מטלה 3 -רשתות תקשורת

מגישים: 318845500 , 324460682

חלק ג':

בחלק הזה הרצנו את התוכנית של ה TCP 16 פעמים ואת ה RUDP 4 פעמים על חבילה בגודל 2.5MB בצורה הבאה:

חילקנו את זה ל 4 הרצות כל פעם עם loss אחר ובכל loss בTCP הרצנו אלגוריתם שונה ב sender או ב receiver

את ההקלטות PCAP ניתן לראות בגיט ביחד עם צילומי הפלט של כל אחד מההרצות (צילום של ה sender וצילום של ה receiver)

עשינו לפי סעיף הבונוס והרצנו על חבילה גדולה בכל השילובים של האלגוריתמים.

ריכזנו פה בטבלאות את הנתונים שאספנו עבור כל אחד מהמקרים:

ממוצע זמן השליחה של הקובץ בכל אחד מהמקרים (ms):

	TCP				RUDP
Sender:Receiver	reno:reno	reno:cubic	cubic:reno	cubic:cubic	
0 % loss	3.07	3.08	3.42	3.4	23.05
2 % loss	3.75	3.08	5.63	3.41	620.7
5 % loss	51.92	45.6	23.12	4.83	1976.44
10 % loss	303.64	124.78	106.12	19.38	3498.16

מהירות השליחה הממוצע בכל אחד מהמקרים (MB/sec) :

	TCP				RUDP
Sender:Receiver	reno:reno	reno:cubic	cubic:reno	cubic:cubic	
0 % loss	816.93	760.34	808.66	784.91	115.51
2 % loss	713.40	840.34	571.00	741.99	4.47
5 % loss	455.80	495.85	610.56	721.41	1.24
10 % loss	69.58	445.05	457.61	593.84	0.69

*את הנתונים על כל חבילה ניתן לראות בצילומי מסך של הריצות. בכל מצב שלחנו בין 6 ל 10 חבילות על מנת לקבל דאטה אמין.

נענה על השאלות לפי הנתונים:

שאלה 1: ב TCP, איזה אלגוריתם נתן תוצאות יותר טובות?

באופן כללי, כאשר ה loss היה נמוך (0 או 2 אחוז) אז ראינו תוצאות מאוד דומות בין ה reno וה cubic אבל ה reno היה טיפה יותר מהיר.

כאשר ה Loss היה יותר גבוה אנחנו רואים בבירור שה cubic משפר את מהירות השליחה ומצליח לשלוח במהירות יותר גבוהה.

הגענו למסקנה הזאת לפי מדידת הזמן שלקח לכל חבילה להגיע, ולפי כמות הדאטה חישבנו את המהירות שליחה בכל אחד מהאחוזי loss וחישבנו את ממוצע הזמנים והמהירויות והשוונו את הממוצעים אחד לשני על מנת למדוד את הביצועים.

<u>שאלה 2:</u> איך היו הביצועים של ה RUDP ביחס ל TCP הרגיל?

לפי הנתונים של הרצת ה RUDP וה TCP בכל אחד מה loss ראינו שה RUDP משמעותית פחות טוב מה TCP הרגיל. זה מפני שמימשנו את ה RUDP עם פרוטוקול stop and wait והגבלנו את כמות הבייטים שאפשר להעביר בכל חבילה. בנוסף לא מימשנו בקרת זרימה או בקרת גודש.

אפשר לראות שבכל מצב המהירות שליחה של ה RUDP קטנה מהמהירות של ה TCP וגם שהזמן שלוקח להעביר את החבילות בעזרת ה RUDP תמיד גדול משמעותית מכל קומבינציה של TCP.

<u>שאלה 3:</u>

לפי הנתונים שאספנו לא נרצה להשתמש ב RUDP שלנו בשום מצב, כי ראינו שהביצועים שלו פחות טובים בכל המקרים.

אם אנחנו מדברים באופן כללי על RUDP שממומש יותר טוב (עם פרוטוקול יעיל יותר ואלגוריתמים של בקרת זרימה ובקרת גודש יעילים), נעדיף להשתמש בו כאשר יש יותר חשיבות למהירות שבה המידע מגיע. לדוגמה בשידורים חיים, משחקי מחשב מקוונים, שיחות אינטרנט . ב TCP נעדיף להשתמש כאשר אנחנו רוצים שהמידע יגיע בצורה אמינה ללא חשיבות גדולה לכמות הזמן שלוקח. לדוגמה: מייל, מסמכים וכדו'.

<u>שאלות על החומר:</u>

שאלה 1:

מוצע להגדיל את ה SSThreshold בתחילת הקשר. נבדוק מה זה עושה ומתי זה יועיל לנו.
כאשר אנחנו שולחים חבילה ב TCP אנחנו שולחים בהתחלה מספר פאקטות וכל פעם מחקים ל ACK כדי להגדיל את החלון. עם חלון יותר גדול נשלח יותר פאקטות ביחד (בלי לחכות ל ACK על כל אחד מהם) וברגע שקיבלנו עליהם ACK נגדיל את החלון עוד. אנחנו ממשיכים ככה עד שמגיעים ל SSThreshold.

אם אנחנו נגדיל את ה SSThreshold אז אנחנו נהיה יותר זמן במצב ההתחלתי של ה slow start כלומר נמשיך להגדיל את החלון ונשלח כל פעם יותר פאקטות בחלון בלי לחכות לACK. ככה נוכל לשלוח יותר מידע מהרי יותר אבל לא יהיה לנו את הוודאות שזה מגיע .

לפי דעתנו, המקרה שבו הכי עדיף להגדיל את ה SSThreshold הוא מקרה 1.

כלומר כאשר אנחנו במצב של קשר ארוך (TCP עם הרבה מידע) על גבי רשת אמינה עם RTT גדול.

כאשר ל TCP יש הרבה מידע לשלוח, הגדלת ה SSThreshold תעזור לנו לשלוח את המידע הזה בחלון יותר גדול וככה המידע ישלח יותר מהר.

כאשר הרשת אמינה, כלומר מעט מאוד חבילות הולכות לאיבוד, אז החיסרון בהגדלת ה SSThreshold (שאין לנו ודאות על החבילות ששלחנו שהם מגיעים) קטנה. כלומר אם מגדילים את ה SSThreshold, אז עדיף לעשות את זה במקרה שאנחנו יודעים שהרשת אמינה.

כאשר ה RTT גדול, אז לחכות כל פעם ל ACK על חבילות שנשלחים בקבוצות קטנות (בחלון קטן) יקח הרבה מאוד זמן. לכן כדאי להמשיך להגדיל את החלון על ידי הגדלת ה SSThreshold.

שאלה 2:

לפי הנתונים, בקשר TCP אנחנו מתחילים עם slow start.

התחלנו עם חלון בגודל 1MSS וכל פעם שקיבלנו ACK אנחנו מגדילים את החלון פי 2. כלומר כל עוד לא איבדנו חבילות והגענו ל timeout ולא קיבלנו duplicate ACK החלון ימשיך להכפיל את עצמו פי 2 עד שיגיע ל SSthresh

לפי הנתונים, לא היה אובדן חבילות במהלך הקשר וגם החלון של המקבל (rwnd) תמיד היה יותר גדול מה SSthresh כלומר לא שלחנו ל B יותר ממה שהוא מסוגל לקלוט ולכן לא קיבלנו חזרה duplicate ACK.

לכן כל RTT החלון יגדיל את עצמו פי 2 עד שנגיע ל S*MSS, כלומר יהיו $\log_2(s)$ פעולות של שליחת מידע וקבלה חזרה של ACK לכן כמות הזמן יהיה $\log_2(s)RTT$

בנוסף ניתן לחשב את כמות המידע שנשלח כך:

$$1MSS + 2MSS + 4MSS + \cdots + sMSS = \sum_{i=0}^{\log_2(s)} 2^i\, MSS$$

נשים לב ש $\sum_{i=0}^{\log_2(s)} 2^i$ הוא טור הנדסי סופי. ולכן לפי הנוסחה לחישוב טור הנדסי סופי נקבל: $\sum_{i=1}^{\log_2(s)} 2^i = 2\frac{2^{\log_2(s)}-1}{2-1} = 2\frac{s-1}{1} = 2(s-1)$

כלומר כמות המידע שעבר הוא בערך: 2(s-1)MSS

לכן התשובה הנכונה היא א' : בערך $2s\frac{MSS}{\lg S\ RTT}$

שאלה 3:

הספרת ביקורת הכי נמוכה היא 0 לכן נשתמש בספרת ביקורת השנייה **X=2**

נחשב את זמן ההשהייה (propagation delay) כך: מרחק לחלק בקצב ההתפשטות.

נתון שהמרחק הוא 1km כלומר 1000m וקצב ההתפשטות הוא $2 \cdot 10^8$ m/sec

כלומר זמן ההשהייה הוא: $\frac{1000}{2\cdot 10^8} = \frac{10^3}{2\cdot 10^8} = \frac{1}{2} \cdot 10^{-5} = 0.000005$

נחשב את הזמן שלוקח להעביר חבילה אחת: גודל החבילה חלקי קצב התקשורת

נתון שקצב התקשורת הוא 8Gbps ולפי הספרת ביקורת גודל החבילה הוא 2Kbyte

לכן הזמן שלוקח להעביר חבילה הוא: $\frac{2\text{kbyte}}{8\text{Gbps}} = \frac{2\cdot 8\cdot 10^3}{8\cdot 10^9} = \frac{2}{10^6} = 0.000002$

עכשיו אפשר לחשב את ה RTT:

$$\text{RTT} = 2 \cdot (\frac{2}{10^6} + \frac{1}{2} \cdot 10^{-5}) = 2 \cdot (2 \cdot 10^{-6} + 5 \cdot 10^{-6}) = 2 \cdot (7 \cdot 10^{-6}) = 14 \cdot 10^{-6}$$

בגלל שאין איבוד מידע כל הפקטות יגיעו, ואין שידורים חוזרים אז כל ה ACK יגיעו. לכן אנחנו יכולים להגדיל את החלון כדי לנצל את כל רוחב הפס.

לכן נחשב את גודל החלון : קצב התקשורת כפול RTT לחלק בגודל החבילה

$$\text{window size} = \frac{\text{RTT} \cdot \text{קצב תקשורת}}{\text{גודל חבילה}} = \frac{14 \cdot 10^{-6} \cdot 8 \cdot 10^9}{2 \cdot 8 \cdot 10^3} = 7$$