מבוא לרשתות תקשורת | 67594

הרצאות | פרופ' מיכאל שפירא

כתיבה | נמרוד רק

תשפ"ג סמסטר א'

תוכן העניינים

שבוע $\mathbb{I}$ | מבוא

הרצאה

נעסוק בעקרונות מרכזיים ברשתות תקשורת, ובראשם האינטרנט ואתגרים שיש להתמודד איתם בקשר אליו. האינטרט מאוד מורכב ואי אפשר להחליף אותו, אפשר לכל היותר לבצע בו שינויים שקשה מאוד לעשות והם איטיים.

אפליקציות באינטרנט

דוגמה הרשת משרתת מטרה כלשהי למשתמש כלשהו, לדוגמה סטרימינג של סרט. כיום סטרימינג קורה ע"י בקשה של הלקוח מהשרת של כמה שניות של סרט בכל פעם, באופן אינקרמנטלי. כך, אם הרשת לא יכולה להעביר (בין אם יש צוואר בקבוק אצל המשתמש או בכל מום אחר) מספיק סרט מספיק מהר, נקבל תקיעות ונצטרך להוריד רזולוציה כדי לקבל חוויה יותר טובה.

אם ידוע שיש בעיה ברשת, חלק מהשירותים שמספקים ווידאו בלייב שומרים באפר של כמה שניות מהשידור האמיתי כדי שאם יש תקיעה פתאומית באינטרנט החוויה תראה עדיין אחידה.

סטרימינג היא דוגמה לאפליקציה מעל הרשת - שימוש בפרוטוקולים שבקרוב נלמד. חווית הצפיה שתוארה למעלה נקראת Quality of Experience ואלג' הסטרימינג שהצגנו נקרא Dynamic Streaming over HTTP. העניין של QoE הוא בעייתי מאוד כי כולם מתחרים על אותו רוחב פס ושירותים רבים דורשים latency נמוך (לדוגמה שיחת זום, איכות יכולה להיות נמוכה אבל חייב להגיע מהר) לעומת throughput (כשקיבולת הרבה יותר חשובה מהזמן שלוקח שזה יגיע).

האינטרנט בעצמו הוא די פרוץ והרבה זמן מנסים לשפר את האבטחה של הרשת אבל זה מאוד קשה כי יש הרבה שחקנים והרשת מבוזרת ולהגיע להסכמה זה כמעט בלתי אפשרי.

בקרוב נדבר על מודל 7 השכבות של אבל לעתה נכווץ אותו ל-3 שכבות: החומרה (ראוטרים, מחשבים, סוויצ'ים וכו'), פרוטוקולים (הדרך שבה מדברות החומרות ביניהם, פרוטוקלים כמו TCP, BGP וכו') ואפליקציות (אתרים, אפליקציות וכו'). השכבה העליונה והתחתונה משתנות כל הזמן ומשתפרות בקצב תדיר, אבל השכבה האמצעית נשארת מאחורה כבר מאז שנות ה-90.

מהירות מעבר מידע

הרבה מהמידע באינטרנט עובר במהירות האור (לדוגמה בין יבשות) ולכן לכאורה אפשר לשלוח מידע מירושלים לניו יורק במהירות האור. הבעיה עם זה היא שראשית החומרה עצמה מאפשרת $\frac{2}{3}$ מהירות האור, רוחב הפס יכול להיות מוגבל, מספר תחנות הביניים בין היעדים יכול להיות משמעותי ויכול להיות שנלחם על מקום עם פקטות אחרות, לכן במקום ה-$30ms$ ההיפותטי של מהירות האור, זה לוקח כמעט פי 2.

בזמן 70 המילישניות האלה שנדמה שהן כלום זמן, מעבד ממוצע מספיק לעשות מיליוני סייקלים ולכן מבחינתו האינטרנט הוא סופר איטי. הזמן הזה הוא קריטי כשמדובר במניות, או דברים שדורשים תיאום בין אנשים שונים, וזו בעיה כי האינטרנט בפועל מיתרגם לתקשורת א-סינכרונית.

האינטרנט ועניינים נוספים

הרשת מחלוקת לשלושה משתתפים - קודקודים (ראוטרים, סוויצ'ים), קשרים (סיב אופטי וכו') ומשתמשי קצה (מחשב, טלפון אבל גם מקרר, טלוויזיה וכו').

כיום מספר המשתמשים ברשת גדל אקספוננציאלית, בגלל שלכל בן אדם יש הרבה יותר ממחשב אחד או טלפון אחד, אלא מקרר ומיקרו וטלוויזיה וכו'. יש הרבה סוגים שונים של משתמשי קצה, המון סוגים של קודקודים וקישורים, וגיוון גדול מאוד בדרישות אפליקציות (האם צריך דו-כיווניות, האם צריך מהירות, האם צריך אמינות וכו' וכו').

הרשת היא התשתית להעברת מידע בין משתמשים, ורק בה נעסוק. מעליה אפשר לבנות רשתות מבוזרות שונות שהן ברמת האפליקציה וזה א מה שיעניין אותנו.

רשת הטלפוניה

כדי להבין איך האינטרנט עובד, נבין קודם איך רשת הטלפוניה עבדה.

כשרוצים לדבר עם מישהו, קודם כל נרצה לדעת האם אפשר להגיע אליו באמצעות circuit, מסלול שהוא רק שלנו ושאי אפשר לעצור אותו באמצע. אחרי שיש קו, נעביר מידע על הקו עד שתסתיים השיחה. בסוף השיחה, נפנה את המשאבים.

המרכזן הוא זה שיוצר את הקו ומבצע switching - ניתוב של תקשורת שמגיעה מכניסה מסוימת ליציאה מסוימת. כיום טלפוניה מנותבת אוטומטית באמצעות אלג' מורכבים.

כיצד נוכל לנתב שתי כניסות לאותה היציאה?

- Time-Division - נחלק את הזמן לשברירים מחזוריים ונבצע round robbin על רוחב הפס.
- Frequency-Division - נחלק את הפס לתדרים שונים, כאשר כל תדר אחראי על משתמש אחד.

למה האינטרנט לא נראה כמו הטלפוניה

מה קורה אם אם משתמש לא משתמש בסלוט שלו (שותק בשיחה)? בטלפונים זה עבד בסדר, אבל באינטרנט זה לא פרקטי.

הרשת שתיארנו זה עתה היא מאוד יציבה, פשוטה, מהירה (העיכוב הוא רק המרחק הפיזי על הפס), צפויה, קל לגלות איפה יש בעיה.

אז למה לא להשתמש בה גם באינטרנט?

- אי-עמידות בפני כישלון: אם אין רוחב פס או אם יש בעיה באמצע, נוותר ולא נקבל שום תקשורת.
- בזבוז רוחב פס: טלפוניה לא שורדת תחת עומס כי רוחב הפס נגמר מתישהו ובאינטרנט זה לא מתקבל על הדעת. בטלפניה נשמור את רוחב הפס המקסימלי הנדרש לקוים הפתוחים עליה (P), לעומת האינטרנט שבו התקשורת המקסימלית אולי גבוהה אבל ממוצע הפקטות שמועבר (A) משמעותית יותר נמוך ולכן לא צריך לשמור את כל רוחב הפס המקסימלי כל הזמן.

  בטלפניה היחס $\frac{P}{A}$ הוא $1:3$. באינטרנט היחס הוא יותר מ-$1:100$.
- עיצוב מותאם אפליקציה: טלפוניה עוצבה לטלפונים, בניגוד לאינטרנט שנועד לאפליקציות מסוג אחר.
- זמן הכנה: בטלפוניה לוקח זמן לא זניח ליצור את הקו, בעוד באינטרנט צריך שזה יהיה (כמעט) מידי.

כדי להתגבר על כל הבעיות האלה, ב-64' הוצע השימוש בפקטות במקום קו שמור, ביזור הרשת והעברת הפקטות דרך הקודקודים ברשת ובהמשך גם ניתוחים סטיסטיים של מערכות כאלה.

רשתות ברודקאסט

ברשת ברודקאסט כל המידע מועברה לכל קודקוד וכולם חייבים להקשיב לפקטות. זה קורה ב-WiFi ולרוב בשאר LAN-ים (Local Area Networks), ומבחינה אינטואיטיבית גם בהרצאה.

יש כמה בעיות בשיטה זו, ביניהן:

- קשה להעביר את הפקטות מרחקים גאוגרפיים גדולים;
- קשה לתאם גישה של כמה משתמשים לרשת (בהרצאה רק אחד מדבר, אבל מה אם שני סטודנטים רוצים לדבר באותו הזמן?), זו בעיית ה-Multiple Access Problem.
- אין פרטיות בתקשורת (למרות שאפשר להצפין את המידע).

שבוע III | האינטרנט ומודל השכבות

הרצאה

רשתות switched

בניגוד לרשתות ברודקאסט, רשתות שהן switched מאפשרות לרבים לחלוק את אותו המשאב ולאנשים שונים לדבר עם אנשים אחרים באותו הזמן. מתחת לסיווג הזה יש עוד שני סוגי רשתות:

- circuit-switched - נקצה משאבים לקיבולת מקסימלית לאורך כל השיחה.
- packet-switched - נעביר פקטות בהתאם לכמה מידע נרצה להעביר ולא נקצה מראש את כל המשאבים שנדרשים.

פקטות

ברגע שלפקטה לא מוקצים משאבים, מסלול וכו', היא חייבת להכיל עליה את המידע שמספר לרשת מאיפה ולאן היא צריכה להגיע - כי אף אחד אחר לא יעשה את זה בשבילה.

הפקטה מכילה את גוף הפקטה (payload), שהוא המידע שאנחנו רוצים להעביר, ו-header שמכיל מידע שנועד רק לרשת (כמו פרטים על מעטפה).

כל פקטה מנתובת באופן עצמי ולכן היא יכולה לעבור מסלול שונה גם אם יצאה מיד אחרי פקטה אחרת. נצטרך לדאוג שהעובדה הזו לא תהפוך תקשורת לבלתי אפשרית ואיכותית.

כשפקטה יוצאת, היא עוברת דרך כמה תחנות ביניים וכל תחנת ביניים לכאורה מתעלמת מהגוף (אלא במקרים של אבטחה שלא נעסוק בהם), ומעבירה הלאה את הפקטה לתחנה הבאה. לקודקוד יש שיקול דעת לאן לנתב או לא לנתב פקטה (ואז היא תיפול). הרעיון הזה של שליחי ביניים נקרא store-and-forward - הקודקוד שומר את המידע וכשיכול (או לא), מעביר את המידע הלאה.

הערה בכל תחנת ביניים נוסף עוד דיליי של זמן עיבוד בקודקוד (וזה יכול להצטבר במרחקים ארוכים או קודקודים חלשים).

פרצים

מהתיאור הנ"ל, פקטות הן פתרון לא מוצלח ולא יציב בלי הבטחה של ביצועים כלשהם. אם כן, למה זה עדיין חשוב לנו?

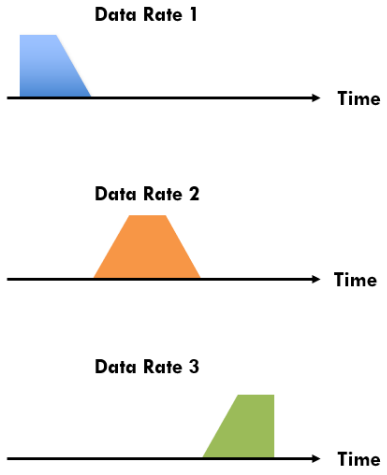
תקשורת באינטרנט הרבה פעמים היא בפרצים, כלומר דממה להרבה זמן ואז הרבה מידע יותר מרוחב הפס האפשרי, לדוגמה חיפוש בגוגל - נחפש, נסתכל על התוצאות, ואז שוב נחפש - וקבלת המידע מגוגל היא הפרץ. לעומת זאת, סטרימינג מגיע באופן יותר רציף מעל האינטרנט.

כיצד נטפל בפרצים? פתרון אחד הוא פשוט להפיל כל פקטה שאנחנו לא יכולים להעביר מיד. לחלופין, נוכל להוסיף באפרים ששומרים פקטות שמחכות לעבור. עם זאת, תמיד יהיו לנו יותר פקטות מאשר מקום בבאפר, ולכן תמיד נצטרך להפיל חלק מהפקטות.

דוגמה נניח שיש לנו שלושה פרצי מידע כבאיור (כחלק משיחה שלוקחת את כל ציר הזמן), כאשר במקסימום כל שיחה דורשת קצת יותר משליש מרוחב הפס.

אם היינו בטלפוניה, היינו מקצים לראשון ולשני מקום והשלישי היה נזרק בחוץ, והיינו מבזבזים בכל רגע נתון שני שליש מרוחב הפס.

עם זאת, היו מספקים משאבים לכולם כי הם לא רצו לדבר במקביל. ההנחה כאן היא שעומס התקשורת בא והולך אבל מתפלג באיזו שהיא צורה לא מתאומת שמאפשרת מעבר של רוב התקשורת בעומס סביר.



איור 1 : דוגמה לתקשורת עם פרצים

Statistical Multiplexing

Statistical Multiplexing הוא הרעיון שלפיו נוכל לשרת משתמשים שונים על אותה התשתית גם אם רשמית אין לכולם מקום ביחד (לא כולם יכולים לחפש באותה השנייה בגוגל משהו, אבל בגלל שזה קורה בזמנים שונים במקומות שונים אנחנו כן מצליחים). גם ביטוח והלוואות עובדים ככה - הבנק לווה כסף בידיעה שלא כולם הולכים למשוך את הכסף שלהם ולכן מרשה לעצמו "להמציא" כסף, או בביטוח, שבו אם כל אחד היה בתאונת דרכים הביטוח היה בבעיה, אבל סטטיסטית זה לא סביר.

אפשר להסתכל על Stat Mux גם בתור חלוקה של הזמן לפריימים (שמחולקים אף הם לסלוטים). בכל פריים נייצר לכל היותר P פקטות ובממוצע נייצר A פקטות. אם P גדול מ-A משמעותית זה לא חכם להקצות משאבים לשיחה ספציפית כמו שהזכרנו כבר הרבה פעמים.

עם זאת, חוק המספרים הגדולים אומר לנו שסביר שערכים מהתפלגות יהיו קרובים לתוחלת, יותר מאשר לערך הכי גדול שאפשר - נקבל הרבה יותר התנהגות דומה ל-A פקטות לפריים מאשר P פקטות לפריים.

השוואה בין רשתות מנותבות Circuit ל-Packet

לרשת circuit-switched יש כמה יתרונות וחסרונות, ביניהם

יתרונות	חסרונות
רוחב פס מובטח	רוחב פס מבוזבז בפרצים
אבסטרקציה נוחה	חיבורים חסומים כשאין משאבים
מסלול העברת מידע פשוט	קודקודים חייבים להיות מודעים לשיחות שעוברות דרכם
תקורה נמוכה פר-פקטה	דיליי בהתחלת השיחה

החסרונות האלה הם קריטיים מדי לשירותים שלנו באינטרנט שבגללה אנחנו לא משתמשים בטלפוניה לאינטרנט.

קריטריון	packet	circuit
אמינות	אם יש כשל הרשת תחשב מסלול מחדש	אם הקו נופל לא נשחזר אותו
יעילות	ניצול של Stat. Mux לרוחב פס יעיל יותר	רוחב פס מוגבל ע"י הקוים שהוקצו להם משאבים
קלות מימוש	רשתות יכולות להתחבר בקלות כי הן לא מקצות אחת לשנייה משאבים	חיבור שתי מרכזיות דורש הקצאת משאבים מכל צד
טיפול בעומסים	צריך להתמודד עם עומסים (לזרוק פקטות, להוריד את הקצב,...)	לא רלוונטי

מודולריות

לחלק את הקוד לאבסטרקציות ומודולריות זה חשוב מכל מיני סיבות. בין היתר, אפשר לשנות את המימוש למימוש שונה/יותר יעיל בלי שאחרים ידעו. בנוסף, אפשר להוסיף עוד פונקציונאליות לקוד עם מודולים חדשים נפרדים, וזה בעיקר מאפשר להמשיך לשדרג ולהתקדם בחזיתות שונות באותו הזמן.

ברשתות מודולריות מקבלת נדבך נוסף - המימוש הוא מבוזר בין הרבה מכונות. לכן הושמו כמה עקרונות כדי לפתור בעיות כאלה.

- שכבות היררכיות - כל שכבה מדברת רק עם השכבה מעליה ומתחתיה ולא מודעת/מתעניינת באחרות.
- עקרון קצה לקצה - הרשת מאפשר חיבוריות וזהו, כל דבר אחר (כמו הצפנה) יעשה בידי המשתמשים עצמם. בנוסף, היא לא מבטיחה שפקטה תגיע, או שתגיע בסדר הנכון ולכן אפליקציות צריכות לדאוג גם לזה.

העקרון הזה נועד כדי לאפשר אפליקציות שאנחנו עוד לא יודעים שיהיו להן דרישות אחרות לעשות מה שהן רוצות בצורה המיטבית ביותר.

- גורל משותף - אם החיבור נופל, כולם שוכחים ממנו ולא נדע שהוא היה קיים לפניכן כי אנחנו לא שומרים מידע כזה, רק בנוגע למה שקורה עכשיו.

הערה העקרון השני והשלישי מופרים ע"י ספקי אינטרנט באמצעות Firewalls וכל מיני מנגנונים אחרים כמו בדיקה של מעבר פקטות בלינקים חלשים מאוד (כמו סלולר). עם זאת, העקרונות האלה הם כן מנחים, רק לא ממומשים במציאות.

נבנה את המשימות שלנו מלמטה למעלה.

- העברת אלקטורנים על כבל.
- העברת ביטים על כבל.
- העברת פקטות על כבל.
- העברת פקטות על רשת מקומית (LAN) עם כתובות מקומיות באמצעות ברודקאסט.
- העברת פקטות בין רשתות מקומיות שונות עם כותבות גלובליות.
- להבטיח שהפקטות מגיעות ליעד.
- לעשות משהו עם המידע הזה.

יש פה רבה משימות שונות ולשם כך אנחנו צריכים מודולים שונים - שכבות - שיהיו אחראים על כל משימה בנפרד.

דוגמה גם במציאות יש איזושהי שכבתיות. מנכ"ל א' שולח למנכ"ל ב' מכתב. המזכירה לקחה את המכתב, שמה אותו במעטפה, שלחה לחברת שליחויות.

זו שמה בעוד מעטפה למעבר פנימי בתוך תשתיות החברה, מעבירה למזכירה של המנכ"ל השני כשהיא הורידה את המעטפה הנוספת, המזכירה מוציאה מהמעטפה ונותנת את המכתב למנכ"ל.

סה"כ המנכ"ל שלח מידע והמנכ"ל האחר קיבל מידע. כל מה שנוסף הוא מטא-דאטא והמסלול שנוצר כאן הוא אנלוגי לחלוטין למה שקורה לפקטות באינטרנט.

מעבר לכך, כל שכבה תיקשרה באמצעות שפה ייחודית לה (מכתב, מעטפה ומעטפה פנימית) וזה אנלוגי לפרוטוקולים השונים של השכבות. אף שכבה לא צריכה להבין את השפה של אף שכבה אחרת.

שכבה	שפה	סוג מידע רלוונטי
מנכ"ל	מכתב	תוכן טקסטואלי
מזכירה	מעטפה	זהות הנמען
חברת שליחויות	מעטפה פנימית	מיקום הנמען

שבע המשימות שתיארנו למעלה מתאימות לחמש שכבות במודל השכבות באינטרנט.

- Pysical Layer - העברת ביטים (ואלקטרונים) בכבל.
- Link Layer - העברת פקטות ברשת מקומית (ובכבל).
- Network Layer - העברת פקטות בין רשתות מקומיות.
- Transport Layer - ווידוא הגעת הפקטות.
- Application Layer - לעשות משהו עם המידע.