תשובות על השאלות
שאלה 1:

מקרה	Cubic	reno
0	Time – 2432.50 Bandwidth - 33	Time – 2734 Bandwidth - 29
2	Time – 749 Bandwidth - 106	Time – 965 Bandwidth - 82
5	Time – 1854 Bandwidth - 45	Time – 6766.50 Bandwidth - 33
10	Time – 1788 Bandwidth - 44	Time – 6608 Bandwidth - 12

לפי הטבלה ניתן להסיק שCUBIC עובד יותר טוב מRENO בכל המקרים המהירות שבה הוא עובד קטנה יותר.
בRENO ראינו שהנתונים הללו עולים כל פעם שעליונ במקרים השונים.

שאלה 2:

ההשוואה בין היישום שלנו של Reliable UDP (RUDP) לבין TCP הרגיל הראתה תוצאות מפתיעות ומעניינות. בכל התרחישים שבדקנו, כולל אחוזי איבוד פקטות של 0%, 2%, 5%, ו-10%, RUDP הציג ביצועים עדיפים על פני TCP.

RUDP היה מהיר יותר בהעברת הנתונים ולקח לו פחות זמן להשלים את העברת הקבצים בכל התרחישים. זה מפתיע במיוחד בהתחשב בכך שTCP נחשב בדרך כלל לפרוטוקול יעיל ואמין יותר.

לגבי השאלה איזה פרוטוקול טוב יותר עבור איבוד פקטות גבוה, התוצאות שלנו מצביעות על כך ש RUDP תפקד טוב יותר גם במצבים של איבוד פקטות גבוה (עד 10%). זה מפתיע, כי בדרך כלל מצפים שTCP יתפקד טוב יותר במצבים כאלה בזכות מנגנוני בקרת הגודש והזרימה המתקדמים שלו.

ייתכן שהביצועים הטובים יותר של RUDP נובעים מכמה גורמים:

1. פשטות: RUDP עשוי להיות בעל תקורה נמוכה יותר מTCP, מה שמאפשר לו להגיב מהר יותר לאיבוד פקטות.

2. התאמה ספציפית: יתכן שהיישום שלנו של RUDP הותאם במיוחד לתנאי הרשת והתרחישים שבדקנו.

3. גודל חבילות: ייתכן שגודל החבילות שבחרנו ל-RUDP היה אופטימלי לתנאי הרשת שלנו.

4. מנגנון אישור מותאם: ייתכן שמנגנון האישור והשליחה מחדש שיישמנו ב-RUDP היה יעיל במיוחד בתנאים אלו.

חשוב לציין שתוצאות אלו מפתיעות ועשויות להיות ספציפיות לסביבת הבדיקה שלנו. בתנאי רשת אחרים או ביישומים מורכבים יותר, ייתכן שנראה תוצאות שונות. כמו כן, TCP מציע יתרונות נוספים

כמו בקרת גודש מתקדמת וקישוריות מלאה, שלא בהכרח באים לידי ביטוי בבדיקות הפשוטות יחסית שערכנו.

לסיכום, בתנאי הבדיקה שלנו, RUDP הציג ביצועים טובים יותר מTCP, גם במצבי איבוד פקטות גבוהים.

שאלה 3:

מתי עדיף להשתמש ב-TCP:

1. יישומים הדורשים אמינות ושלמות נתונים: TCP מספק העברת נתונים אמינה ומסודרת, תוך וידוא שכל הנתונים מגיעים ליעדם ללא אובדן או שיבושים. זה הופך אותו למתאים ליישומים כגון העברת קבצים, דואר אלקטרוני או גלישה באינטרנט, שבהם חשובה שלמות הנתונים.

2. יישומים הדורשים בקרת זרימה ובקרת עומס: TCP כולל מנגנונים מובנים לבקרת זרימה ובקרת עומס, המסייעים למנוע הצפת הרשת ומבטיחים חלוקה הוגנת של רוחב הפס בין זרמי נתונים מרובים. זה שימושי ביישומים כגון הזרמת מדיה או העברת נתונים בין שרתים, שבהם נדרש ניהול יעיל של רוחב הפס.

3. יישומים הפועלים ברשתות לא אמינות: TCP מטפל בבעיות כגון אובדן חבילות, שיבושים או עיכובים משתנים ברשת. הוא מספק אמינות על ידי שליחה מחדש של חבילות שאבדו ובקרת סדר החבילות. זה הופך אותו לבחירה טובה ליישומים שפועלים ברשתות לא אמינות או בסביבות הרגישות לבעיות רשת.

מתי עדיף להשתמש ב-RUDP:

1. יישומים בזמן אמת או רגישים לעיכובים: RUDP יכול להציע עיכובים נמוכים יותר בהשוואה ל-TCP, מכיוון שהוא אינו משתמש במנגנוני בקרת הזרימה והעומס של TCP. זה הופך אותו למתאים יותר ליישומים כגון משחקים מרובי משתתפים ברשת, שיחות קוליות או וידאו, או יישומי זמן אמת אחרים, שבהם עיכובים נמוכים קריטיים.

2. יישומים עם דרישות מותאמות אישית: RUDP מאפשר למפתחים ליישם אלגוריתמים ומנגנונים מותאמים אישית לבקרת אמינות, זרימה ועומס. זה שימושי ביישומים בעלי דרישות ספציפיות, כגון פרוטוקולי תעבורה מותאמים אישית או פרוטוקולי רשת מיוחדים, שבהם נדרשת בקרה מדויקת יותר על התנהגות הפרוטוקול.

3. יישומים עם רמות שונות של אמינות: RUDP מאפשר ליישומים להגדיר רמות שונות של אמינות עבור זרמי נתונים שונים. לדוגמה, יישום עשוי לדרוש אמינות גבוהה עבור נתוני בקרה קריטיים, אך לסבול איבוד חבילות מסוים עבור נתוני מדיה. RUDP מאפשר למפתחים להתאים אישית את רמת האמינות בהתאם לצורכי היישום.

הבחירה בין TCP ל-RUDP תלויה בדרישות הספציפיות של היישום. TCP מתאים יותר ליישומים הדורשים אמינות מלאה, בקרת זרימה ובקרת עומס, ומסוגלים לסבול עיכובים גבוהים יותר. RUDP, לעומת זאת, מתאים יותר ליישומי זמן אמת או רגישים לעיכובים, עם דרישות מותאמות אישית וצורך ברמות אמינות שונות. בחירת הפרוטוקול המתאים דורשת ניתוח קפדני של דרישות היישום והתפשרויות בין ביצועים ואמינות.

תשובות לחלק האחרון של השאלות:

שאלה 1:

נראה שהגדלת ה-SSThreshold (Slow Start Threshold) בתחילת הקשר ב-TCP עשויה להועיל במידה המירבית במצב מספר 5: "בקשר ארוך על גבי רשת אמינה עם RTT קטן".

הנימוק לבחירה זו:

1. קשר ארוך: מכיוון שיש הרבה מידע לשלוח, הגדלת ה-SSThreshold תאפשר ל-TCP להגיע מהר יותר לקצב שליחה מרבי ולהישאר בו לאורך זמן. זה יאפשר ניצול יעיל יותר של רוחב הפס הזמין.

2. רשת אמינה: בהינתן שהרשת אמינה ומעט מאוד חבילות הולכות לאיבוד, ניתן להניח שרוב החבילות יגיעו ליעדן בהצלחה. לכן, הגדלת ה-SSThreshold לא תגרום לשליחה מחדש מיותרת של חבילות או להאטה משמעותית בקצב השליחה.

3. RTT קטן: זמן ההלוך ושוב (Round Trip Time) הקטן מצביע על השהיה נמוכה ברשת. ככל שה-RTT קטן יותר, כך TCP יכול להגיע מהר יותר לקצב שליחה מרבי. הגדלת ה-SSThreshold תאפשר ל-TCP להימנע מהאטת השליחה שלא לצורך ולנצל ביעילות את רוחב הפס הזמין.

בשאר המצבים, הגדלת ה-SSThreshold עשויה להיות פחות אפקטיבית או אפילו מזיקה:

- בקשר קצר, ההשפעה של הגדלת ה-SSThreshold תהיה מוגבלת, מכיוון שהקשר יסתיים לפני שהשינוי ישפיע באופן משמעותי על ביצועי התקשורת.

- ברשת לא אמינה, הגדלת ה-SSThreshold עלולה לגרום לשליחה מחדש תכופה יותר של חבילות עקב איבוד חבילות, מה שיוביל לירידה בביצועים.

- ב-RTT גדול, ההשפעה של הגדלת ה-SSThreshold תהיה מוגבלת יותר, מכיוון שייקח זמן רב יותר ל-TCP להגיע לקצב שליחה מרבי בגלל ההשהיה הגבוהה ברשת.

לסיכום, הגדלת ה-SSThreshold בתחילת הקשר עשויה להועיל במידה המירבית בקשר ארוך על גבי רשת אמינה עם RTT קטן, מכיוון שהיא תאפשר ניצול יעיל יותר של רוחב הפס הזמין ותאפשר ל-TCP להגיע במהירות לקצב שליחה מרבי, תוך מזעור הסיכון לשליחה מחדש של חבילות או האטה בביצועים.

שאלה 2:

התשובה הנכונה לשאלה זו היא ג':

הסבר:
- הקשר מסתיים כאשר גודל החלון מגיע ל-SSthresh, שהוא S*MSS.
- בתחילת הקשר, גודל החלון הוא MSS ומוכפל בכל RTT במהלך שלב ה-Slow Start.
- סך הבתים שנשלחים במהלך הקשר הוא בערך 2S*MSS.
- משך הזמן הכולל של הקשר הוא בערך $\log_2(S)$*RTT.

- לכן, התפוקה הממוצעת של הקשר היא סך הבתים שנשלחו חלקי משך הזמן, כלומר:
$RTT / (S*MSS) \approx (\log_2(S)*RTT) / (2S*MSS)$

שאלה 3:
ספרת הביקורת (X) היא 1, הפתרון יהיה כדלקמן:

בהינתן:
- קצב התקשורת הוא 8 Gbps
- גודל כל חבילה הוא X * KByte = 1 * KByte = 1,024 bytes
- קצב ההתפשטות הוא 2 * 10^8 m/sec
- המרחק בין התחנות הוא 1 Km

חישוב זמן ההלוך ושוב (RTT):
RTT = (2 * Distance) / Propagation Speed
RTT = (2 * 1,000 m) / (2 * 10^8 m/sec)
RTT = 10^-5 sec = 0.01 ms

כדי להשיג תפוקה מקסימלית, גודל חלון המשלוח (במונחי מספר החבילות) צריך להיות:
Window Size = Bandwidth * RTT / Packet Size

נציב את הערכים:
Window Size = (8 * 10^9 bps) * (10^-5 sec) / (1 * 1024 * 8 bits)
Window Size = 80,000,000 / 8,192
Window Size ≈ 9,766

מסקנה:
כדי להשיג תפוקה מקסימלית, כאשר ספרת הביקורת שלך (X) היא 1, גודל חלון המשלוח צריך להיות בערך 9,766 חבילות.