב עושה סינון להודעות שנשלחות ולמעשה משנה את הודעות הDNS לDoH כך שהוא עושה תרגום בשליחה וקבלת ההודעות.	ההודעות נשלחות באופן מאובטח כבר מהמחשב עצמו.	כלל ההודעות מאובטחות ופרטיות שכן לא ניתן לקרוא אותם מחוץ למחשב הלקוח.

יתרונות וחסרונות של המימושים השונים:

מימוש	יתרונות	חסרונות
ברמת האפליקציה	גמישות – כל אפליקציה שדורשת פרטיות יכולה לעשות זאת, לעומת זאת בשימושים שונים בהם לא נדרש פרטיות ניתן להשתמש בפרוטוקול הDNS ששכיח יותר ומהיר יותר.	דורש מהאפליקציות לפתח בעצמם את גרסתם, חוסר אחידות ומקשה על הבטחת ביטחון הלקוח שכן זה נתון לבחירת המפתחים. מקשה על פיתוח והטמעה של הDoH שכן כל אפליקציה שמעוניינת להיות מאובטחת צריכה לעשות זאת בעצמם.
ברמת שרת proxy ברשת	יותר שרתים נגישים, מהירות – כאשר שרת הproxy יודע לענות על הבקשה, בהכרח הבקשה/תגובה יהיו מהירים יותר. יתרונות השימוש בproxy תקפים גם כאן.	אבטחה ופרטיות – למעשה זה מיותר להשתמש בDoH שכן ניתן לקרוא את הבקשות ברשת, אינו מוצפן או פרטי, אינו מונע מאחרים לראות את הבקשות או התגובות.
ברמת שרת proxy מקומי	אבטחה ופרטיות – במקומות פרטיים כמו מקומות עבודה בהם מיישמים שרת proxy מקומי ניתן להניח כי הרשת הפנימית בטוחה מספיק ולכן, מאחר והבקשות נשלחות באופן מאובטח לרשת הציבורית ישנה פרטיות ואבטחה. באמצעות מימוש זה ניתן לממש פילטר לכתובות לא רצויות שכן ההודעות אינן פרטיות ברשת הפנימית ולכן ניתן לעשות את הסינון הרצוי.	לא ניתן להטמעה באופן נרחב מספיק כדי ליישם זאת לכולם, כחלק מהחסרונות של DoH יש צורך בשרתים ייעודיים. מהירות – זמני השליחה והקבלה איטיים יותר בהכרח מאשר שימוש בDNS שכן הבקשות נשלחות גם באופן מאובטח וגם עושות עצירה בשרת הproxy.
ברמת הגדרות המחשב	אבטחה ופרטיות – כל ההודעות נשלחות באופן מאובטח אפילו ברשת הפנימית. בנוסף, באמצעות מימוש זה ניתן למעשה לממש סינון בקשות מקומי (כמו שגם באנטי וירוסים בDNS רגיל) לעומת מימוש ברמת האפליקציה.	מהירות – זמני השליחה איטיים יותר מDNS. תלותי בשרתים תומכי DoH (עדיין לא שכיח מספיק).

לפי דעתי, השיטה המועדפת מבין הארבעה היא מימוש בהגדרות המחשב.
למרות שמימוש ברמת שרת proxy מקומי מעניק יתרונות רבים הוא יעיל לחברות ורשתות פרטיות בעלי שימוש בproxy. לעומת זאת מימוש בהגדרות המחשב מאפשר גם שימוש גם בDNS וגם בDoH כך שניתן יהיה להגדיר חריגות במידת הצורך על מנת לשפר את המהירות או שימוש בשרתים שאינם תומכים בDoH על ידי שימוש בשרתים נאמנים (כך נשמר גם על אבטחה – כנגד spoofing וכו').
בנוסף, על ידי מימוש ברמת הגדרות המחשב ניתנת הבחירה ללקוח וגם מתאפשרת הטמעה רחבה יותר של DoH שכן כאשר ההגדרה מופעלת כלל האפליקציות יתמכו בכך. בכך רמת האבטחה והפרטיות של הלקוחות תגדל.
יתרונות אלו גוברים על החסרונות שכן יש במימוש זה גמישות ואופן מרחב פעולה שמאפשר פתרונות רבים לבעיות הנובעות מDoH, כך למשל ניתן לעשות סינון כתובות לא רצויות ברמת אפליקציה או ברמת מע' ההפעלה (כמו אנטי וירוסים) או שירות יעודי במע' ההפעלה שיבצע את הסינון.

5. יתרון בשימוש בפרוטוקול DoH לעומת שימוש בDo53 כאשר ברשת קיים איבוד פקטות הינו שימוש בTCP שכן TCP הוא פרוטוקול שמבוסס על שליחת מידע באופן אמין ככל הניתן. כאשר TCP שולח את ההודעה, הוא שולח אותה בחלקים ממוספרים כך שאם חלק אחד אבד או לא התקבל כראוי, הוא מבקש את החלק הנדרש ומחבר אותו במקומו כך שלמעשה אין צורך לשלוח את הבקשה כולה מחדש כמו בDo53 שמבוסס על UDP. דבר זה חוסך זמן ומאפשר קבלת מידע באופן אמין יותר – כך נוכל לשלוח פחות בקשות, נאבד פחות מידע.

חלק ב' - Congestion Control

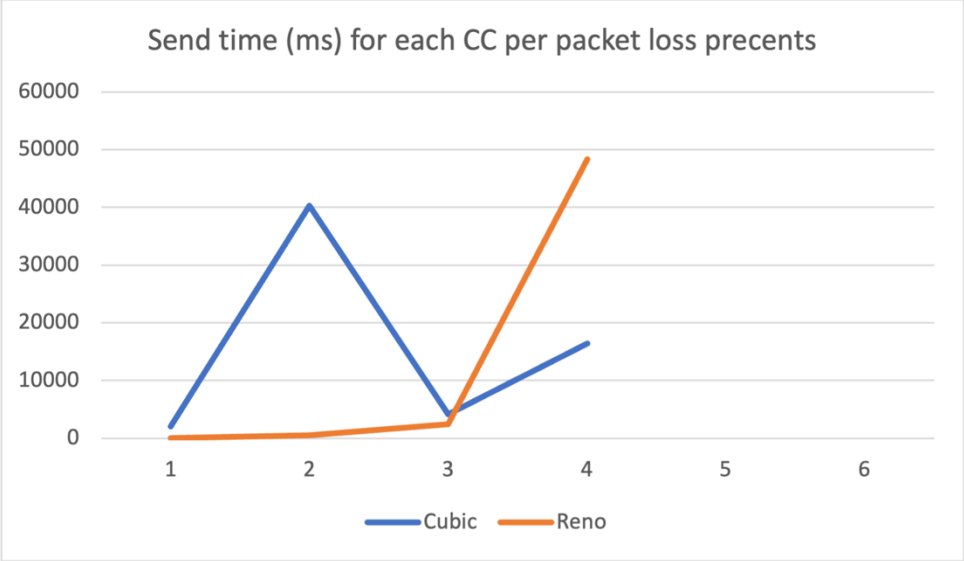
רקע:
האלגוריתמים Cubic ו reno הינם אלגוריתמים לניהול עומס השונים בנקיטת צעדיהם כאשר נדרש שינוי בקצב התעבורה ואופן העברת ההודעות. האלגוריתם cubic למעשה מנסה לשלוח ולקבל הודעות בקצב הגבוה ביותר שמתאפשר. לעומת זאת אלגוריתם הreno מאט את קצב העברת הנתונים בחצי בכל פעם שמגיע לתקרת השליחה.

נוכל לצפות קפיצת זמן משמעותית עבור reno בעוד סוג של קעירות או שאיפה כללית (או איזהשהו ממוצע) עבור cubic.

נראה את התוצאות:
הערה על הגרפים: למעשה המספרים שבציר האופקי מייצגים packet loss באחוזים של 0, 10, 15, 20, 25, 30 בהתאמה (1 עבור 0%, 2 עבור 10%, 3 עבור 15%, 4 עבור 20% וכו').

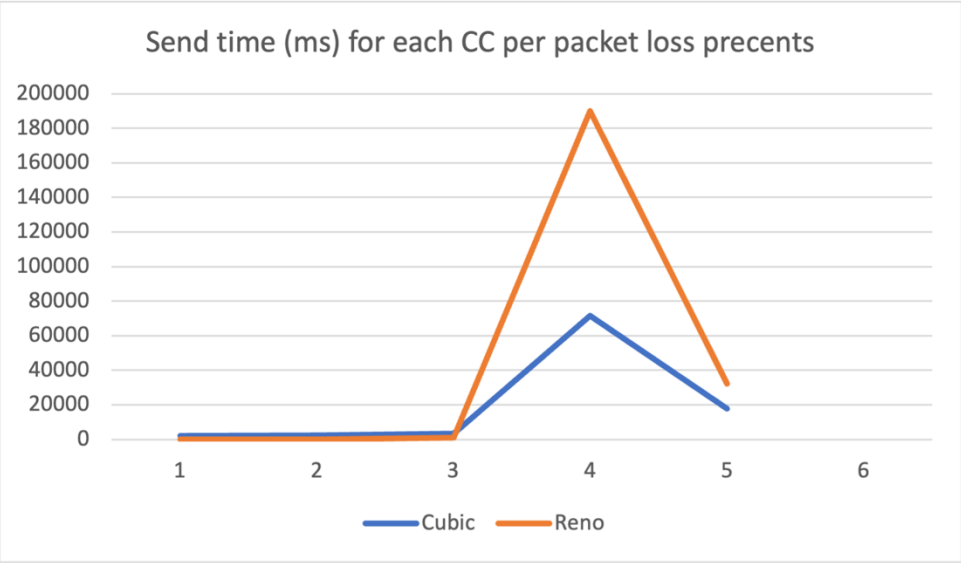
עבור קובץ של 5MB:

Packet Loss % / Algorithm -> time in ms	Cubic	Reno
0%	2005.763534	4.809600
10%	40286.726508	468.255394
15%	4151.111682	2471.512769
20%	16433.182434	48332.870703
25%		
30%		



עבור קובץ של 2MB:
נוכל לומר כי עבור 20 אחוז בreno ניתן לעשות באופן יחסי עבור התוצאה של 5MB ולכן אשים זאת בגרף.

Packet Loss % / Algorithm	Cubic	Reno
0%	2001.77921	0.6262
10%	2299.54576	154.305794
15%	3317.15902	923.058386
20%	71574.3538	
25%	17986.9799	32256.10474
30%		



עשיתי מגוון ניסיונות על גדלים שונים שתוצאתם לא נראתה לי (למעלה מ20 גדלים שונים בטווח של KBים לGBים) שכן הם היו שונים. גם לאחר שחזרתי על אותם גדלים פעמים רבות קיבלתי תוצאות שונות שלא נראו לי שכן היו הבדלים קיצוניים.

לפי דעתי ניתן להבין את תוצאות אלה בכך שהפרשי הזמן (למעשה השיפוע) בreno כאשר יש packet loss קיצוניים משמעותיים מאשר cubic שכן cubic מנסה למעשה לשמר על מהירות גבוהה בעוד שreno פועל בכך שמאט את מהירותו באופן קיצוני יותר כדי שכמות המידע שתאבד תהיה קטנה יותר למעשה.
מאחר וזה גרף תלוי זמן הרי שניתן לשער שבפעמים בהם יהיו קבצים גדולים יותר Cubic יהיה מהיר יותר (השטח מתחת לגרף קטן יותר).