מבוא לרשתות תקשורת

מרצה

פרופ' מיכאל שפירא

מתרגל | שלמה פרלס

דניאל דייצ'ב

דויד קיסר שמידט

deychev.com

מבוא לרשתות תקשורת | 67594

נכתב ע"י דויד קיסר-שמידט ודניאל דייצ'ב

15 בפברואר 2023

מרצה | פרופ' מיכאל שפירא

מצאתם שגיאה? ספרו לנו! שלחו לנו מייל לאחת מהכתובות שכאן:

daniel.deychev@mail.huji.ac.il
david.keisarschm@mail.huji.ac.il

תוכן העניינים

רשימת אלגוריתמים

רשימת אלגוריתמים

חלק I

הרצאות

1 מבוא

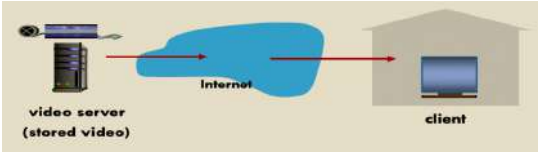
הרצאה 1

בקורס יש לנו כמה מטרות. הראשונה היא לדבר על מהי רשת תקשורת - אתגרים אלגוריתמיים, הנדסיים, אילו עקרונות מנחים אותנו וכו'. השנייה היא להבין בפרט איך האינטרנט עובד. נציין שלמרות שהאינטרנט עובד וכנראה לא היינו שורדים בלעדיו (הרי איך אפשר בלי טיקטוק), עדיין יש בו בעיות: בעיות אבטחה וחולשות, פרוטוקולים בעייתים, ביצועים, הרשת שברירית ובעלת המון חורים וכו'. כל זאת נראה במהלך הקורס. מלבד זאת, כבר אי אפשר להחליף את האינטרנט! לשנות את האינטרנט היום יהיה תופעה גלובאלית, שתשפיע על המון אנשים וזה פשוט לא פרקטי. האינטרנט הוא מערכת מבוזרת שמדברת בשפה, וקשה להחליף שפה בלי הסכמה מכולם, וקשה להשיג הסכמה מכולם.

עובדה מעניינת: בארה"ב (ובאירופה), לא משנה איזו חבילת גלישה יש לנו מהספק (עשר מגה, מאה מגה, סיבים עד הבית), בממוצע אנשים רואים אמזון ונטפליקס ב-HD פחות מ-40% מהזמן.

למרות שכדי לראות סרט ב-HD צריך מהירות גלישה של 10MB/s, ואנו מקבלים מהספק הרבה יותר, אנחנו לא יכולים לצפות רוב הזמן ב-HD כי הבעיה היא בכלל לא במהירות הגלישה מהספק (אז מה כן הבעיה? נראה בהמשך). כשאנחנו צופים בסרט בנטפליקס לדוגמה, כל כמה שניות המחשב שלנו מבקש את הכמה שניות הבאות של הסרט. ככל שנבקש רזולוציה יותר גבוהה, גודל הקובץ שמכיל את אותן כמה שניות יהיה יותר גדול, ואז יקח לו יותר זמן להגיע. אם הסרט נתקע כל כמה זמן, זה אומר שהשניות הבאות לא מגיעות מהר מספיק, וכדאי להוריד רזולוציה.

שאלה דומה היא למה קורה שהשכנים שלנו רואים לפנינו שהיה גול במשחק כדורגל? זה כי כשאנחנו צופים בשידור חי, אנחנו לא רואים אותו באמת בשידור חי - קודם הדפדפן שלנו צובר כמה שניות של השידור, כרשת בטחון, ושומר אותו באיזשהו באפר, ורק אז כשמצטברות מספיק שניות אנו רואים את התוצאה. עכשיו, אם אצל השכנים שלנו רשת הבטחון הזו היא 5 שניות ואצלנו היא 10, אנו נראה את התוצאה חמש שניות אחרי השכנים. מנגנון זה קורה כי הדפדפן לא סומך על המערב הפרוע הקרוי "אינטרנט", שיעביר את המידע באופן רציף ואמין כך שנוכל לצפות בשידור חי באמת ובלי תקיעות.



איור 1: לא משנה כמה השירות חכם, הוא צריך להתמודד עם מערב פרוע בשם האינטרנט באמצע

העניין הוא שישויות שונות באינטרנט רוצות דברים שונים: אם אני בזום לא אכפת לי אם איכות הוידאו נמוכה, אך חשוב שלא יהיה דיליי, ואם אני צופה בסרט אז לא אכפת לי שיהיה דיליי של כמה שניות עד שהסרט יתחיל אם בתמורה איכות הוידאו גבוהה. לפעמים אני רוצה **קיבולת** גבוהה ולפעמים אנחנו רוצים מסלול **מהיר**. הרשת עצמה, שמשרתת המון גורמים, לא יודעת מה כל אחד רוצה, וגם אם הייתה יודעת אין לה את הכלים להתאים את עצמה לכל שירות ושירות. דבר זה משפיע, על מה שהמשתמש חווה - Quality of Experience. הרשת היא כנראה הדבר שהכי משפיע על ה-QoE.

הרשת צריכה לטפל בבקשות של המון ישויות, כל אחת עם מטרה משלה ושרוצה כמה שיותר משאבים לעצמה, והמון פעמים תנאי הרשת עצמם הם מאתגרים, לדוגמה טלפון שצופה בסרטון אבל תוך כדי נוסע באוטובוס ומשנה מקום.

עובדה מעניינת: האינטרנט כה שברירי, שאם ממש היינו רוצים והיה לנו חלום להיכנס לכלא, היינו יכולים לנתב את כל התעבורה של גוגל העולמית לאוניברסיטה העברית - זה קל!

את הבעיה הזו מנסים לתקן המון זמן, ומדהים לחשוב ששום דבר לא מונע מאדם להשתלט על תעבורה של מישהו אחר, וכל מה שצריך זה את הנתב הנכון שמדבר בפרוטוקול הנכון.

קשיים ברשתות תקשורת

ברשתות יש לנו "מודל 7 השכבות". נסתכל על שלוש שכבות. שכבת האפליקציה, שם רצות אפליקציות שרוצות להעביר מידע דרך הרשת. מתחתיה, שכבת הרשת, שם מחליטים על הלוגיקה של העברת המידע - ניתוב, קיבולת, זיהוי ועוד. מתחתיהן שכבה פיזית, שהיא הטכנולוגיה שמבצעת את העברת המידע. מה הטרגדיה ברשתות? יש חדשנות באפליקציות, וגם בטכנולוגיה. אבל - הלוגיקה של הרשת, לא השתנתה בגדול כבר שלושים שנה! יחד עם זאת, באופק כבר רואים התחדשות, ונעשו שינויים במבנה הרשת.

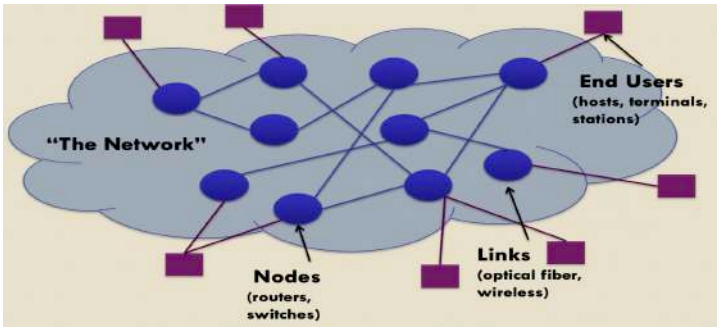
מאחורי בניית רשת יש הרבה פיסיקה. למשל, כמה זמן ייקח להעביר מידע מירושלים לתל אביב? כדי לעשות זאת, נבדוק כמה זמן ייקח להגיע לאור מירושלים לתל אביב. אם מהירות האור היא c והמרחק הוא d נקבל cd, שזה בהצבת הנתונים, 30ms בקירוב. זה חסם תחתון על הביצועים שלנו, היות שבפועל, כשנעביר מידע, המהירות תלויה בתכונות הפיסיקליות של הרכיבים. למשל סיב אופטי לא מאפשר העברה ב-c אלא ב-$\frac{2}{3}c$. מעבר לכך, המידע עלול להמתין בתוך רכיבים אחרים כמו נתבים, מה שמאט את התהליך. למעשה, אם נבצע ניסוי זה, נקבל פקטור 2 מהחסם התחתון, שזה סביר. נבחין כי ייתכן שנצטרך לשלוח את המידע כמה פעמים, היות שהרשת תזרוק אותו עקב עומס.

עולה השאלה, מה תפקידו של המעבד בסיפור הזה. נשאל, כמה Cycles מבצע מחשב PC לפני שהוא מקבל תשובה לשאלה שהוא שלח לשרת בניו יורק? זה תלוי כמובן בביצועי המעבד, אבל אם נניח שהוא רץ בקצב של 3GHz, ושהמרחק הוא $140\,\text{msec}$ (לפי מהירות האור), נקבל $\frac{3\cdot 10^9\text{cycles}}{0.14\text{sec}} = 42\cdot 10^7\text{ cycles}$. מבחינת המעבד, זה נצח, והוא לא יעצור עד שהתשובה תגיע, אלא ימשיך בפעולותו. היות שקשה לסנכרן מערכות תקשורת, הן תמיד אסינכרוניות, והתשובה שאנחנו מקבלים נכונה לרגע שהצד השני שלח אותה, ולא בהכרח לרגע שהיא הגיעה אליכם. מעבר לכך, יש לכך השלכה חישובית - אלגוריתם יעיל שדורש תקשורת מרובה יותר, עלול להיות פחות טוב מאלגוריתם פחות יעיל שדורש פחות תקשורת. מפתה לשאול, מה יקרה אם נמקם מחשבים אחד ליד השני בכבל. התשובה היא שזה ישפר ביצועים, אבל עדיין - המעבד יבצע לפחות $6\cdot 10^5\text{ cycles}$ בכל פעם - זה הרבה מאוד. **לא ניתן להזניח את התקורה מתקשורת!**

> **מערכת מבוזרת**: מערכת של רכיבים אסינכרוניים נקראת "מערכת מבוזרת", ועולה השאלה במה היא שונה מהרשת? חשוב להדגיש, הרשת נמצאת מתחת למערכת, ומאפשרת לה להעביר מידע בין משתמשים - היא לא מייצרת מידע, מי שמייצר אותו הוא המערכת. מהרגע שיש רשת, אפשר יהיה לבנות מערכת.

הרשת מורכבת מכמה רכיבים:

- משתמשי קצה (End Users) - כל רכיב שיכול לתקשר: המחשב שלנו, מקרר, שעון חכם, טלוויזיה וכו'. לתקשורת משתמשים שאינם המחשב שלנו (כלומר שאין מאחוריהם בן אדם), אנו קוראים $\text{IoT} - \text{Internet of Things}$, והיא גדלה משמעותית.
- לינקים (Links) - מה שמחבר פיזית ישויות שונות ברשת: סיבים, כבל קלאסי וכו'.
- צמתים (Nodes) - רכיבים ברשת שכל יעודם להעביר את המידע של משתמשי הקצה למקום הדרוש. לדוגמה, נתבים, Switches.

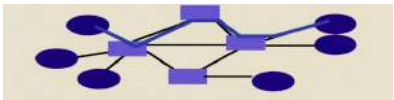


איור 2: רכיבים ברשת. ה-Links מאפשרים ל-Nodes לתקשר אחד עם השני, והם מאפשרים למשתמשי הקצה לתקשר.

רקע היסטורי - רשת הטלפוניה

הרשת כיום שונה מאוד מנקודת הפתיחה שלה - רשת הטלפונים.

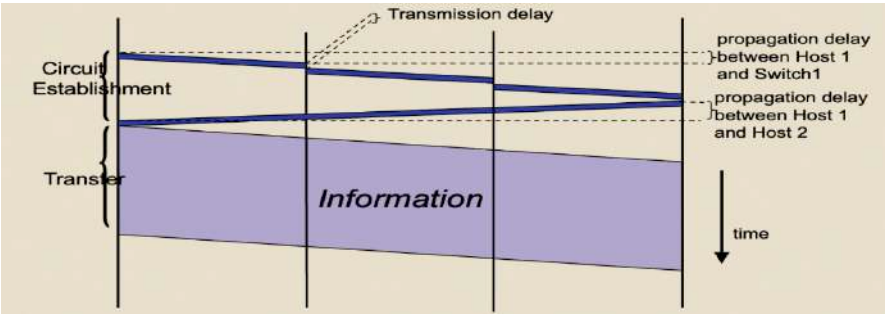
ברשת זו, על מנת לתקשר, צריך לבדוק האם יש אפשרות לעשות זאת. לזה אנו קוראים Circuit - מסלול של Nodes שמקשר שתי ישויות שרוצות לתקשר, ויש שם קיבולת מוקצית עבורו. הרכיבים במסלול ישמרו מידע על החיבור, ואם הוא לא זמין, ישלח "סיגנל עסוק". חלוקת המשאבים כאן שונה מהרשת שאנחנו מכירים - קיבלנו קיבולת שנשמרת עד סיום החיבור, ואם אין כזו, אז אין חיבור - זה בינארי:



איור 3: רשת הטלפונים - כדי להעביר מידע, אין צורך לספק כתובת יעד, היות שהרכיבים במסלול יודעים את היעד, ומעבירים את המידע בצורה רציפה.

כשהסתיימה השיחה, יש פעולה שנקראית Tear Down שמשחררת את המשאבים.

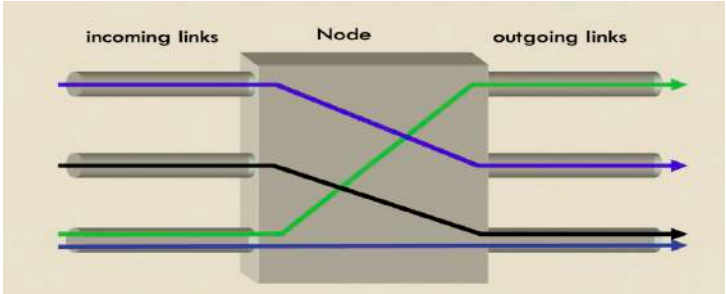
בפועל, כשאנחנו מעבירים מידע, התהליך נראה כך מבחינת זמנים:



איור 4: ציר ה-x הוא ציר המרחק, וציר ה-y הוא זמן. ניתן לראות שהעברת המידע לוקחת זמן.

Multiplexing/Demultiplexing

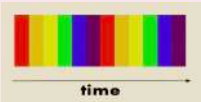
כדי להעביר מידע, הרשת משתמשת ברכיב הנקרא Switch:



איור 5: ערוץ קלט מועבר לערוץ פלט, כך שיתכנו קלטים שונים באותו ערוץ פלט

עולה השאלה כיצד הוא מסוגל להעביר מידע שונה באותו ערוץ פלט? יש כמה דרכים:

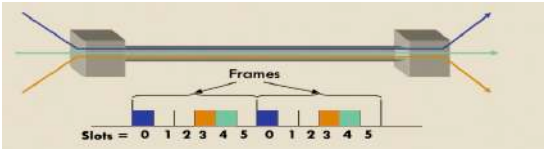
פתרונות אפשריים



איור 6: חלוקת הקיבולת בין ערוצים באופן מחזורי

איור 7: העברה של תדרים שונים דרך אותו ערוץ פלט

לשיטה הראשונה יש חסרון. במידה שאין עומס על הרשת, אנחנו עלולים לבזבז קיבולת:



איור 8: סלוט מספר 1 לא התמלא, בזבוז

יחד עם זאת, רשת הטלפוניה נחשבת לרשת נפלאה. הבעיה היא שרשת האינטרנט לא יכולה לעבוד כך. היא מאוד פשוטה, יש מסלול אחד להעברת המידע, ואם משהו השתבש, אפשר לבדוק את המסלול - באינטרנט זה לא ככה. החסרון הוא שבמצב עומס, לא יתאפשרו שיחות. נמנה חולשות נוספות:

1. אין עמידות בפני תקלות - אם יש תקלה, אין מסלול זמין.
2. ניצול לא טוב של הרשת - אם יש קיבולת P בשיא וקיבולת A בממוצע, צריך לתמוך ב-P, אך אם הפער בין P ל-A גדול ($\frac{A}{P}$ קטן), אנחנו בבעיה. ברשת הטלפוניה היחס הוא בערך $\frac{1}{3}$. באפליקציות באינטרנט, הפער מאוד מאוד גדול, ולא מודל הטלפוניה לא מתאים.
3. רשת הטלפוניה מיועדת לטלפונים. היא מוגבלת לאפליקציה הזו, ולא לדברים אחרים. רשת האינטרנט תומכת בכל האפליקציות.

רשתות תקשורת כיום

ב-1964 הציעו מודל חדש: המערכת תהיה מבוזרת, והמידע שתשלח יהיה בלוקים של הודעות (פקטות), שיואחסנו ויועברו ברשת - זה בסיס האינטרנט כיום.

נוכל לסווג רשתות תקשורת לפי הדרך בה הם מעבירים מידע בין Nodes.

רשת Broadcast

הרשת הפשוטה ביותר, היא רשת Broadcast, אחת שמעבירה מידע, שמגיע ישירות לכל הרכיבים ברשת, כמו באולם ההרצאה - המרצה מדבר, והקהל שומע, אין כאן תמיכה בתקשורת פרטית. איפה זה קורה באינטרנט? זה קורה בעיקר כשהמרחק הגאוגרפי מצומצם, למשל רשת WIFI. אנו קוראים לרשת כזו - LAN – Local Area Network. כפי שציינו, צריך לנהל כאן תקשורת במקביל - Multiple Access Problem. עולה השאלה כמה אפשר למתוח את הרשת הזו, הטווח של התקשורת מוגבל. לבסוף, צריך לתמוך בפרטיות - אפשר לעשות זאת רק צריך הצפנה.

רשת Switch

אל מול רשת ה-Broadcast, קיימת רשת ה-Switch, שהיא רשת שמסוגלת להעביר מידע ליעד, מבלי לשלוח אותו לכל שאר הרכיבים ברשת. לרשת זו יש שני סוגים, אחת מבוססת הקצאת משאבים מראש כמו ברשת הטלפוניה, והשנייה מבוססת פקטות. בפועל, הרשת שלנו מבוססת פקטות. הרצאה 2

> **!** האינטרנט מכיל כמה מושגים חשובים: Net רשת, inter בין, intra בתוך. כלומר, אינטרנט היא רשת, שבין רשתות. מה המשמעות של זה? גם לפני האינטרנט היו רשתות, אבל האינטרנט מאפשר קיום של רשתות פנימיות, כך שכל רכיב ברשת יכול להיות רשת בפני עצמו.

מטרתנו היא לבנות את הרשתות מלמטה למעלה, מהרשת הקטנה ביותר עד הרשת הגדולה ביותר.

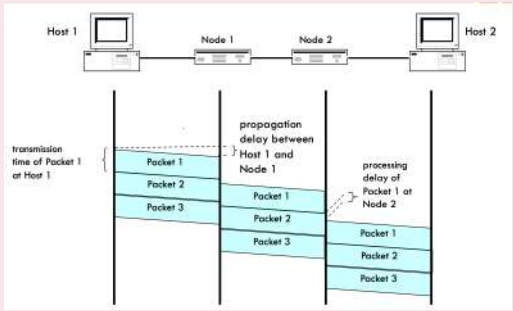
1.1 העברת פקטות (Packets/Circuit Switching)

מבנה הפקטה שאנחנו שולחים לרשת מורכב משני רכיבים:

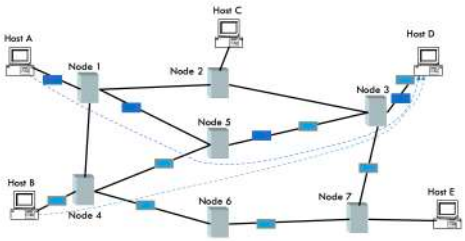
1. מידע (Payload) - מה שאנחנו רוצים לשלוח בפקטה, למשל, הודעת טקסט.
2. תחילית (Header) - המידע שאנחנו מספקים לרשת, לאיך לטפל בפקטה. זו תחילית, כיוון שאלה הבתים הראשונים בפקטה.

את המידע, היוצר מספק, והרשת מתמודדת עם כל פקטה באופן בלתי תלוי. במידע שאנחנו מספקים ב-Header אנחנו כוללים בפרט את כתובות היעד. ניתן לקבל המחשה באיור הבא:

המחשה



איור 9: העברת פקטה בין ישויות ברשת. בכל שלב, הרכיב שמקבל את הפקטה מסתכל על ה-Header של הפקטה ומעביר אותה הלאה, בהתאם.



איור 10: גלובלית, העברת הפקטות נראית כך, כאשר פקטות שונות מחכות בתור של רכיבי רשת, ולעיתים נזרקות, זה קורה, כיוון שלא הקצנו מראש משאבים, ולכן יתכן ששלחו אלינו מידע מהר יותר משאנחנו יכולים לקבל. מבנה רשת זו מראה לנו שיש צורך בניתוב.

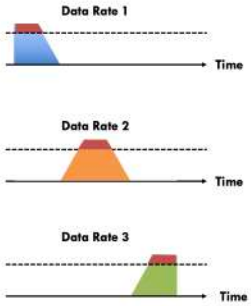
1.1.1 שימוש ב-Buffer

הבעיה המרכזית שעולה מרשת זו, היא אי הקצאת המשאבים מראש. הפתרון הנאיבי הוא לזרוק כל פקטה שמגיעה כשאנחנו לא פנויים. אמנם, זה אומר שברגע נתון אנחנו עלולים לזרוק מאות פקטות, ואחר כך שוב, ורק במקרים נדירים באמת נקבל פקטה. לכן, פתרון טוב יותר יהיה לשמור buffer שבו נאחסן פקטות, ונזרוק פקטות חדשות שמגיעות כאשר הוא התמלא.[1]

יחד עם זאת, גם עם buffer אינסופי, פקטות עדיין יכולות להאבד. זה יכול לקרות בגלל בעיה ברשת עצמה, שינוי בניתוב ועוד.

1.1.2 Statistical Multiplexing

ב-Packets Switching אנחנו מניחים שישויות שונות שולחות מידע בתדירות גבוהה, אבל בזמנים שונים ולא במקביל. נביט באיור הבא:



איור 11: אם כל אחד מהערוצים שולח מידע בקצת יותר משליש ממה שהרשת יכולה להכיל, ביחד, רק שניים יכולים לקבל מענה. אבל, אם כל אחד מגיע ל-Peek בזמן אחר, כולם יקבלו מענה. זו ההנחה שלנו.

אנחנו מאמינים שהמודל עובד, בגלל הנחת היסוד שלנו.

[1] יש תורה שלמה למידול הגעה של פקטות, זמן המתנה בחוצצים, זמן ממוצע ועוד. אנחנו לא נגע בזה.

חוק המספרים הגדולים וחלוקה כאשר אנחנו מקצים משאבים, ראינו שבעיה שיכולצה לצוץ היא ש-$A < P$. יחד עם זאת, מחוק המספרים הגדולים, לכל התפלגות $\mathbb{P}$ כך ש-$\mathbb{E}[\mathbb{P}] = A$, אם נדגום N פקטות מ-$\mathbb{P}$ נקבל כי בהסתברות גבוהה דרושים $N \cdot A$ משאבים, ולא $N \cdot P$. זה אומר שכאשר אנחנו מחלקים את הרשת בין ישויות, אנחנו מחליקים את הקפיצות!

1.1.3 השוואה בין Circuit/Packets

נסקור את היתרונות והחסרונות של שתי הרשתות.

Circuit Switch

יתרונות

- משאבים מובטחים.
- הפשטה - הם מאפשרים לישויות לתקשר בפשטות, מבלי לדאוג לסדר של המידע.
- העברה מידע פשוטה - ההעברה מבוססת על תדירות או על $time - slots$ ולא על ה-header.
- מעט overhead - צריך רק להעביר את המידע, לא צריך לעבור על ה-header.

חסרונות

- משאבים מבוזבזים - תעבורה גדולה מיישות אחת לא תתפזר על כל המשאבים, אלא רק על אלה שהוקצו לה.
- חסימת חיבורים - אם אין משאבים, לא ניתן לתקשר.
- הרשת חייבת לשמור את המידע על השיחות הקיימות. זה אומר ש-Switch צריך לדעת על כל השיחות שעוברות דרכו.
- דיליי - אין אפשרות לתקשר בלי שהחיבור נוצר.

Packets Switch

יתרונות

- אמינות - כאשר רכיב ברשת נופל, אפשר לעבור לרכיב אחר ולהשתמש בנתיב שונה.
- יעיל יותר מ-Circuit Switching - כי משתמש ב-Stat. Mux.
- קל יותר לחבר רשתות - כל שצריך לעשות הוא להסכים על מבנה הפקטות, שכן הן לא צריכות להבטיח משאבים מסויימים.

חסרונות

- עומס (Congestion) - צריכים לטפל במקרה של עומס יתר על הרשת, היות שהרשת לא מקצה משאבים, יישויות ישלחו מידע ויצפו לתשובה, גם אם הרשת עמוסה.
- רכיבי רשת מורכבים יותר - הם צריכים להיות מסוגלים לקרוא את ה-Header, לשמר Buffer, ולזרוק פקטות לפי מדיניות מסויימת.
- קשה להבטיח משאבים - רשת זו כלל לא נועדה להבטיח משאבים.

2 שכבות ו-Broadcast

רשת האינטרנט בנויה באופן מודולרי - משימות יחולקו לרכיבים שונים, וכל רכיב יוכל לסמוך על האחר, מבלי להתעסק במשימה של הרכיב האחר. כל רכיב יתקשר עם הרכיב במשימה הבאה, ולא תתאפשר תקשורת בין רכיבים ממשימות אחרות. המטרה שלנו היא למדל את הרשת, ולכן אנחנו מגדירים לה שלושה עקרונות מנחים:

עקרונות הרשת

- Layering - חלוקת הרשת למודולים שונים, שמתקשרים אחד עם השני באופן היררכי. זו גם הצורה הספציפית של מודולריות באינטרנט.
- עקרון ה-$End-to-End$ - הרשת מבטיחה לנו שני דברים - העברת מידע, ודבר שני - כל מה שהיא תספק, זה העברת מידע! היא לא מבטיחה שהוא יגיע, לא אבטחה, ולא שמירת מידע על התקשורת.
- $Fate-Sharing$ - (אחדות הגורל) מי שמודע לקיום של שיחה אלה רק הצדדים שמקיימים את השיחה, לא ה-routers. כלומר, המקום היחיד ששומר מידע שקשור לשיחה, הוא הקצוות, לא הנתבים.
- בפועל הרשת מפרה את עקרונות אלה. למשל, ה-firewall של המחשב שלנו עובר על כל המידע בפקטה, ולא רק על המידע הרלוונטי ברכיב הנתון.

2.1 Layering

כדי ליצור מודולריות אנחנו צריכים להגדיר את המשימות של הרשת, שכן לא ניתן לצפות שיהיה אלגוריתם אחד שיטפל בהכל. המשימות שלנו הן הבאות:

1. העברת אלקטרונים בכבל.
2. העברת ביטים בכבל.
3. העברת פקטות בכבל.
4. העברת פקטות בתוך רשת מקומית.
5. העברת פקטות דרך רשתות מקומיות שונות.
6. ווידוא הגעה של פקטות.
7. ביצוע מניפולציה למידע.

הדרך שלנו לטפל בכל משימה היא באמצעות שכבות - כל שכבת אחראית למשימה אחת, ומתקשרת עם השכבות שמעליה. השכבות ממוספרות מ-1 עד 7 כאשר שכבת עם מספר נמוך כנראה מתעסקת במשימה שהיא יותר $low-level$ ולהפך.

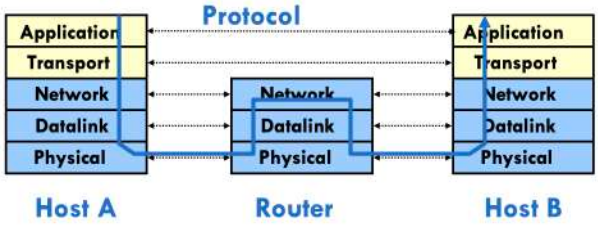
כאשר נשלח מידע, הוא יעבור מהשכבה העליונה ביותר, לשכבה הנמוכה, ולאחר מכן, מהשכבה הנמוכה לגבוהה ביותר. יש לזה אנלוגיה במציאות, שכן זהו מודל היררכי. נניח שיש לנו חברה עם מנכ"ל, מנהל צוות, ועובד בצוות. כשהמנכ"ל משנה את המוצר, הוא מעביר את המידע לראש הצוות, שמעביר את המידע לעובדים שלו, שאחראים לשינוי המוצר, וכשהשינוי מבוצע על ידי העובדים, הם מעבירים את המידע לראש הצוות, שמעביר למנכ"ל - מעבר מלמעלה למטה, ומלמטה למעלה.

נרשום את השכבות והמטרות שלהן:

1. העברת אלקטרונים בכבל - כלול בשכבה הבאה.
2. **השכבה הפיסית** - העברת ביטים בכבל.

3. העברת פקטות בכבל - כלול בשכבה הבאה.
4. **שכבת הקו** - העברת פקטות בתוך הרשת המקומית.
5. **שכבת הרשת** - העברת פקטות בין רשתות מקומיות שונות.
6. **שכבת התעבורה** - ווידוא שהמידע הגיע ליעד.
7. **שכבת האפליקציה** - ביצוע מניפולציה על המידע.

העיסוק המרכזי שלנו יהיה שלוש שכבות הביניים - **שכבת הקו, שכבת הרשת ושכבת התעבורה.** זאת מכיוון ששכבת האפליקציה עוסקת בבניית האפליקציות ולא ברשת, והשכבה הפיסית עוסקת בפיסיקה של העברת המידע, ולא ברשת עצמה. נמחיש זאת באמצעות האיור הבא:



איור 12: שכבה נמוכה עוטפת את הפקטה במעטפה שלה אנו קוראים header, ואותו מערכת ההפעלה צריכה לפענח. ככל שעולים בהיררכיה כך עולה מספר המעטפות. כשהפקטה נשלחת, כל שכבה פותחת את המעטפה הנוכחית, ומעביר את המידע הלאה לשכבה הבאה, ככה שכל שכבה פותחת את המעטפה הרלוונטית לה. כאשר צריך להעביר את המידע, כל שכבה עוטפת את הפקטה מחדש מלמטה למעלה.

ניתן להסתכל על מודל השכבות מנקודת מבט נוספת - רשת ה-overlay. רשת זו היא רשת שבנויה מעל תתי רשתות אחרות. למשל, היא תתייחס לרשתות שונות כמשתמש בפני עצמו ובכך תאפשר תקשורת בין-רשתית. במקרה של מודל השכבות, כל שכבה בנויה מעל שכבה אחרת, שהיא מעין רשת בפני עצמה, ומטפלת בבעיות שלה בעצמה. כמו שרשת ה-overlay לא צריכה להתייחס לשינויים בתתי הרשתות, כך השכבות במודל לא צריכות להתייחס לשינויים בשכבות אחרות.

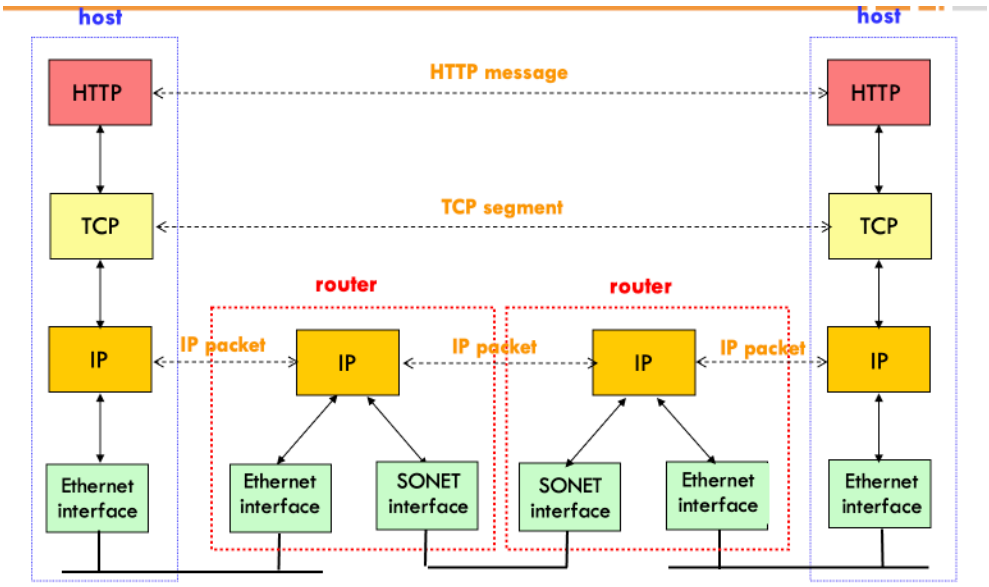
3 פרוטוקולים

הרצאה 3

הגדרה 3.1. פרוטוקול הוא הסכמה בין משתמשים על איך לבצע תקשורת. פרוטוקול משרה שתי תכונות:

1. סינטקס: איך הפרוטוקול בנוי מבחינת פורמט, סדר שליחה וקבלת ההודעות.
2. סמנטיקה: מה משמעות כל ביט - כיצד להשיב להודעות.

קיימים פרוטוקולים רבים בכל שכבה. למשל, בשכבת האפליקציה כל משתמש יכול להגדיר פורטוקול תקשורת. דבר זה גורר קיום של פרוטוקול אוניברסלי שייצור הסכמה בסיסית על אופי התקשורת, ויחד איתו, יאפשר פרוטוקולים נוספים. בלעדיו, כל פרוטוקול היה צריך לתמוך בכל פרוטוקול אחר, במקום לתמוך פשוט בפורטוקול הבסיסי. נמחיש זאת:



איור 13: ניתן לראות פרוטוקולים בשכבות שונות.

הפרוטוקול הבסיסי הוא פרוטוקול ה-IP בשכבת הרשת.

נציג את השכבות במודל במונחי: שירות, ממשק, ופרוטוקולים.

השכבה הפיסית

1. שירות: העברת ביטים בין מערכות המחוברות בכבל פיסי.
2. ממשק: מגדיר כיצד לשלוח ולקבל את הביטים.
3. פרוטוקולים: קידודים לייצוג הביטים, רמות מתח וזמן החיים של ביט.

פרוטוקולים בשכבה זו הם סיביים אופטיים, רשת אלחוט, ועוד.

שכבת הקו

1. שירות: לאפשר לרכיבים להעביר מידע ברשת המקומית. בפרט מתן כתובות מקומיות לרכיבים.
2. ממשק: שליחת הודעות לרכיבים אחרים
3. פרוטוקולים: קידודים לייצוג הביטים, רמות מתח וזמן החיים של ביט. $MAC - Multiple\ Access\ Protocol$.

פרוטוקולים בשכבה זו כוללים $Etherent, 802.11, Frame\ Relay, ATM$.

שכבת הרשת

1. שירות: העברת מידע ליעד ספציפי באמצעות כתובות גלובליות (IP). העברת המידע היא בין רשתות של שכבת הקו.
2. ממשק: שליחת פקטות ליעד ברשת, קבלת פקטות.
3. פרוטוקולים: בניית טבלאות ניתוב.

הפרוטוקול של שכבה זו הוא ה-IP, כאמור, זה הפרוטוקול הבסיסי עליו מסכימים כל הרכיבים.

שכבת התעבורה

1. שירות: העברת מידע בין תהליכים.
2. ממשק: שליחה וקבלת מידע בין תהליכים.
3. פרוטוקולים: הפיכת מידע לפקטות, אמינות, בקרה.

בין הפרוטוקולים בשכבה זו: $TCP, UDP, SCTP, DCCP, T/TCP$.

שכבת האפליקציה (שכבה 7 ולא 5)

1. שירות: כל שירות ללקוח.
2. ממשק: תלוי באפליקציה.
3. פרוטוקולים: תלוי באפליקציה.

בין הפרוטוקולים בשכבה זו: $Skype, SMTP\ (Email)\ , HTTP\ (Web)\ , Halo, BitTorrent$.

4 שכבת הקו (DataLink Layer)

Link הוא רשת Broadcast מעל השכבה הפיסית. כלומר המידע שמועבר מגיע לכל הרכיבים. ה-Link ממומש באמצעות כרטיס הרשת במחשב. זהו רכיב חומרה שקיים בכל מחשב. הוא יכול לתקשר עם רכיבים שלהם גם כרטיס רשת, כמו מחשב, או נקודת Access Point ל-Wifi.

הבעיה העיקרית היא קביעת סדר ברשת, כלומר מי מדבר ומתי. את זאת קובעת השכבה השנייה. אנחנו רק מוודאים שניתן יהיה להעביר מידע בין יישויות. מלבד זאת, קיימת הבעיה של **התנגשויות**.

- **ברשת קווית** - עד שנות ה-90 הרכיבים ברשת זו היו באותו אזור והיו מחוברים עם כפל פיסי אחד לשני. לכן היו שם התנגשויות. מאז החלו להשתמש באביזר הנקרא Switch, עליו כבר דיברנו. הוא בעצם מחלק את האזורים לתתי-אזורים, ככה שאזור ה-Broadcast של כל רכיב הוא בינו לבין ה-Switch.
- **ברשת** Wireless - גם יש התנגשויות בגלל התאבכות של תדרים.

הפרוטוקול האידאלי היה מקנה לרכיב קצה העברת מידע קבוע ברשת R, כאשר מתקיימים התנאים הבאים:

1. כאשר רכיב רוצה להעביר מידע, הוא שולח אותו בקצב R.
2. כאשר M רכיבים רוצים לתקשר, כל אחד שולח מידע בקצב $\frac{R}{M}$.
3. אין רכיב מרכזי להעברת המידע (אם היה כזה, היינו נשארים עם אותן בעיות).
4. אין סנכרון של שעונים.

הדרך שלנו לקרב פרוטוקול זה באינטרנט הוא באמצעות Random Access Protocol - שימוש באקראיות. הרעיון כאן הוא להתחיל בקצב המקסימלי האפשרי. אם המידע עברה, הכל בסדר. אחרת, ייתכן שהייתה התנגשות. על כן, עלינו לזהות האם יש התנגשות. משימה זו נפתרת על ידי קבוצה של פרוטוקולים שנקראים Multiple Access Protocols (MAC). הפרוטוקולים שנעסוק בהם מסתמכים על רעיון ה-MAC וביניהם: Slotted ALOHA, Pure ALOHA, CSMA, CSMA/CD.

4.1 פרוטוקול Slotted ALOHA

בפרוטוקול זה אנו מניחים את הדברים הבאים:

1. כל מסגרות המידע באותו גודל.
2. הזמן מחולק למקטעים שווים, כאשר כל מקטע הוא הזמן להעברת מסגרת.
3. רכיבים מתחילים להעביר מידע רק בתחילת כל Slot.
4. הרכיבים מסונכרנים.
5. אם שני רכיבים מעבירים מידע מאותו Slot, כל הרכיבים ברשת מודעים להתנגשות.

הפרוטוקול עצמו משתמש באלגוריתם הפשוט הבא:

1. אם יש פקטה חדשה - שלח אותה.

(א) אם אין התנגשות, תחזור לשלב 1 ועבור ל-Slot הבא.

(ב) אם יש התנגשות, תשלח את הפקטה בכל תור הבא בהסתברות p (כלומר בהסתברות $1-p$ היא לא תשלח), עד שהיא תגיע.

אחד החסרונות המרכזיים של אלגוריתם זה הוא אי ניצול של משאבים. שכן קיימת הסתברות $(1-p)^m$ שכל m הפקטות ברשת לא ישלחו ונקבל Slot ריק. כמו כן, יתכנו התנגשויות. היתרונות הם העברת מידע בקצב קבוע מקסימלי, פשטות וגם העובדה שהרשת היא Decentralized, כלומר הסנכרון הוא בין הרכיב ל-Slots ולא לשאר הרכיבים.

הגדרה 4.1. ה-Goodput הוא האחוז מהזמן שבו הצלחנו להעביר מידע.

הגדרה 4.2. ה-Throughput הוא האחוז מהזמן שבו הייתה העברה, לא בהכרח מוצלחת.

דוגמה 4.1. נניח שאנחנו בפרוטוקול ALOHA עם m קודקודים כאשר כולם נכשלו בהתחלה ועתה מנסים לשלוח מידע בהסתברות p. מה ההסתברות שקודקוד הצליח? זה ההסתברות שכל השאר נכשלו כלומר עבור קודקוד ספציפי $p(1-p)^{m-1}$ ועבור המאורע שקיים קודקוד כנ"ל נקבל $mp(1-p)^{m-1}$ ואנו כופלים ב-m כי המאורעות זרים. מתברר שעבור ה-p הכי טוב, ובהנחה שיש הרבה מאוד קודקודים, היינו $m \to \infty$ נקבל יעילות מקסימלית של $\frac{1}{e} = 0.37$ שזה לא משהו, במיוחד בהתחשב בעובדה שיש כאן הרבה הנחות מקלות.

עושה רושם שההנחות שלנו לא נכונות. כנראה שלא ניתן להניח שכל רכיב יתחיל לדבר בתחילת כל Slot. לכן העולם שלנו מסתבך, וכל אחד שולח מידע בזמנים שונים רציפים.

הרצאה 4 נסתכל על המקרה בו השעונים של הרכיבים לא מסונכרנים. במקרה זה הפרוטוקול הוא Pure ALOHA. כאשר רכיב i ירצה לשלוח פקטה בזמן t_0, הוא יצטרך לוודא שאף רכיב אחר לא שלח מידע בטווחי הזמנים החופפים עם $[t_0, t_0+1]$, שהן $[t_0-1, t_0], [t_0, t_0+1]$. נחשב את ההסתברות להצלחה במקרה זה.

$$\begin{aligned}
\mathbb{P}[\text{הצלחה של רכיב אחד}] &= \mathbb{P}[\text{הרכיב האחד מעביר מידע}] \cdot \mathbb{P}[\text{כל שאר הרכיבים לא מעבירים מידע} \\
&\text{איתלות} = \mathbb{P}[\text{הרכיב האחד מעביר מידע}] \cdot \prod_{T \in \{[t_0-1,t_0],[t_0,t_0+1]\}} \mathbb{P}[\text{שאר הרכיבים שותקים בטווח T}] \\
&= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1} \\
&= p(1-p)^{2(N-1)}
\end{aligned}$$

עבור p האופטימלי ו-$N \to \infty$ נקבל $\frac{1}{(2e)} = 0.18$, שזה גרוע יותר מהמקרה הקודם.

נבחין כי הירידה בביצועים, נובעת מגורם מרכזי אחד - התנגשות. בגלל שכל הרכיבים היו בלתי תלויים אחד בשני, הם לא בדקו אם מישהו מעביר כבר מידע.

4.2 פרוטוקול (Carrier Sense Multiple Access) CSMA

כדי לשפר ביצועים. נציע את השינוי הבא בפרוטוקול ALOHA - אם ערוץ העברת המידע תפוס, לא נעביר מידע, אחרת, נתפוס את הערוץ. במילים אחרות - לא נפריע לאחרים.

התוספת הנ"ל תמנע חלק מהתנגשויות, אבל לא את כולן. למשל, אם שני רכיבים מנסים לשלוח מידע באותו הזמן, כאשר הערוץ פנוי - הם ישלחו, ויתנגשו. אמנם מקרה זה זניח, היות שבטווחי זמן של מיקרו-שניות, זה לא סביר, קיימות גם התנגשויות אחרות, שנגרמות מקצב העברת המידע. למשל, אם רכיב אחד שלח מידע, ייקח זמן עד שרכיב אחר יראה אותו. במקרה של התנגשות בין השניים, לאחר שכבר נשלח המידע, אף אחד לא יעצור.

4.2.1 פרוטוקול (Collision Detection) CSMA/CD

כדי לטפל בהתנגשויות שתיארנו, נוסיף התנהגות חדשה - כאשר זוהתה התנגשות נעצור את העברת המידע. זה אפשרי בחיבורים קווים. לעומת זאת, ב-Wireless, זה יותר קשה, היות שהסיגנל יכול לדרוס סיגנלים אחרים.

מלבד זאת, עלינו לקבוע כמה זמן כל רכיב צריך להמתין עד שהוא שולח פקטה. נסמן את הזמן המקסימלי שלוקח לשני רכיבים ברשת לדעת על מידע שאחד מהם שלח ב-PROP - כלומר הזמן שלוקח לביט לעבור פיסית בין שתי הנקודות הכי רחוקות ברשת.

אזי, אם רכיב אחד שלח מידע, הרכיב השני עלול לשלוח את המידע לאחר $\mathrm{PROP}-\varepsilon$ עבור ε קטן כרצוננו, ואז למרות שהוא חיכה זמן רב, המידע ייזרק. עולה השאלה, כמה זמן לוקח למידע ששלח הרכיב הנ"ל עד שהרכיב הראשון רואה אותו? התשובה היא שגם PROP. כלומר במצב הנתון, הרכיב הראשון גילה על ההתנגשות לאחר זמן 2PROP לכל היותר.

מכאן, על הרכיבים ברשת להמתין 2PROP עד שהם שולחים מידע - כדי לדעת אם המידע שהם שלחו קודם ניזוק.

מקרה מבחן: מה היה קורה אם הם היו מעבירים מידע בזמן $2\mathrm{PROP}-\varepsilon$? הרכיב הראשון היה שולח מידע שהיה מגיע לרכיב השני לאחר זמן PROP. הרכיב שולח מידע, בהנחה שהמידע מהרכיב הראשון כבר הגיע. אמנם המידע החדש היה יוצא בזמן PROP, ומגיע לצד השני בזמן 2PROP. אך בזמן $2\mathrm{PROP}-\varepsilon$ כבר שלח מידע חדש, ולכן לאחר ε האותות יתנגשו. כלומר, הרכיב מחכה פרק זמן זה, כדי לזהות התנגשות עם **המידע שכבר שלח בעצמו**.

עובדה זו מאלצת את הרשת לספק זמן העברת מידע $\mathrm{TRANSP}\geq 2\mathrm{PROP}$. במילים אחרות, יש זמן מינימלי להעברת פקטה ברשת. נבדוק כמה השתפרו הביצועים שלנו. ניזכר כי $\mathrm{goodput}=\eta=\frac{\text{הזמן שלוקח להעביר מידע}}{\text{הזמן שלוקח להעביר מידע + תקורה}}$. במקרה שלנו, $\eta=\frac{\mathrm{TRANSP}}{\mathrm{TRANSP}+2\mathrm{PROP}}$. כדי לנתח אותו נחשב את ההסתברות האופטימלית לשליחת פקטה מוצלחת.

כמו כן, ההסתברות שרק רכיב אחד מעביר מידע ב-SLOT נתון היא $\alpha(p)=\binom{N}{1}p(1-p)^{N-1}$ וגזירה של α תתן

$$\frac{d\alpha}{dp}=N(1-p)^{N-1}-pN(N-1)(1-p)^{N-2}$$

ולכן $p=\frac{1}{N}$ אופטימלי, מה שמניב $\alpha_{\max}\approx 37\%$. לשם נוחות נניח כי $\alpha_{\max}=40\%$. אם כך, נסמן ב-A את תוחלת מספר ה-$\mathrm{Time}-\mathrm{Slots}$ שהפסדנו **לפני** שהצלחנו להעביר פקטה. אזי

$$A=\alpha\cdot 0+(1-\alpha)(1+A)$$

החלק הראשון הוא המקרה בו הצלחנו להעביר, והוא קורה בהסתברות α. החלק השני הוא המקרה בו הפסדנו. במקרה זה, צריך להעביר את הפקטה שנכשלה (1), ועוד A, כל הפקטות לאחר מכן. על כן $A=1.5$. בפרט, כמות ה-Slots הדרושים להעברת פקטה מוצלחת כולל המקרה בו הצלחנו הן $1+A=2.5$. נבחין כי $1+A$ הוא משתנה גאומטרי $\mathrm{Geo}(\alpha_{\max})$, וכרצוי מספר ה-Slots הממוצע הוא $\frac{1}{\alpha_{\max}}$. על כן

$$\begin{aligned}\eta_{\mathrm{CSMA/CD}}&=\frac{\mathrm{TRANSP}}{\mathrm{TRANSP}+\mathbb{E}\left[\#\text{Wasted Slots Per Packet}\right]}\\&=\frac{\mathrm{TRANSP}}{\mathrm{TRANSP}+A(2\cdot\mathrm{PROP})}\\&=\frac{\mathrm{TRANSP}}{\mathrm{TRANSP}+3\mathrm{PROP}}\end{aligned}$$

נסמן ב-$\alpha=\frac{\mathrm{PROP}}{\mathrm{TRANSP}}$ ונקבל $\eta_{\mathrm{CSMA/CD}}=\frac{1}{1+3\alpha}$. על כן, נקבל $\eta_{\mathrm{CSMA/CD}}$ טוב, כאשר α קטן כלומר $\mathrm{PROP}\ll\mathrm{TRANSP}$. מכאן אנחנו מסיקים כי רשת Broadcast לכל העולם זה רעיון רע - אנחנו נקבל PROP מאוד גדול, ולכן α יגדל, ו-$\eta_{\mathrm{CSMA/CD}}$ ייקטן.

הערה 4.1. נעיר כי ניתוח מדויק יותר היה מניב $\eta_{\mathrm{CSMA/CD}}=\frac{1}{1+5\alpha}$.

4.3 חיבור לינק לוגי בפועל

עד כה דיברנו על פרוטוקולים אפשריים לטיפול בבעיית ה-Multiple Access. עכשיו נראה מה הפרוטוקול שמשתמשים בו בפועל.

כתובת MAC הדרך שבה הרשת המקומית מבדילה בין רכיבים בהעברת מידע דרך לינק, היא באמצעות כתובת MAC, המכילה $6\mathrm{bytes}=48\mathrm{bits}$. הכתובת צרובה על גבי כרטיס הרשת של המחשב (NIC), ולא מעידה על מיקומו של המחשב.

היות שהיא צרובה עליו, היא לא תשתנה לעולם. מכאן גם נובע החסרון שלה - בסקאלה גלובלית היא לא מאפשרת לזהות מיקום. הקצאת הכתובת מנוהלת על ידי ה-IEEE, והיצרן קונה מהם קבוצה של כתובות שהוא יכול לספק למכשירים שלו. אפשר להקביל כתובת זו לתעודת הזהות שלנו.

4.3.1 פרוטוקול Ethernet

הפונקציונליות שכתובת ה-MAC מאפשרת היא העברת מידע מכרטיס רשת אחד דרך לינק, לכרטיס רשת אחר. הפרוטוקול שאחראי לפתרון בעיית ה-MAC **בחיבור פיסי**, הוא פרוטוקול ה-Ethernet, שמסתמך על ה-NIC. ראשית, הוא לא יקר לשימוש והכי חשוב - הוא פשוט.

הפרוטוקול משתמש באלגוריתם CSMA/CD, באופן הבא:

1. אם ה-NIC קיבל מידע משכבת הרשת, ניצור Frame.
2. אם ה-NIC מוצא חיבור פנוי, הוא מתחיל העברת המידע. לעומת זאת, אם הוא החיבור תפוס, הוא מחכה עד שהוא יתפנה ורק אז מעביר.
3. אם ה-NIC העביר את כל ה-Frame מבלי לגלות העברה אחרת, הוא סיים.
4. אחרת, אם ה-NIC מגלה על העברה אחרת תוך כדי העברת המידע שלו, הוא מפסיק להעביר מידע, ושולח סיגנל JAM (בגודל 48 ביטים). כלומר, הוא צועק בכל רחבי הרשת מספיק חזק, כדי שכולם יבינו שהייתה התנגשות.
5. לאחר הפסקת העברת המידע עקב הכשלון, ה-NIC נכנס למצב של המתנה. כדי שמי שכבר שיש יותר התנגשויות נחכה יותר, ניצור תלות בין זמן ההמתנה לבין מספר ההתנגשויות. תלות זו תהיה אקספוננציאלית - לאחר ההתנגשות ה-m-ית, הוא יחכה פרק זמן התלוי ב-m באופן אקספוננציאלי ולאחר מכן יחזור לשלב 2 של האלגוריתם. הדרך בה הוא בוחר את הזמן, היא באמצעות דגימה אחידה של K מהטווח $\{0, 1, 2, \ldots, 2^m - 1\}$ והמתנה של $K \cdot 512$ זמן להעברת ביט יחיד ברשת (שחסום מלמטה על ידי 2PROP).

הרצאה 5

במקרה זה $\eta \approx \frac{1}{1+5\cdot\frac{T_{prop}}{T_{transp}}}$. כמו כן, יש כאן מנגנון CSMA/CD היות שאנחנו שולחים $jam - signal$ בגודל 48 ביטים כדי ליידע את כולם.

> **!** על אף שרכיבים הפיסיים ותכונות הרשת השתנו עם השנים, כמו למשל ה-Bandwidth שגדל משמעותית עם השנים, הפרוטוקול שרד, והוא עדיין בשימוש נרחב.

4.3.2 פרוטוקול IEEE 802.11

כאן יש שימוש Wireless וב-$access - point$, אליה כל רכיב שולח מידע במטרה להעביר אותו לרשת. זה אומר שיש אי שוויון בין רכיבים - ה-$access - point$ שולט על כולם. הבעיה כאן היא שלא ניתן להפעיל מנגנון זיהוי התנגשויות, כי ייתכן שסיגנל אחד פשוט דרס סיגנל אחר, והסיגנל שדרס לא יידע בכלל שהייתה בעיה. לכן צריך ליצור מנגנון טוב יותר. עבור השולח, נשלח RTS (request to send) ל-$access - point$. אם הוא יחזיר CTS (clear to send), נתחיל להעביר את המידע.

Circuit Switch

השולח

1. כאשר הערוץ פנוי לפרק זמן DIFS, נעביר את כל המידע. כלומר, כאשר שלחנו RTS וקיבלנו CTS מה-ap.

   (א) במקרה של התנגשות ב-ap נצטרך לשלוח שוב RTS.

   (ב) הרכיב לא יודע איפה נמצאת ה-ap אך הוא כן יודע שהיא באותו אזור broadcast.

2. אחרת, אם הערוץ תפוס, נגריל את הזמן להמתנה, וננסה להעביר שוב בסופו.

3. לאחר העברת המידע, נחכה ל-ack מה-access − point. אם לא הגיע, נחזור לשלב 2.

access − point

1. אם הגיעה בקשת RTS והערוץ אכן פנוי, נחזיר CTS לכולם.

   (א) כולם מקבלים CTS וככה רק מי ששלח מידע, ידע שהוא יכול לשלוח, וכי מי שלא שלח, ידע שהוא לא יכול להפריע.

2. אם הגיעה הודעה תקינה, נחזיר ack לשולח לאחר SIFS זמן.

אנחנו ממתינים זמן המתפלג אקראית, כדי להתמודד עם התנגשויות, בדומה ל-Ethernet.

בזאת אנחנו מסיימים את הדיון שלנו על רשתות broadcast, ועוברים למודל של switch בו רשת ה-broadcast עצמה היא בין הרכיב ל-swtich.

4.4 Switch

כפי שתיארנו בעבר, ה-switch ינתב בקשות של רכיב אחד לרכיב אחר. עד כה לא הסברנו איך הוא יודע לעשות זאת. באמצעות כתובות MAC. כאשר רכיב אחד ירצה להעביר מידע לרכיב אחר, אם ה-switch יודע מה המיפוי של כתובת ה-MAC לפורט אליה הרכיב מחובר ל-switch, הוא יוכל להעביר את המידע. אחרת, הוא יצטרך להעביר את המידע לכולם ולבצע את המיפוי לפי מי שיענה. כלומר, הוא ישמור אצלו switch − table של פורטים וימפה אותן לכתובות MAC:

```
record line associated with sending host
index switch table using MAC dest address
if entry found for destination {
        if dest on segment from which frame arrived {
                drop the frame
        } else {
                forward the frame on interface indicated
        }
} else { flood }
```

הבעיה היא, שזה מגביל את גודל הרשת למספר הפורטים של ה-switch, לכן, נוכל להרחיב אותה על ידי הוספה של switches, כך שעכשיו פורט יוכל להיות ממופה ל-switch. כך נקבל מעין רשת היררכית, בה יש switches שמחברים switches אחרים, שמחברים בסוף רכיבים. בצורה זו, לא נקבל flooding ברשת, היות שכל switch לומד לאן להעביר את המידע.

! ה-switch דורס רשומות בכל פעם שהוא מקבל מידע מפורט, על מנת לעדכן את המיפוי לפי השינויים ברשת. יחד עם זאת, זה יכול להוביל לבעיה הבאה, כאשר הגרף של הרשת מכיל מעגל. במקרה זה, ייתכן שרכיב A ישלח הודעה לפורט 1 ב-switch A, אך גם ל-switch B באותו אזור broadcast. שניהם יעבירו את המידע לרכיב B בפורט 2. אך, הפלא ופלא, ייתכן היות שה-switches באותו אזור broadcast, ייתכן שההודעה שהעביר switch B ל-B הגיעה ל-switch A, ולכן הוא ידרוס את המיפוי של A לפורט 1 וימפה את A לפורט 2, וזו בעצם סתירה.

מכאן אנו מסיקים שצריך לדאוג שלא יהיו מעגלים. היות שהגרף שלנו קשיר - נשתמש בעץ פורש. היות שהגרף שלנו הוא לא רק עץ, אם צלע נפלה, נוכל לעדכן את העץ הפורש שלנו. בצורה זו, ה-switches ברשת יתעלמו מפורטים שהם לא בעץ.

4.4.1 פרוטוקול Spanning Tree

המטרה שלנו היא ליצור ST של ה-LANS, אותו נחשב באמצעות עץ פורש של ה-switches.

החישוב של העץ הפורש הוא מבוזר - ולכן האלגוריתם הוא לא רב-שלבי קלאסי. אנחנו נעמיד פנים שהוא כן, ונזכור שבפועל הם מתרחשים במקביל.

בחירת השורש של העץ נניח שלכל switch יש מזהה ייחודי. כל switch יעביר את המזהה שלו לשכנים שלו, וכל אחד יזכור את ה-id הכי נמוך שהוא ראה עד כה - זה השורש. זה אומר שהשורש מתעדכן בזמן ריצה.

חישוב המרחק הקצר ביותר לשורש כל קודקוד ימצא את המרחק הקצר ביותר שלו מהשורש. אנחנו עושים זאת באמצעות אלגוריתם $\text{Bellman}-\text{Ford}$ מבוזר אופן הבא: בהנתן שורש s.

(1) : נאתחל $\text{dist}_s = 0$, כלומר המרחק הקצר ביותר שלו מהשורש (מעצמו) הוא 0.

(2) : לכל v נעדכן את המרחק הקצר ביותר שלו להיות $\text{dist}_v = \min\{\text{dist}_u + 1 \mid u \in \text{neighbors}(v)\}$, בצורה זו, הוא קובע את האבא שלו בעץ. במקרה הכללי, נשתמש בפונקציית משקל $\text{dist}_v = \min\{\text{dist}_u + w(u,v) \mid u \in \text{neighbors}(v)\}$ כל עוד המשקולות אינן שליליות.

האלגוריתם עובד באופן בלתי תלוי במרחקים ההתחלתיים, וגם כאשר הרשת היא אסינכרונית.

למעשה המרחק מיוצג על ידי LAN בין switch ל-switch.

הרצאה 6 בשלב זה יש לנו ST של ה-Switches, אבל אנחנו רוצים ST של LANS, נבחין כי אפשר להשיג אותו על ידי בחירה של פורטים בכל switch.

אנחנו מסיקים את ה-ports בכל switch קודם כל לפי המרחק שלו, ולאחר מכן לפי המספר שלהם (שובר שוויון), כלומר נזהה את ה-(RP) Root Port שהוא האבא של ה-switch בעץ ואת ה-Designated Ports שהם הילדים שלו בעץ. בנוסף, ההודעות הקשורות לחישוב ה-ST עוברות דרך כל הפורטים. מכאן, כדי לחשב את ה-LAN ST נבחין כי כל אזור broadcast בעל לפחות switch אחד, ורק אחד יעביר פקטות, על ידי הסימון של ה-Designated ports.

5 שכבת הרשת (Network Layer)

שאלה מדוע הרשת איננה רשת LAN אחת גדולה?

ראינו שרשת מסוג זה ננהל באמצעות פרטוקול ה-ST, ויש לכך השלכות רבות. נניח למשל שמישהו רוצה לבצע תקשורת באמריקה, אם הוא לא יודע את היעד, ההודעה תעבור ברחבי העץ ותגיע לכל הרכיבים, כולל אלינו באוניברסיטה העברית. זה אומר שנשמור מידע על הרכיבים בקצה השני של העולם. זה מצב בלתי נסבל, היות שהמידע רב, ולא באמת נחוץ. **הדבר נובע מכך שכתובת** MAC **לא מעידה על מיקום הרכיב**. מכאן, טבעי לנסות להגדיר רשת שמספקת כתובות "שימושיות" לזיהוי על המפה, ולא רק לזיהוי רכיב ספציפי.

שכבת הרשת נותת לבעיה זו מענה על ידי סיפוק כתובת חדשה, המכונה כתובת IP. כתובות אלה נמצאות תחת קטגורייה של CIDR : Classless InterDomain Routing, ומורכבות משני חלקים. ראשית, אנחנו רוצים לתת משמעות למיקום של הרכיב, ולכן החלק הראשון בכתובת ייצג זאת. שנית, עלינו לזהות את הרכיב עצמו, ולכן החלק השני ייצג אותו. פרקטית, אנחנו מגדירים תת-רשת, בה צריך לזהות את הרכיב, ולכן עלינו לזהות רק את תת-הרשת שהרכיב נמצא בה, ואז בפנים את הרכיב עצמו.

הכתובת עצמה היא בפורמט של $a.b.c.d/x$ כאשר a, b, c, d בטווח של $0 - 255$, כלומר בית אחד, ו-x הוא מזהה תת-הרשת. כלומר, הוא מספר הביטים שמייצגים את תת-הרשת (Subnet) בכתובת. למשל, אם $x = 23$ זה אומר שמספר שעלינו להסתכל על 23 הביטים הראשונים בכתובת, והביטים שאחריהם ייצגו את המזהה של הרכיב בתוך הרשת. x קרוי גם ה-$subnet - mask$.

דוגמה 5.1. נביט בכתובת 200.23.16.0/23. מהי ה-subnet? אנו צריכים 23 ביטים מצד שמאל של הכתובת, כלומר

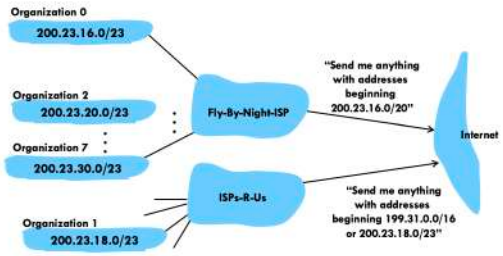
$$11001000.00010111.00010000$$

כאשר הביט האחרון לא נכלל, היות שנדרשים 23 ביטים ולא 24.

היופי במנגנון זה הוא שצריך לדעת רק את ה-subnet של האזור כדי לגשת לרכיבים שלו. באופן זה לא צריך לתשאל את כל הרשת איפה נמצא כל רכיב, או לשמור את המידע.

הרצאה 7

כל נתב ברשת, יכיל כתובת IP **שונה** בכל פורט. במודל שכבת הרשת, אנחנו מכירים רכיב חדש - הנתב. הנתב ינתב מידע בין subnets שונות, כאשר, כל subnet בפני עצמה תנוהל ב-$layer - 2$, כלומר באמצעות switch. באופן זה, אנחנו מסוגלים לחלק את הרשת לתתי-רשתות ב-$Layer - 3$, כאשר כל תת-רשת מחולקת גם היא לתתי-רשתות, לפי פורטים בנתבים. כלומר, פורט בנתב בעל כתובת IP שבפועל מייצגת רשת, בתוכה יש רכיבים. מעבר לכך, נוכל לחבר תתי רשתות לתתי רשתות אחרות, ובמובן זה, לעבור בין ספקי אינטרנט, כל זאת על ידי הצהרה, שאנחנו בעלי תתי הרשתות הנ"ל:



איור 14: מעבר בין ספקי אינטרנט, על ידי בקשת כתובת IP יותר ספציפיות מתוך אזור מסויים. במקום לבקש את 200.23.0.0/16, ביקשנו בדיוק את 200.23.18.0/23.

מכאן עולה בעיה - מה אם אנחנו לא באמת אחראים על תת הרשת שאנחנו מצהירים עליה? למשל נבקש קבוצת כתובות ספציפיות מתוך ה-subnet של google.
זה אכן מפגע אבטחה, אך הוא דורש שיתוף פעולה של הרבה גורמים.
המידע עצמו של הפקטה בשכבה זו, יכול להיות מוצפן, אבל ה-$meta-data$ לא יכול להיות, כדי שהנתבים יבינו מה לעשות עם המידע. לכן, תאורטית, נוכל לעקוב אחר מידע זה. ה-NSA דיווח שהוא "הורג אנשים באמצעות $meta-data$".

שאלה שעולה היא כיצד כל רכיב מקבל כתובת IP. היות שאנחנו רוצים שהכתובת תשליך על המיקום, לא ייתכן שרכיב יחזיק בכתובת אחת לכל ימי חייו, מהסיבה הפשוטה שהוא כנראה זז. על כן, דרוש מנגנון לחלוקת הכתובות - DHCP.
עלינו גם להבין כיצד כתובת כמו google.com מתורגמת לכתובת IP - DNS.

נבחין כי יש לנו 2^{32} כתובות. זה נשמע הרבה, אבל היסטורית, החלוקה של הכתובות היא מוטה. MIT למשל, בעלת 8/ כתובות, כלומר 2^{24} כתובות IP. בעבר, האוניברסיטה העברית הייתה בעלת מספר גבוה יותר של כתובות IP מאשר סין. בפרט, יש לה 16/ כתובות, שזה 2^{16} כתובות IP.

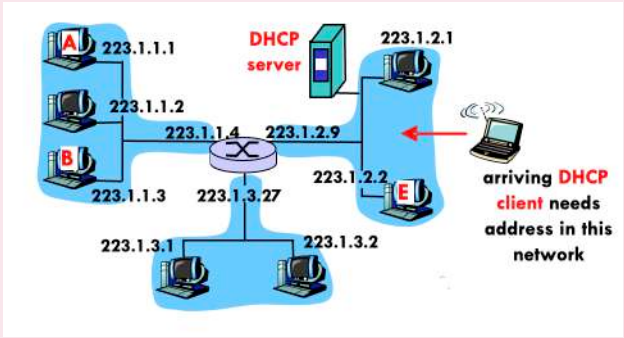
5.1 פרוטוקול DHCP לחלוקת כתובות

הדרך שבה רכיב ברשת יקבל כתובת IP היא באמצעות שליחת בקשה לשרת DHCP (Dynamic Host Configuration Protocol).
המטרה בפרוטוקול זה היא חלוקת כתובות בצורה דינאמית, כך שרכיב יוכל לעבור לכתובת אחרת, וכתובת שהייתה בשימוש תוכל לעבור לרכיב אחר. לא פחות חשוב, נרצה לאפשר לרכיבים חדשים להצטרף לרשת.

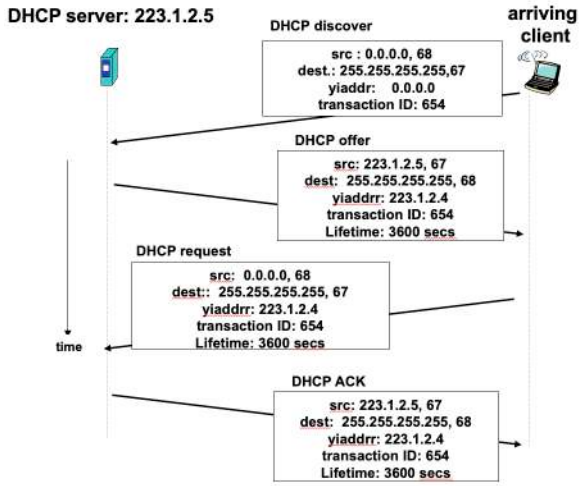
הדרך שבה זה יתבצע, היא באמצעות צעקה. כלומר:

1. הרכיב שרוצה כתובת IP ישלח בקשת DHCP על ידי צעקה ברשת.
2. שרת ה-DHCP ישמע את צעקה, יזהה שדרושה כתובת, וישלח DHCP ack עם כתובת IP.

בקשת כתובת IP



איור 15: רכיב חדש המצטרף לרשת שולח בקשה לשרת ה-DHCP ורכיבי הרשת מפעפעים הודעה זו לשרת, שבתגובה, שולח כתובת.



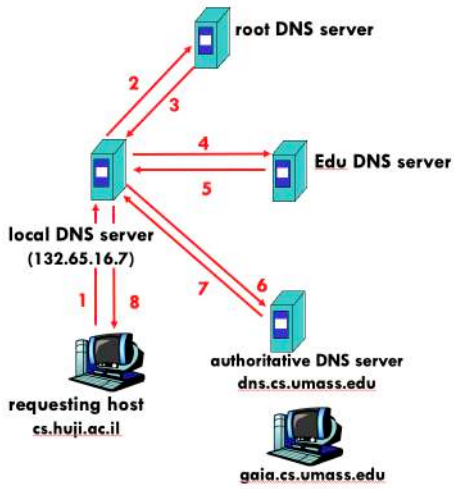
איור 16: תוכן התקשורת בין הרכיביםχ תחילה שולח הרכיב בקשה עם כתובת מקור 0.0.0.0 שמשמעותה "אין לי כתובת", עם יעד 255.255.255.255 שמשמעותו "העבר לכולם". הבקשה מגיעה לשרת, שמעביר כתובת IP כאופציה, עם הזמן שהכתובת תהיה זמינה לרכיב, שכן, אם הרכיב עוזב את הרשת/נכבה, אין סיבה שהיא תשמש כמזהה שלו.

בנוסף ה-DHCP אחראי גם על מתן כתובת שרת ה-SubnetMask ושרת ה-DNS עליו נרחיב מיד.

5.2 פרוטוקול DNS

הרשת מכילה רכיבים עם שמות שונים, כמו למשל www.google.com. כיצד אנחנו יודעים בזמן אמת לגשת אליהם? הרי אנו צריכים את כתובת ה-IP שלהם. ראשית עלינו להבין את מבנה השם. השם בנוי בצורה היררכית, כלומר כל רכיב מסמל אזור היררכי בפני עצמו, כאשר השם הימני ביותר הוא האזור הכללי ביותר. כלומר כדי להבין איפה נמצא השרת בדוגמא שלנו צריך ללכת לפי www $\rightarrow$ google $\rightarrow$ com $\rightarrow$. כאשר . הוא האזור הכולל את כל הרשת, ומכונה ה-root. בצורה זו הרשת היא היררכית.

הפרוטוקול באמצעותו אנחנו מסיקים כתובות IP נקרא DNS (Domain Name System). הדרך בה הוא פועל היא בדומה למה שתיארנו - באופן היררכי. כלומר, יש שרת DNS שאחראי על כל אזור היררכי הוא יענה על שאילתות הקשורות לאזור שלו. למשל, עבור www.google.com יפנה הרכיב לשרת DNS שקיבל מה-DHCP, שאחראי על האזור שלו. השרת, יפנה לשרת . שישלח לו את כתובת ה-ip של com.,, הוא יפנה לשרת זה, שישלח לו את הכתובת של google.com., יפנה לשרת זה, שישלח לו את הכתובת של www.google.com וככה יקבל את הכתובת. שרת שאחראי על אזור נקרא שרת Authoritative.



איור 17: תהליך קבלת הכתובת

היות שכל תשאול DNS כנ"ל הוא תהליך יחסית ארוך, השרתים שומרים תשובות ב-Cache, כדי לחסוך זמן. בפרט, מי שישמור את הכתובת עצמה ב-Cache. כל רשומה כזו תהיה בעלת שדה ttl (time to live), ותמחק אחרי הזמן שצוין שם. רשומות שהגיעו מ-cache הן לא תשובות authoritative בניגוד לתשובות משרת authoritative.

אין שרת root אחד, וזה במטרה למנוע ניתוק של הרשת, שכן העלמות שלו, תמנע מאיתנו גישה לאינטרנט. כמו כן, קיימים סוגי רשומות DNS שונים:

1. רשומת A - ממפה שם לכתובת, למשל google.com $\rightarrow$ 6.6.6.6. מה שאנחנו מקבלים לאחר ששלחנו בקשה לשרת ה-DNS שלנו.
2. רשומת NS - מיפוי של שם של שרת לכתובת. למשל com. $\rightarrow$ 1.1.1.1, מה שהשרת DNS שלנו מקבל כאשר הוא שולח בקשה לשרת DNS אחר, שאחראי על אזור של שרתים אחרים.

3. רשומת PTR - מיפוי הפוך, כלומר כתובת IP לשם, רק שלכתובת אנחנו מוסיפים סיפא:

$$6.6.6.6.\text{in}-\text{addr.arpa} \rightarrow \text{google.com}$$

למעשה, ב-$\text{in}-\text{addr.arpa}$ נמצאים כל רשומות ה-ptr, והן מהוות מאין אסמכתא נוספת לכך שהמיפוי של $\text{name} \rightarrow \text{ip}$ אמין.

4. רשומת CNAME - מיפוי של כתובת alias לכתובת מקורית. למשל $\text{google.com.alias} \rightarrow \text{google.com}$. יש לזה שימוש במתקפות, נרחיב על כך בהמשך.

5. רשומת MX - מיפוי של כתובת mail לשרת mail.

פקטת DNS מורכבת משלושה Sections - Query, Answer, Additional, כאשר מטרת ה-Additional היא שמירה של מידע רלוונטי.

דוגמה 5.2. כשנרצה לקבל את הכתובת של www.google.com נתחיל בתשאול של . ונקבל (השרת DNS שלנו יקבל) רשומת NS מהצורה (למשל) $\text{a.gtld}-\text{servers.net}$, כאשר $\text{gtld}-\text{servers}$ זה ממלא מקום לשרת גלובאלי שאחראי על אזור מסויים, למשל edu/com/io, נקבל גם רשומת A שממפה את הכתובת הנ"ל לכתובת IP. לאחר מכן נפנה לשרת זה (באמצעות כתובת ה-IP שקיבלנו) ונקבל רשומת NS מהצורה ns1.google.com עם רשומת A מתאימה. נפנה לשרת זה, ואז נקבל כבר רשומת A בלבד שתמפה $\text{www.google.com} \rightarrow 6.6.6.6$.

בכל שלב אנחנו מקבלים שתי רשומות: NS, A. רשומת ה-A מתקבלת כדי שלא נצטרך לבצע עוד תשאול עבור הדומיין שקיבלנו ברשומת ה-NS. כמו כן, נבחין כי לו היינו מבצעים תשאול נוסף ולא מקבלים רשומת A, היינו נכנסים לבעיה. ייתכן ששרת שאחראי על google.com נמצא בתוך האזור google.com למשל ns1.google.com ואז כשננסה להבין מה הכתובת שלו, יפנו אותנו לשרת שאחראי על com. שיפנה אותנו לשרת שאחראי על google.com שהוא ns1.google.com ואז נגיע לאותה בעיה. כלומר יש כאן לולאה. על כן, ההוספה של רשומת A מצילה אותנו. בפרט, במקרה המעגלי הנ"ל אנחנו קוראים לרשומת ה-Aגם glue record. היות שאזור ה-Answer מכיל רשומת NS, אזור ה-Additional יכיל את ה-glue record.

השרת שמבצע את התשאול הנ"ל הוא לא שרת DNS רגיל, ונקרא Recursive Resolver היות שהוא מבצע תשאול רקורסיבי, ושומר ב-Cache מידע.

הוספת שרתי DNS

הרצאה 8

נניח שהקמנו סטארטאפ חדש בשם FooBar. קיבלנו מה-ISP בלוק של כתובות, למשל 212.44.9.128/25, קנינו את הדומיין foobar.com, וסיפקנו כתובת לשרת ה-DNS ה-Authoritative לאזור זה באמצעות רשומת

foobar.com NS dns1.foobar.com

dns1.foobar.com A 212.44.9.129

האם זה מספיק? התשובה היא שלא. ייתכן שהשרת ה-Authoritative לא יידע לענות על הדומיין www.foobar.com ולכן נצטרך להקים שרת dns פנימי שיהיה אחראי על זה, לכן נצטרף לספק לשרת ה-Authoritative רשומת A לשרת הפנימי שלנו. נצטרך לספק גם עוד מידע כמו MX record.

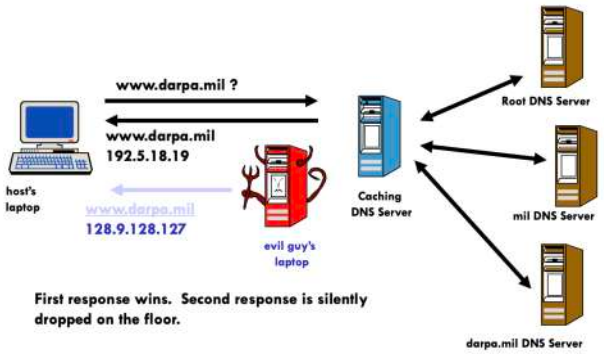
אבטחה, חולשות ופתרונות

שימוש ב-DNS צריך להיות מאובטח. למשל, כאשר אנחנו מחדשים סיסמא, מה יקרה רוב הפעמים? ישלח אלינו מייל עם לינק לחידוש הסיסמא. אבל, שליחת הודעה לכתובת מייל, דורשת ביצוע resolving לכתובת IP, ולכן, אם נשתלט על שרת

המייל שאחראי למיפוי כתובת Mail – IP, נוכל לכוון את המייל לתיבת הדואר שלנו, כתקופים, ולשלוט בסיסמא החדשה של הלקוח, זו רק מוטיבציה.

אחת הבעיות הבסיסיות, היא הגנה מפני מתקפות doS (Denial of Service). במתקפות מסוג זה, אנחנו שולחים הודעות קטנות לשרת, שגורמות להרבה overhead ובכך הופכות אותו ללא זמין למשתמשים. זה נשמע חמור. אבל ב-DNS צריך להבהיר כמה דברים. נניח שביצענו מתקפת doS על כל שרתי ה-root. כמה זה ישפיע על המשתמשים? ייקח הרבה מאוד זמן עד שזה ישפיע, היות שהכתובת של שרתי ה-gtld, ctld שמורות ב-cache של שרתי DNS נמוכים בהיררכיה. אבל, מה יקרה אם נבצע מתקפה זו על ה-recursive resolver? כאן אנו נבחין בזה די מהר, היות שה-ttl של כל רשומה יחסית קצר, ולכן לא נוכל לגשת לאינטרנט, כי לא נקבל כתובות IP. במקרה זה נוכל להגדיר בעצמנו שרת DNS אחר, שעליו אנחנו כן סומכים כמו למשל 8.8.8.8 של גוגל. נעיר כי למתקפה זו וריאנט שנקרא ddoS (Distributed doS). כיום יש כלים שמנתחים את התעבורה ברשת במטרה לעצור מתקפות מסוג זה, אך עדיין יש שרתים פגיעים.

בעיה נוספת היא התחזות לשרת DNS. למשל, לקוח יכול היה לבקש כתובת IP של דומיין כלשהו. ה-Resolver היה מבצע תשאול רקורסיבי עבור הדומיין, אך במקביל, תוקף היה שולח אלינו תשובת DNS עם A רקורד, שממפה את הדומיין ל-IP זדוני שלנו.



איור 18: המחשה למתקפה. התוקף מתחזה בפקטה לשרת ה-Resolver על ידי השמה של כתובת ה-IP של ה-Resolver בשדה ה-src ב-Header ה-IP בפקטה. זה חוקי לחלוטין.

במצב כזה, הוא יכול היה לגרום לנו לגלוש באתרים שאוספים עלינו מידע ועוד. היות שתשאול רקורסיבי נדרש מה-Resolver יש לנו סיכוי לא זניח להצליח במתקפה. הבעיה היא שההצלחה כאן היא מקומית. כלומר בפעם הבאה שהלקוח ייגש לדומיין הוא ישלח שוב בקשה ל-Resolver, ולכן נצטרך לתקוף שוב. יחד עם זאת, שינוי הרכיב נתקף מניב מתקפה אפקטיבית.

5.3 DNS Cache Poisoning

נזכר שה-Resolver מתחזק Cache ובכך מונע תשאול רקורסיבי חוזר של דומיינים שבוצע עבורם מיפוי לאחרונה. זה עוזר למנוע מתקפות doS, ומפחית תעבורה ברשת. לכל רשומה ב-Cache יש שדה ttl שנקבע לפי ה-load balancing בתוך הרשת, כלומר לפי זמינות של השרתים לאורך זמן, למשל אם ידוע שכתובת ה-IP של הדומיין לא תשתנה בשעה הקרובה, נקבע ttl להיות קצת פחות משעה. יחד עם זאת, יש כאן פוטנציאל לתקיפה. נניח שאנו מבצעים את המתקפה הקודמת, אבל במקום לשלוח את התשובה ישירות ללקוח, נשלח אותה ל-Resolver ונתחזה לשרת Authoritative. ה-Resolver יעשה Cache לבקשה. היות שאנחנו, השולחים, קובעים את שדה ה-ttl של הפקטה, היא תשאר שם לזמן רב, וכך נשלוט על הדומיין עבור הלקוח למשך זמן רב. למעשה, היות שהתשאול הרקורסיבי מעמיס על ה-resolver, אנחנו ככל הנראה ננצח במתקפה. יחד עם זאת, אם הפסדנו, אנחנו נצטרך לחכות עד שה-ttl יעבור.

כדי למנוע מתקפות מסוג זה, בדומה ל-DHCP, התקשורת בין ה-Resolver לבין שרתי ה-Authoritative מאובטחת באמצעות שדה ה-transcation id (בקיצור, trxid), שהוא שדה בתוך ה-Header של פקטת ה-DNS, מיוצר על ידי ה-resolver וכשהוא מקבל תשובה מהשרתים, הוא מוודא שהיא מכילה את אותו trxid. שדה זה הוא בגודל 16 ביטים, ולכן ממצב של נצחון כמעט וודאי, ירדנו לנצחון בהסתברות של $\frac{1}{2^{16}}$ שזה כמעט אפסי.

> **!** מכאן נסיק כי צריך לוודא שה-trxid אכן מתפלג אחיד מבחינה חישובית. כמו כן, מתקפת bruteforce לא תעבוד כאן, שכן מספיק שהשרת ה-authoritative ענה לפנינו, מה שיקרה כמעט בוודאות, שכן אנחנו שולחים 2^{16} פקטות, ולא נוכל להמשיך במתקפה, אלא נצטרך להמתין לפקיעת ה-ttl. גם אם נתקוף בכל שאילתה דומיין אחר, זה בעייתי, היות שקודם כל נצטרך לחכות שהלקוח באמת ביקש אותם (אחרת ה-resolver לא יחכה לתשובה וידחה את הפקטה שלנו), וגם כי ככלה נראה אלא יהיו דומיינים לא מאוד נפוצים, היות שהדומיינים הנפוצים ככל הנראה כבר ב-cache, אז לא נשיג שליטה משמעותית.

5.4 המתקפה של קמינסקי

הבעיה המרכזית במתקפות שהצגנו היא שמספיק שהשיגו אותנו פעם אחת, ונצטרך לחכות הרבה זמן עד שנוכל לתקוף שוב. יחד עם זאת, הסתכלות זהירה במבנה של פקטת DNS והעברת המידע בין שרת Authoritative ל-Resolver, מניבה שיפור משמעותי, עליו Kaminsky ב-2008. כאשר השרת שולח לנו שרת ייעודי לאזור הדומיין שביקשנו, הוא שולח לנו שתי רשומות: NS, A. פקטת ה-DNS עצמה, כפי שציינו, מחולקת לשלושה sections, וכאשר השרת ה-Authoritative שולח תשובה, כל אחד מכיל מידע באופן הבא: ה-Query מכיל את הבקשה המקורית של הלקוח. ה-Answer מכיל רשומות NS מהשרת. אבל איפה נכנס ה-A רקורד? הוא נכנס ב-Additional Section, במטרה למנוע תלות מעגלית, וכדי לחסוך בתשאולים. כאשר ה-Resolver רואה את התשובה, הוא מסתכל על שרשרת ה-NS שנמצאת ב-Answer, ובודק אם השרת סיפק גם רקורד A עם ה-IP של השרת הרלוונטי. במידה שכן, הוא עושה לתשובה זו Cache בדיוק כמו כל A רקורד.

> **!** חשוב להדגיש, במקרה שהגיעה תשובת חדשה, שיכולה לדרוס רקורד קיים ב-Cache, ה-Resolver יקבל אותה אך ורק עם ה-ttl של הרקורד ב-cache פקע, אחרת הוא יזרוק את התשובה.
> כמו כן, השרת יכול לעשות cache ל-NS רקורדס.

על כן, נוכל במקום לתשאל כל פעם את www.google.com, דווקא את x1.google.com. כך, ונפנה ברשומת ה-NS ל-www.google.com, ונכלול ב-additional section רשומה www.google.com A 6.6.6.6, וכך נוכל להרעיל את השרת. אם נכשלנו, ננסה שוב, אבל הפעם נתשאל את x2.google.com. כלומר, עקפנו את מנגנון ה-ttl, היות שגם אם נכשלנו, השרת לא ישלח ל-resolver רשומת A עבור www.google.com כי היא לא רלוונטית, הוא מנסה למצוא את ה-IP של x1.google.com, וככל הנראה, יענה שאין בכלל IP לדומיין זה. מכאן, נוכל להמשיך בתהליך כאשר בכל פעם, אנחנו כוללים trxid שונה. עד שנצליח. חזרה על התהליך מספר מסויים של פעמים, ומתן תשובות עם trxid ראנדומי, תבטיח התנגשות, מפרדוקס יום ההולדת. נבהיר - בכל פעם נשלחת בקשה, ואנחנו "מפציצים" את ה-resolver בכמה שיותר תשובות עם trxid ראנדומי, ואנחנו פשוט חוזרים על הניסוי כמה שיותר פעמים.

נדגיש שנדרש כאן רכיב נוסף שישלח את הבקשות ל-Resolver. כמו כן, במקרה בו ה-trxid מתפלג אחיד, כל שאנחנו יכולים לעשות הוא מתקפות בסגנון של פרדוקס יום ההולדת, כלומר לשלוח הרבה בקשות, ועבור כל בקשה לשלוח הרבה תשובות עם trxid שונים, כל אחד מתפלג אחיד. בצורה זו, נקבל התנגשות בין התשובה שלנו לבקשה, וככה נצליח לנחש את ה-trxid.

יחד עם זאת, בפועל, ה-trxid לא מתפלג אחיד, אלא מיוצר על ידי יצרן פסאודו אקראי G, שבהנתן trxid s, מחשב את ה-trxid הבא על ידי $G(s)$. היות שאנחנו לא יכולים באמת ליצור ירפ"א, אלא רק ליצור הוריסטיקות, ייתכן ש-G פגיע, ויכול לצמצם מאוד את מרחב האפשרויות.

למשל, אם בהנתן ש-s זוגי, G יכול לייצר 10 ערכים אפשריים שניתנים לחיזוי בהנתן s, אנחנו יכולים לתקוף בהצלחה יתרה את ה-resolver על ידי בקשה של כתובת כלשהי, עד שמתקבל trxid זוגי, ואז ביצוע המתפקה של קמינסקי, כאשר בכל פעם אנחנו שולחים 10 פקטות עם 10 הערכים האפשריים ל-trxid הבא.

נבחין כי כל המתקפות שהצגנו הסתמכו על זה ש-2^{16} זה לא יותר מדי אפשרויות. לכן, הוספה של ראנדומיזציה בתקשורת, יכולה למנוע אפילו את המתקפה של קמינסקי. למשל, על ידי ראנדומיזציה של הפורט ממנו נשלחת הפקטה ב-resolver, למספר בן 16 ביטים נקבל מרחב אפשרויות של 2^{32}, brute force לא יעבוד. למרות זאת, מדובר בתרחיש של חתול ועכבר, ולרוב נמצאות חולשות חדשות, הכוללות שימוש במתקפות זמן ועוד.

אבטחה באמצעות DNSSEC

כדי להוסיף מנגנון אבטחה חזק יותר מה-trxid, נוכל להוסיף מנגנון אבטחה היררכי. בכל פעם שמישהו ישלח אלינו הודעה, הוא יוסיף חתימה מפונקציה קריפטוגרפית, בצורה היררכית. כלומר שרת השורש יוסיף את החתימה שלו, השרת com. יוסיף את שלו בצורה תלויה בחתימות הקודמות, וכן הלאה. כך, כאשר נקבל תשובה, אנחנו נבצע תשאול לכל אחד מהשרתים ונוודא שחתימות תואמות. אם כולם כוללים אותה חתימה, או שהתלות מתקיימת, נדע שזו הודעה אמינה (או שכל השרתים זדוניים, שזה תרחיש פחות סביר), אם יש סתירה, נדע שהייתה בעיה ונדחה את התשובה.

5.5 פרוטוקול IPv6

שאלה האם יש מספיק כתובות IP? הרצאה 9

יש לנו 32 ביטים לכל כתובת, ולכן סך הכל 2^{32} כתובות, ובדיונים קודמים, ראינו שהחלוקה היא לא פרופורציונית, כך שבפועל, מרחב הכתובות יחסית צפוף. לכן, בפועל, זה לא מספיק. הפרוטוקול שראינו עד כה נקרא IPv4 ואחריו הוצג פרוטוקול חדש, שנקרא ה-IPv6. בפרוטוקול זה כל כתובת מכילה 128 ביטים, שזה משמעותית יותר טוב. כמו כן ב-IPv6 פרגמנטציה לא מתאפשרת, בניגוד ל-IPv4.

יחד עם זאת, IPv4 עדיין בשימוש תדיר, היות שכדי לעבור ל-IPv6 צריך הסכמה של כל רכיבי הרשת, שידעו לעבור על הבתים הנכונים בפקטה בקריאת הכתובת.

בינתיים, הפתרון למחסור בכתובות הוא שימוש בטכנולוגיית NAT, עליה לא נרחיב במסגרת הקורס.

שאלה מדוע הרשת איננה רשת IP אחת גדולה, בלי switches?

תשובה אין סיבה טובה ללמה לא, הארכיטקטורה שראינו עובדת טוב, אך יש לה אלטרנטיבות.

5.6 Topologies

בכל הדיון שלנו על IP, עוד לא דיברנו על הרשת עצמה. כן הסתכלנו על IP מפרפסקטיבה של שכבה 2, כלומר איך נעזרים בה כדי לקבל כתובות. עתה נעסוק בטופולוגיה של הרשת, כלומר במבנה הפיסי שלה וכיצד נוכל לנתב מידע בין ישויות. אנחנו נסתכל על הרשת כגרף שכל קודקוד בו הוא נתב, שמחובר לרשת ממנה הוא מקבל מידע.

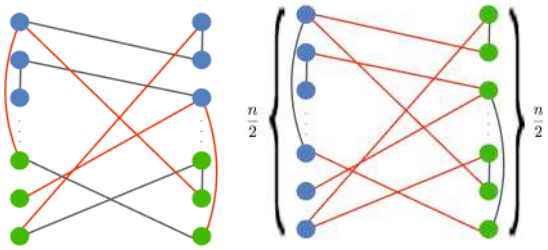
הגדרה 5.1. עבור גרף, נגדיר את המושגים הבאים:

1. קוטר של גרף הוא המרחק המקסימלי בין שני קודקודים בו.
2. דרגה של קודקוד היא מספר הקודקודים המחוברים אליו.

מושגים אלה קשורים לניתוב המידע. למשל, ככל שהדרגה גבוהה יותר, כך המרחק קצר יותר מיותר רכיבים. אם כך, עולה השאלה מהי רשת טובה? אנחנו מבינים שדרגה גבוהה עוזרת לניתוב, אך יש לה גם השלכות אחרות, כמו על כמות המידע שעובר ביחידה אחת ברשת. יתר על כן, עלול להווצר צוואר בקבוק, שיפגע בשימוש ברוחב הפס. כלומר כל המידע יגיע אל יחידה אחת, והיא תעביר אותה ליחידה הבא דרך אותו broadcast domain.
יחד עם זאת, עלינו להתחשב גם ברוחב הפס של הרשת. אפשר לחשוב על זה כרשת זרימה, בה אנחנו מוצאים זרימה מקסימלית מקודקוד מקור לקודקוד יעד. אמנם ברשת, אין לנו רק יעד אחד ומקור אחד, אלה מקורות ויעדים רבים. לכן, אנחנו נסתכל על זה כמקרה קיצון בו ברשת עם n קודקודים, $\frac{n}{2}$ קודקודים הם מקורות ו-$\frac{n}{2}$ קודקודים הם יעדים. כאן, כבר לא ברור איך למצוא זרימה מקסימלית. לכן, אנו מגדירים גודל זה להיות ה-

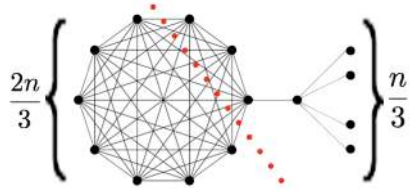
הגדרה 5.2. עבור גרף, נגדיר את ה-bisection bandwidth, להיות רוחב הפס המינימלי בין חצי מהקודקודים לחצי השני, כאשר עוברים על כל החלוקות האפשריות. כלומר

$$\min_{|T|=\frac{n}{2}=|S|,S\cap T=\emptyset} \text{bandwidth}\,(S,T)$$



איור 19: דוגמא ל-$\text{bandwidth}\,(S,T)$. בחלוקה השמאלית, נקבל 3, ובחלוקה הימנית נקבל 7 (נספור את הצלעות שיוצאות מקבוצה אחת לקבוצה השנייה). אך זה לא קובע את ה-bisection boundary היות שיש עוד הרבה חלוקות אפשריות.

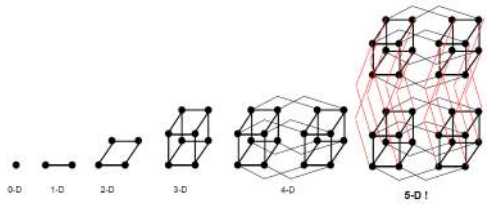
מבחינה חישובית, יש $\frac{1}{2}\binom{n}{n/2}$ (החצי בגלל הסימטריה) חלוקות אפשריות, ולכן לא יעיל לחשב אותו. אך זו לא הבעיה יחידה.



איור 20: בגרף הנ"ל $\frac{2}{3}$ מהקודקודים הם קליקה, ולכן עבור כל חלוקה, לפחות $\frac{1}{2}-\frac{1}{3}=\frac{1}{6}$ מהקודקודים יהיו חלק מהקליקה, ולכן נקבל בפועל bisection boundary גבוה, על אף שיש לנו צוואר בקבוק קריטי בין שני חלקי הגרף.

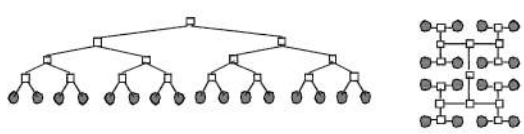
על כן, במקום להסתכל על ה-bisection boundary בלבד נוכל להסתכל על גודל נוסף. בפרט, נרצה להסתכל על ממוצע הצלעות שיוצאות מקבוצה מסויימת, בגודל לכל היותר $\frac{n}{2}$, ונבחר את המינימום. כך נקבל את צוואר הבקבוק. כלומר, $\min_{S\subseteq V, 0<|S|\leq\frac{n}{2}} \frac{\text{EdgesBetween}(S,V\setminus S)}{|S|}$. לזה אנו גם קוראים Expansion של הגרף. ככל שהוא יותר גדול, כך הגרף יותר טוב מבחינת רוחב פס.

> **!** עלינו להבחין בין שני סוגים של גרפים - כאלה שהבנייה שלהם מפורשת, וכאלה שלא. בגרפים מפורשים, יש לנו הגדרה פורמלית של הקודקודים והצלעות. בגרפים לא מפורשים, כדוגמת האינטרנט, אין לנו הגדרה מפורשת, וזה מאפשר הרחבה של הגרף בצורה פשוטה.



איור 21: נביט בקובייה ה-n-מימדית. כגרף מפורש, יש לה 2^n קודקודים, כאשר כל קודקוד מתואר על ידי ווקטור ב-$\{0,1\}^n$, ויש צלע בין שני קודקודים אם"ם מפריד ביניהם שינוי בקואורדינטה אחת בלבד. על כן, הקוטר של הגרף הוא n, שכן לכל היותר נבצע n שינויים לקואורדינטות, מבלי חזרות. מהו ה-bisection boundary? נבחר חלוקה ספציפית בשביל אינטואיציה. נבצע חלוקה לפי הקאורדינטה הראשונה $1,0$. אזי נשארו 2^{n-1} קודקודים בכל קבוצה ולכן זו חלוקה. כמו כן, יש צלע בין שני קודקודים, אם"ם הם זהים ב-$n-1$ הקואורדינטות הנותרות, שכן הקאורדינטה ה-n כבר מפרידה ביניהם, לכן נקבל 2^{n-1} צלעות. בפרט, זה גם המינימום.

למעשה הקוביה ה-n-מימדית מאוד נוחה לניתוב, היות שכל שצריך לעשות הוא למצוא את המרחק המינימלי לעריכת רצף אחד של ביטים לרצף אחר.

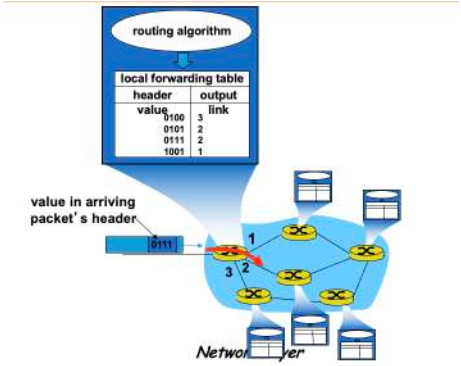


איור 22: דוגמא נוספת היא עץ בעל דרגה קבועה. כאן נוכל לחלק את הקודקודים על ידי העץ הימני והשמאלי מהשורש, ואז נקבל שיש רק צלע אחת בין הקבוצות ולכן נקבל $\mathcal{O}(1)$ bisection boundary. בפרט, הקוטר הוא פעמיים גובה העץ, $2\log N$.

אלה דוגמאות מובהקות לגרף מפורש. הרשת לעומת זאת, לא מפורשת, ואין לנו תיאור פורמלי כמו כאן, שמאפשר חישוב מרחקים מבלי להסתכל בכלל על הטופולוגיה שלה. במקום זאת, יש לנו אלגוריתמי ניתוב.

5.7 Routing

עכשיו כשאנחנו מבינים איך הגרף מתנהג, נוכל ליצור מסלולים בין נתבים. משימה זו מתחלקת לשתי תתי משימות. מציאת מסלולים ב-intradomain, כלומר ברשת פנימית, ומציאת מסלולים בין $inter - domains$, כלומר בין אזורים שונים. המסלול, יקבע על פי forwarding tables, כלומר שבהנתן פקטה ו-header הנתב יידע דרך איזה פורט להעביר אותה:



איור 23: המחשה לטבלאות. בפרט המשימה של מציאת המסלולים הופכת לבעיה של מילוי הטבלאות.

אנחנו נניח שכל צלע המחברת נתבים, בעלת משקל חיובי כלשהו, ולכן עלינו למצוא מסלולים טובים ביותר בגרף בצורה מבוזרת. מסלול טוב ביותר לבד איננו מספיק, היות שהרשת עלולה להשתנות וכך גם המשקלים. למשימה זו יש שני אלגוריתמים:

1. Distance Vector Routing.
   (א) זהו אלגוריתם decentralized, כלומר נתבים מודעים רק לנתבים השכנים שלהם.
   (ב) במקרה זה נדרשת התכנסות של כלל הרשת.
2. Linked State Routing.
   (א) זהו אלגוריתם גלובלי, כלומר נתבים מקבלים מידע מלא על הטופולוגיה של הרשת.
3. בשני האלגוריתמים יש לנו מנהל רשת שקובע משקלים.

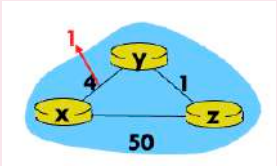
אלגוריתם ה-Distance Vector

אלגוריתם ה-distance vector, בדומה ל-SP של שכבה 2, עושה שימוש בבלמן פורד מבוזר, לפי הכלל

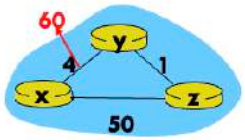
$$d_x(y) = \min_{v \in x.\text{neighbors}} \{c(x,v) + d_v(y)\}$$

היות שנדרש מרחק מינימלי מכלל הקודקודים ברשת, אנחנו מקבלים ווקטור d_x שמכיל את המרחק המינימלי מכל אחד מהקודקודים. בצורה זו, המידע מפועפע על ידי השכנים.

Count to Infinity: מקרה מבחן של אלגוריתם ה-Distance Vector



איור 24: נניח שהרשת התכנסה. לפתע השתנה המשקל בין x ל-y להיות 1 במקום 4.
לכן, y יראה שהוא יכול להגיע ל-x ב-1 במקום 4. z, יסתכל על y, שיצהיר שהוא רחוק מ-x ב-1, ולכן ימשיך לפנות ל-y בהעברת מידע ל-x ואין בעיה.



איור 25: נניח שוב שהרשת התכנסה, אך המשקל בין x ל-y השתנה ל-60.
עתה, y מסתכל על z, ורואה שהמרחק של z מ-x הוא 5, כי בטבלא המקורית, z הסתכל על y וראה 4, ולכן חשב שהמרחק שלו מ-x הוא 5. אך עכשיו, z לא יודע על השינוי במשקל של x, y, ולכן עדיין חושב שהמרחק שלו הוא 5. לכן y, חושב שהמרחק שלו מ-z הוא $5 + d_y(z) = 6$. כלומר הוא מקבל מידע "שקרי", שולח את ההודעה ל-z, שחושב שהמסלול עובר דרך y, ולכן מחזיר את הפקטה ל-y ואז הם חוזרים על התהליך, עד שהמרחק ש-z רואה מ-y, גדול יותר מ-50, ואז z יצהיר שהמרחק שלו מ-x הוא 50 ו-y יבחר ללכת דרך z ו-z יעביר את ההודעה דרך הצלע שלו ל-x.
לבעיה זו אנו קוראים Count to Infinity, היות שככל שמשקל הצלע גדל, כך נדרשות יותר איטרציות להתכנסות.

!

הבעיה המרכזית היא ש-x, y, z לא רואים את הטופולוגיה של הרשת ב-DV, ולכן הם מסתמכים על המידע "היבש" שרשום בטבלאות, מה שמביא לתופעה הבעייתית הנ"ל. בפרט מדוגמאות אלה אנחנו מסיקים כי חדשות טובות מפעפעות מהר, אבל חדשות רעות מפעפעות לאט כשמשתמשים ב-DV.
Poisoned Reverse: נוכל לנסות לפתור את הבעיה על ידי כך שלא נספר לקודקודים שאנחנו עוברים דרכם את המסלול שלנו, ואז הבעיה שראינו בדוגמא הקודמת תפתר. אך יצוצו בעיות אחרות, למשל כאלה שמערבות לולאות ארוכות ומורכבות יותר.

ב-Linked State זה לא היה קורה, היות שהמידע מועבר בצורה גלובלית ולכן z היה יודע על השינוי במשקל הצלע $\{x, y\}$.
יתר על כן, Linked State יודע הכל על הרשת, את המשקלים, הצלעות בין הנתבים, הוא גלובלי לחלוטין.

סיבוכיות

1. **סיבוכיות שליחת ההודעות**: עבור n קודקודים ו-E חיבורים, נקבל עבור LS הודעות שנשלחות $\mathcal{O}(nE)$. עבור DV רכיבים מידע אחד לשני - ההתכנסות תלויה בזמן.
2. **מהירות ההתכנסות**: LS בעל סיבוכיות $\mathcal{O}(n^2)$ ודורש $\mathcal{O}(nE)$ הודעות לשליחה. DV עלול להכיל לולאות, ובעיות $\text{Count} - \text{To} - \text{Infinity}$.

3. **רובסטיות**: LS עלול להפיץ מרחקים לא נכונים, שכן כל קודקוד מחשב את הטבלא של עצמו בלבד. DV גם יכול להפיץ מרחקים לא נכונים, והטבלאות של כל רכיב תלויות בטבלאות של הרכיבים האחרים.

RIP, OSPF

אחד האלגוריתמים לניהול רשת נקרא RIP, והוא משתמש באלגוריתם ה-DV לניתוב. אלגוריתם מקביל, נקרא Open Shortest Path First Protocol (OSPF) שמשתמש ב-LS.
במקור, המשקולות של הצלעות בין הקודקודים, עודכנו באופן דינאמי, על ידי הקודקודים בצורה מחזורית. בצורה זו, מקומות ברשת שהיו "עמוסים" יכלו לעדכן את המשקל להיות כבד יותר, וקח להקל על הרשת, בפרט הדרך בה חישבו את המשקל, היא באמצעות תור, כך שהמשקל היה טרנספורמציה לינארית של כמה מלא התור.
אמנם יש לכך השלכה חישובית, בכל מצב בו יש עומס, צריך לעדכן את כל הרשת שוב, ונוצרות אסוליציות ברשת, כל פעם פונים למסלול הטוב יותר, ולכן המשקל שלו משתנה. מלבד זאת, נראה שמסלולים ארוכים יותר עדיפים על מסלולים קצרים, היות שמה שקובע הוא המשקל - זה בעייתי כי זה אומר שנתחרה עם פקטות אחרות על רוחב פס לאורך מסלול ארוך.
לשינוי המשקולות קוראים גם סיפור ה-ARPAnet.

Traffic Managment

מהדיון הקודם אנחנו מבינים שצריך לקבוע משקולות סטטיות, ובמקרה של עומס, מנהל הרשת יוכל לשנות את המשקולות, הרצאה 10
אך הוא לא יראה בזמן אמת כמה תעבורה עוברת בכל מסלול, אלא רק יסתכל על סטטיסטיקות ארוכות טווח. דבר נוסף שנעשה, הוא לשלוט על התעבורה, אבל שכבה מעלינו, כלומר נשלוט על קצב התעבורה, ומי שישנה זאת, מתייחס למסלול כקבוע, אין לו השפעה על המשקולות. הסיבה שאנחנו מעבירים משימה זו לשכבה הבאה, היא שכאשר ניסינו לשלוט על התעבורה בשכבת הרשת, ראינו שהדרך לעשות זאת היא באמצעות משקולות, מה שהוביל לאי יציבות של הרשת.

5.8 זרימה אופטימלית ברשת

אחד הדברים ששמנו לב אליהם, הוא שלא מספיק למצוא מסלולים קצרים ברשת, אלא שצריך למצוא גם זרימה.

תזכורת | $\text{Max}-\text{Flow}$

קלט גרף לא מכוון עם פונקציית קיבולות $G = \langle V, E, c\rangle$. קודקוד מקור s וקודקוד יעד t.

פלט זרימה מקסימלי ברשת מ-s ל-t.

בעיה-דואלית חתך מינימלי ברשת (צוואר בקבוק) = זרימה מקסימלית ברשת.

ברשת, כפי שתארנו, הבעיה היא אחרת, יש לנו קבוצה של מקורות ויעדים. לכן, אנחנו מגדירים בעיה אחרת:

הגדרה 5.3. $(\text{Multicommodity}-\text{Flow})$ נקבל קלט גרף לא מכוון עם פונקציות קיבולות $G = \langle V, E, c\rangle$. בנוסף, נקבל מטריצת דרישות $D = \{d_{ij}\}$ כך ש-d_{ij} אומר מה הזרימה הדרושה בין הקודקוד ה-i לקודקוד ה-j. הפלט, הוא זרימה f שמקרבת את D ככל הניתן.

בבעיה זו נמקסם את $\sum_{v\in\text{sources}} |f_v|$. אמנם, אם $\text{MaxFlow} = \text{MinCut}$, אז למה שווה $\text{Max}-\text{Multicommodity}-\text{Flow}$? במקרה זה אין לנו אנלוג ל-$\text{Min}-\text{Cut}$, ולכן אנחנו מגדירים בעיה אחרת, שקל יותר להבין. במקום למקסם זרימה, נמזער את העומס על הצלע הכי עמוסה.

הגדרה 5.4. (Minimize Congestion) נמזער את $\max_e \frac{f_e}{c_e}$ כאשר f_e הזרימה לאורך הצלע e, כך שנענה על הדרישות של המטריצה D. כלומר, אנחנו מגדירים זרימה טובה, כאחת בה הצלעות לא עמוסות יתר על המידה.

בבעיה זו אנחנו עלולים לחרוג מהקיבולות, בגלל שאנחנו רוצים לענות על הדרישות. יחד עם זאת, אלה הגדרות משנות ה-90. המטרה הייתה קודם כל לפתור את הבעיה, ולאחר מכן להוסיף קיבולות במידת הצורך, על ידי הוספת כפל אינטרנט פיסי. לכן, אנחנו פותרים את הבעיה בהנחה שיש לנו מספיק קיבולת.
כמו כן, זו בעיית Linear Programming, ולכן ניתנת לפתרון בזמן פולינומיאלי.

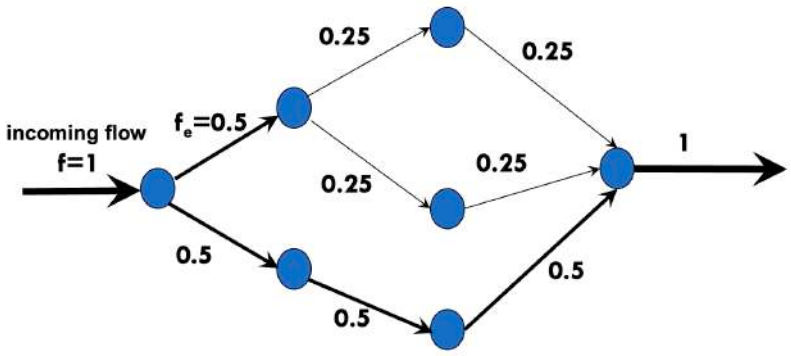
ראינו שבעיה זו ממזערת צלעות עמוסות, וזה מוביל לחלוקה של התעבורה על מסלולים שונים. למשל אם ל-router יש כמה nexthops, אז סביר שהוא יחלק את התעבורה ביניהם.
יחד עם זאת, צריך לבדוק כיצד בעיה זו פוגשת את המציאות. ראינו ש-$IP-Routing$ מוצא מסלולים אופטימליים, אך לא ברור אין הוא מחלק תעבורה בין מסלולים. בפרט, לא מספיק לתת לרשת עם מסלולים אופטימליים, זרימה אופטימלית של $Minimize-Congestion$, שכן אנחנו עלולים לקבל חלוקה למספר אקספוננציאלי של מסלולים, מה שיעמיס על הנתב.

הרשת בסך הכל מחשבת מסלולים אופטימליים בגרף, היא לא מסתכלת על התעבורה עצמה. הדיון על Multicommodity Flow הוא תאורטי, ואנחנו צריכים לנסות להכניס מתוכו אל תוך המשקולות. כלומר אנחנו רק מקרבים את הבעיה.

על כן, נבחר את המשקולות ביחס להפוך לקיבולת, כלומר $\frac{1}{c_e}$, כך המסלולים יתאזנו על צלעות עם הרבה קיבולת.

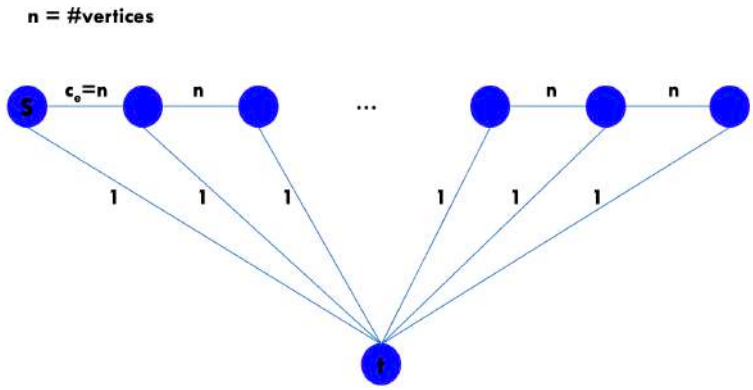
5.9 Equal Cost Multipath (ECMP)

כדי לאפשר חלוקה של הזרימה על גבי מסלולים שונים, נשתמש במנגנון ה-ECMP, בו נחלק תעבורה בין Nexthops. כלומר בהנתן יחידת מידע בגודל x, נחלק אותה ל-$\frac{x}{n}$ יחידות לפי ה-nexthops. כל רכיב שיקבל את היחידה, יחלק אותה ל-$\frac{x}{n \cdot m}$ יחידות בין m ה-nexthops שלו. נדגיש שהיא מפוצלת על פני nexthops באופן שווה, ולא על פני מסלולים.



איור 26: המחשה ל-ECMP. נדגיש שבוחרים ב-NextHops שנמצאו בהרצת אלגוריתמי הניתוב. כלומר אלה NextHops שהם חלק ממסלול אופטימלי ליעד.

מכאן אנחנו מקבלים בעיית אופטימיזציה חדשה, בה אנחנו צריכים למצוא משקולות אופטימליות ברשת, כך שהזרימה של ECMP היא אופטימלית, שכן אנחנו רוצים ש-ECMP יהיה אופטימלי. אבל עולה השאלה, האם תמיד יש משקולות עבורן ECMP הוא אופטימלי? האם יש משקולות שקרובות לפתרון האופטימלי?



איור 27: נניח שאנחנו רוצים לשלוח פקטה בגודל n יחידות מידע. אזי הזרימה האופטימלית תהיה לשלוח 1 ל-t, ואז $n-1$ לקודקוד הבא בשרשרת. בצורה זו נקבל $\text{opt} \leq 1$.
אבל אם נדרוש אילוצים של ECMP, לכל בחירה של משקולות נקבל $\text{opt} \geq \frac{n}{2}$. שכן, אם t הוא ה-$\text{next}-\text{hop}$ היחיד, אז נשלח n יחידות מידע ונקבל $\text{opt} = n$. אחרת, גם הקודקוד הבא בשרשרת הוא $\text{next}-\text{hop}$ ולכן נעביר $\frac{n}{2}$ דרך שניהם, ונקבל $\text{opt} \geq \frac{n}{2}$. אחרת, הכל עובר דרך הקודקוד הבא בשרשרת, ולכן נעביר דרכו n, ואז הבעיה חוזרת על עצמה עם הקודקוד הבא. כלומר בכל מקרה נקבל $\text{opt} \geq \frac{n}{2}$.

מכאן אנחנו מסיקים כי ECMP מביא לכך שהזרימה שנקבל היא לא אופטימלית. לכן נוכל לבנות בעיה חדשה, בה נדרוש משקולות שהכי קרובות לפתרון האופטימלי.

זו בעיה NP-קשה. יותר מזה, כל אלגוריתם מקרב עם קירוב c קבוע, הוא גם NP-קשה, אפילו במקרה של מקור ויעד יחידים!

לכן, מה שעושים בפועל הוא שימוש בהוריסטיקות. כלומר, נתחיל ממשקולות של $\frac{1}{c}$ ונעדכן אותם על ידי חיפוש לוקאלי. זה לא אופטימלי, אבל כן יעיל יותר, ובעל ביצועים טובים יחסית לפתרון האופטימלי. מעבר לכך, זה אמיד בפני כשלונות של הרשת.

ECMP במציאות

בתאוריה, התייחסנו למידע כיחידות מידע קטנות יותר שאפשר לפצל, והזנחנו דבר קריטי - חווית המשתמש. נניח שאנחנו עושים שיחת ZOOM, ופקטת המידע התפצלה בין מסלולים שונים - כל אחד יגיע בזמן שונה, לא בהכרח לפי הסדר, ולכן נחווה תקשורת ברמה ירודה. לכן, כדאי לנתב חלקים של פקטה אחת לאותו מסלול. הדרך שבה נעשה זאת הוא באמצעות Hashing.
לחלקים של אותה פקטה יש אינוריאנטה והיא שה-Header שלהם זהה. לכן, נוכל למפות חלקים למסלולים לפי ה-Header. יחד עם זאת, המיפוי צריך להיות יעיל וכמה שיותר uniform, לכן נשתמש בפונקציית Hash, על ה-Header, כך נמפה חלק ל-nexthop הבא, בצורה דטרמינסטית, ובמחיר נמוך.
בצורה זו מידע של אותה פקטה יעבור באותו מסלול. אך יש בעיה - במקרה בו מידע אם אותו header נשלח בלבד, ננצל רק מסלול אחד, וזה רע. אך זה מקרה קצה, היות שבפועל הרבה מאוד מרכיבים שונים עובר ברשת.

בזאת סיימנו את הדיון על מציאת מסלולים וחלוקת עומסים. עכשיו צריך לדון על שליטה בתעבורה - מה שקורה ב-Layer 4, בפרט סיימנו את הדיון על שכבה 3.

6 שכבת התעבורה (Transport Layer)

אפשר לחשוב על שכבה זו, כרשת overlay מעל שכבת הרשת. כאן, היישויות ברשת הן **תהליכים במחשב**, שמתקשרים אחד עם השני. בפרט, ברשת זו נלמד כיצד לספק מידע אמין באמצעות פרוטוקול TCP, עם שליטה על קצב התעבורה, כפי שהזכרנו בשכבה הקודמת. נלמד גם על התאום שלו -UDP שהוא מינימלי מבחינת אמינות.
כמו כן, ברשת זו, הישויות הן ישויות קצה $(\text{End} - \text{To} - \text{End})$, כלומר הן לא מודעות למה שקורה בדרך להעברת המידע, אלא רק להוצאת המידע בקצב מסויים. מבחינתן שכבת הרשת היא כמו ענן שאפשר להעביר אליו את המידע והוא יעביר אותו הלאה ליישות הרצויה.
בנוסף, השכבה מספקת תקשורת לוגית בין ישויות, למשל באמצעות sockets.

הרצאה 11 משימה שהשכבה מבצעת היא העברה אמינה של מידע, וכפי שהזכרנו בעבר congestion control (למשל שליחה מהירה יותר של מידע). בנוסף, היא אחראית על שינוי סדר של פקטות, שזה נכנס תחת העברה אמינה של מידע - כל זה קורה בקצוות, על אף שזה היה יעיל יותר לעקוב אחר הפקטות בין הקצוות, אנחנו רוצים לשמור על עקרון ה-$\text{end} - \text{to} - \text{end}$, וגם, יש אפליקציות שלא צריכות את כל התוספות של שכבה זו, לכן זה עול חישובי.

באופן כללי, כאשר אנחנו מקבלים מידע מלמעלה, אנחנו מעבירים למטה, אבל כשאנחנו מקבלים מידע מלמטה, אנחנו צריכים לדעת למי להעביר אותו למעלה. בשכבה זו, אנחנו יודעים מי נמצא מעלינו על ידי מזהה שלו אנחנו קוראים port. זה מזהה שמתווסף לכתובות ה-mac, ip שאנחנו מכירים, כלומר עכשיו צריך ה-socket שלנו מזוהה על ידי ה-ip וה-port. לכן אנחנו יכולים להעביר מידע לתהליכים שונים באותו מחשב.

6.1 פרוטוקול Transmission Control Protocol (TCP)

כאשר אנחנו פותחים socket בפרוטוקול זה, אנחנו צריכים לספק את ה-$(\text{ip\_src}, \text{port\_src}, \text{ip\_dest}, \text{port\_dest})$ כלומר הוא מאפשר תקשורת בין שני תהליכים בלבד. על כן אם יש לנו אפליקציה שמקבלת תעבורה מהרבה רכיבים עם כתובות שונות, נצטרך לפתוח הרבה sockets עם מזהי יעד שונים. בניגוד ל-udp שם היינו צריכים רק socket אחד. על כן, פרוטוקול זה מורכב יותר מ-UDP היות שאנחנו צריכים לתחזק "שיחה". לפני כן צריך לבסס אותה, ולוודא שהמידע מועבר בצורה סדרתית, כלומר שנדע מה מספר הפקטה.
ביסוס השיחה יתבצע באופן הבא, שנקרא גם Three Way Handshake:

1. הלקוח שולח פקטת SYN לשרת - נספק את מספר הפקטה ההתחלתי שהוא שולח ליעד (seq). אין מידע מעבר לזה.
2. השרת מקבל את ה-SYN, ושולח פקטת $\text{SYN} - \text{ACK}$ - מקצה מקום לשיחה, ושולח את מספר הפקטה ההתחלתי שלו עבור הלקוח (seq). ה-ack שהוא שולח זה ה-seq שהוא קיבל מהלקוח $+1$.
3. הלקוח מקבל את ה-$\text{SYN} - \text{ACK}$ ושולח פקטת ACK - היא תכלול מידע.

עכשיו אפשר להעביר מידע, כאשר בכל שלב, אנחנו נעדכן את ה-seq שלנו לפי מה שהצד השני מצפה לקבל.

דוגמה 6.1. נניח שעשינו three way handshake והתחלנו עם $\text{seq} = 42, \text{ack} = 79$. צד אחד שלח את הבית c, אזי הצד השני יחזיר לו בהודעה $\text{seq} = 79, \text{ack} = 43, \text{data} = \text{"d"}$ ואז אנחנו נענה ב-$\text{seq} = 43, \text{ack} = 80$. כלומר אנחנו שולחים seq לפי הבתים שאנחנו שולחים, ו-ack לפי הבתים של הצד השני. כלומר סופרים לפי מספר הבתים ולא לפי מספר הפקטות.

עלינו להסביר כיצד נסיים שיחה, באופן הבא:

1. הלקוח שולח הודעת $\text{TCP} - \text{FIN}$ שמודיעה לשרת שהשיחה נגמרה.
2. השרת מקבל ושולח ACK ללקוח ומשחרר את המשאבים.

נדגיש שבמצב זה ערוץ המידע הוא דו-כיווני תמיד, פשוט יכול להיות שאחד הצדדים מעביר מידע שקשור רק לפרוטוקול. כמו כן, כאשר האפליקציה רוצה לשלוח מידע לאפליקציה אחרת, היא שולחת מידע ל-buffer שזמין לשכבת התעבורה, ואז נעביר אותו לפי פרוטוקול tcp.

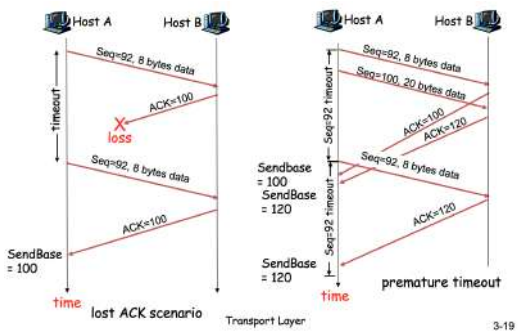
נוכל לסכם את שני צדדי הערוץ:

TCP – Sender/Receiver

השולח

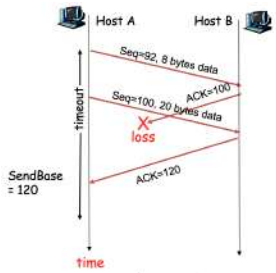
1. ניצור סגמנט (פקטה בשכבה 4) עם seq מתאים, שהוא הבית הראשון של המידע בסגמנט.
2. אם אין טיימור שרץ כבר, נפתח טיימר חדש. אם יש טיימר הוא של הפקטה הכי ותיקה שעוד לא קיבלה ack.
3. אם עבר TimeOutInterval נשלח את הסגמנט שוב, ונאתחל מחדש את הטיימר. יש לנו טיימר אחד.
4. אם הגיע ack לסגמנט קודם שפתחנו לו טיימר, נתחיל טיימר חדש אם יש סגמנטים שממתינים. נעדכן שהגיע ack.
5. **בעולם אידאלי** היינו מאתחלים את הטיימר לכלול את הזמן שכבר המתנו להגעת פקטות חדשות, אך יש לנו טיימר אחד, ולכן אנחנו פשוט מאתחלים אותו שוב. זו הדרך שלנו לתת פתרון מקורב.

המחשה



איור 28: משמאל תחילה נפלה ה-ack, לכן היה timeout ושלחנו את הסגמנט שוב.
מימין, ה-ack הגיעו אחרי הזמן, ולכן שלחנו את הסגמנט הראשון שוב. אך היעד קיבל כבר את הפקטות, ולכן הוא שולח ack שכולל כבר את שתיהן, ולא רק אחת.

כמו כן ייתכן ש-ack פשוט נפל בדרך והמשכנו לשלוח סגמנטים:



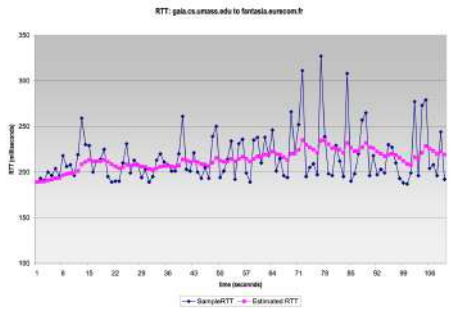
איור 29: על אף שהוא נפל, כשהוא קיבל עוד מידע, הוא שלח ack עד הבית האחרון שהוא קיבל מידע לא "שבור".

6.1.1 Round Trip Time (RTT)

כיצד נקבע את הזמן ל-timeout? נבחין כי כמו בפרוטוקולים קודמים, הוא צריך להיות לפחות הזמן שלוקח לשלוח פקטה, ולקבל ack. לזמן זה אנו קוראים RTT, הוא בפרט כולל בתוכו את $2T_{prop} + T_{ack} + T_{process}$, אך גם עוד גורמים.
למשל, אם הגדרנו timeout קטן יותר, נשלח כל פקטה הרבה מאוד פעמים, כשלא באמת צריך.
על כן צריך לדעת מהו ה-RTT. אנחנו מודדים אותו על ידי מדידת הזמן משליחת פקטה עד לקבלת תשובה, בהתעלם משידורים חוזרים. היות ששערוך זה משתנה, אנחנו ניקח ממוצע משוקלל בכל שלב כדי לקבל ערך פחות רועש

$$\text{EstimatedRTT} = (1-\alpha)\cdot\text{EstimatedRTT} + \alpha\cdot\text{SampleRTT}$$

בדרך כלל נבחר $\alpha = 0.125$, שכן נעדיף לתת לזמן הנמדד חשיבות, אך לא יותר מדי, שכן אם הייתה קפיצה, זה לא בהכרח על ההמשך. כמו כן, זה נותן דעיכה אקספוננציאדית של מה שקרה בעבר.



איור 30: ה-EstimatedRTT בהשוואה ל-SampleRTT

נבחין כי השונות משתנה, וצריך לקבוע את ה-timeout בהתחשב בה. לכן נוכל לשערך שונות:

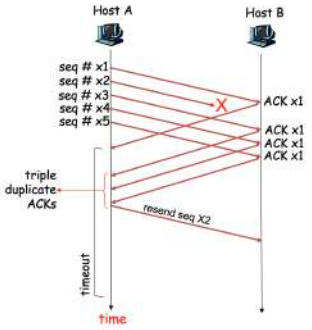
$$\text{DevRTT} = \beta\cdot\text{DevRTT} + (1-\beta)\cdot|\text{EstimatedRTT} - \text{SampleRTT}|$$
$$\beta = 0.25$$

ואז נוכל לקבוע את ה-timeout להיות:

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4\cdot\text{DevRTT}$$

מה שיאפשר התחשבות בעומסים ברשת.

היות ש-timeout יחסית ארוך, יכול להיות מצב שבו סגמנט אחד נפל, והמשכנו לשלוח סגמנטים, ואז קיבלנו המון ack כפולים שמעידים על נפילת המידע. על כן, במקום לחכות ל-timeout נוכל להניח שכל המידע אחרי ה-ack נאבד, וצריך לשלוח אותו שוב.



איור 31: FastRetransmit

6.2 פרוטוקול UDP

כאשר אנחנו פותחים socket בפרוטוקול זה, אנחנו מספקים רק כתובת יעד ופורט, כתובת המקור לא מסופקת! לכן כשנקבל את המידע לא נוכל לזהות מאיפה הוא הגיע, אלא אם נסתכל על מידע משכבות נמוכות יותר.
אין לנו כאן מושג של "שיחה", הוא connectionless. עולה השאלה, למה צריך אותו? נבחין כי ל-tcp יש תקורה משמעותית, ולכן ייתכן שנעדיף שימוש ב-UDP, למשל בשביל זמן ריצה, כדי שפקטות לא ישלחו שוב, כי זה עלול לעכב אותו. למשל אם נשלח שלוש פקטות זהות רק כי ביט אחד נפגע, זה מעכב מאוד. על כן, במקום לתת ל-tcp בנפילה של פקטות, אפשר לומר לאפליקציה לשלוח את הבקשה שוב לאחר timeout מסויים.

! יחד עם זאת, אלה רק סיבות משניות! אם היינו מחליפים כל שימוש ב-udp ב-tcp, האינטרנט לא היה נהיה פחות יעיל, אלא לא היה עובד בכלל!
למשל, לא היינו מסוגלים לתת בכלל כתובות IP, שכן ה-dhcp היה רוצה להעביר ב-broadcast מידע, ומובן הזה לא צריך כתובת יעד, אבל מה עם המקור? על כן אנחנו מסיקים ש-UDP הוא גם מנגנון ל-Bootstraping, שמאפשר ייצור זהויות ברשת.

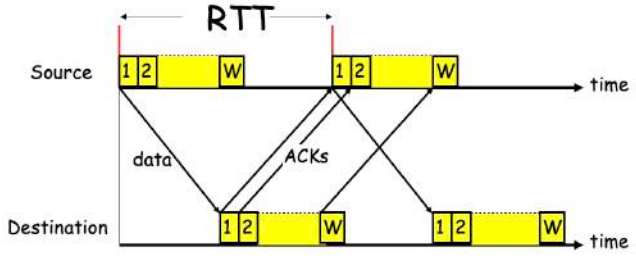
6.3 ניהול עומסים - Congestion Control

הרצאה 12

בניהול עומסים יש לנו שתי בעיות:

- $Flow - Control$ - שלא נשלח פקטות יותר מהר ממה שהיעד יכול לקבל. זו בעיית $End - To - End$ והיא פתורה.
- $Congestion - Control$ - שלא נשלח פקטות יותר מדי מהר, כך שהרווטור הקרוב לא יוכל להכיל אותם ב-buffer שלו ויזרוק אותן. בשכבה זו אנחנו מתייחסים לכל המשקלים והמסלולים כקבועים, ולכן זו בעיה קשה, היות שאין לנו שליטה על המסלול, אלא רק על מהירות שליחת המידע.

UDP לא מבצע Congestion Control. TCP מבצע זאת באמצעות $Window - Based - Protocol$, לפי נשלח פקטות רק בתוך חלון זמן מוגדר, אותו נעדכן בזמן ריצה:



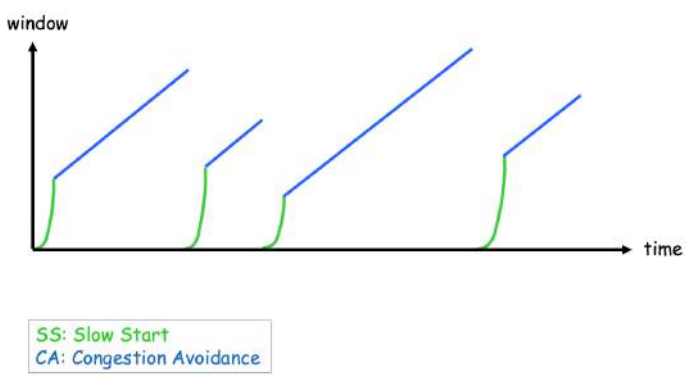
איור 32: המקור שולח פקטות בחלון W, ומצפה לתשובה. הוא לא שולח פקטות לאחר מכן, עד שקיבל תשובה לכל הפקטות בחלון. בפרט נקבל ש-$\text{SourceRate} = \frac{W \times \text{MSS}}{\text{RTT}}$bps כאשר $\text{MSS} = \text{Maximum Segment Size}$. בפרט $\text{MSS} < \text{MTU} = \text{Maximal Transfer Unit}$, שכן צריך להוסיף עוד מידע לפקטה מלבד ה-$\text{TCP} - \text{Segment}$, כמו headers למשל.
גודל החלון הוא ביחידות של MSS, אך כשנרשום $W = W + c$ הכוונה ל-$W = W + c \cdot \text{MSS}$.

עולה השאלה, כיצד לבחור את W, בזה יש הרבה גרסאות של TCP. באופן כללי יש שני חלקים, Slow Start בו אנחנו מתחילים להגדיל את החלון במהירות, ו-Congestion Avoidance בו אנחנו מעדכנים את החלון לפי העומס. כדי לקבוע מה עושים, יש לנו ssthresh שלפיו נקבע מתי לעבור בין החלקים.

6.3.1 TCP Tahoe

בגרסא זו שיצאה ב-1988 (כי ב-1986 הביקוש באינטרנט עלה על ההיצע), אנחנו פועלים באופן הבא:

1. Slow Start: כל פעם שקיבלנו ack נעדכן $W \leftarrow W + 1$, כדי לקבל עלייה אקספוננציאלית.
   (א) כלומר כל פעם שהגיעו ack לכל הפקטות בחלון קיבלנו $W \leftarrow 2W$.
   (ב) נכנס ל-CA כאשר $W \geq \text{ssthresh}$.
2. CA: בכל פעם שקיבלנו ack נעדכן $W \leftarrow W + \frac{1}{W}$ וכך נקבל עלייה לינארית, $W \leftarrow W + 1$ בכל סוף חלון. העדכון הוא רק בסוף החלון, כלומר זה לא שב-ack השני נוסיף $\frac{1}{W+1}$.
   (א) אם זיהינו שנפלה פקטה, נחזור ל-Slow Start אבל הפעם עם $\text{ssthresh} = \frac{W}{2}$.

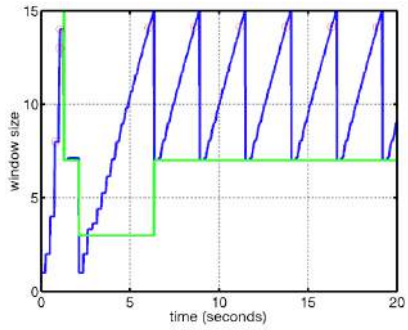


איור 33: לכן ב-SS יש עליה אקספוננציאלית, וב-CA עלייה לינארית. בכל נפילה, אנחנו מבצעים שוב SS עד לחצי מאיפה שעצרנו פעם קודמת.

ראינו כי יש שני מקרים של אבדן פקטה - הראשון הוא שהיה timeout והשני הוא שהמשכנו לשלוח פקטות, וקיבלנו ack כפולים חזרה. במקרה הראשון נרצה באמת לחזור ל-SS כי כנראה יש עומס ברשת. במקרה השני, נעדיף להקטין קצת את גודל החלון אבל לא להתחיל הכל מחדש, ולכן tcp tahoe לא מספיק טוב, כי הוא לא מבחין בין המקרים.

6.3.2 TCP Reno

בגרסא זו אנחנו מתחשבים בשני המקרים שציינו קודם, וכאשר אנחנו מזהים 3 ack כפולים אנחנו מעדכים $W \leftarrow \frac{W}{2}$. כך מתקבל הדבר הבא:



איור 34: גרף "המסור" של TCP Reno. במסור ניתן לראות שאין timeout, אלא רק ack כפולים, וזה קורה כי החלון מביא לכך שאנחנו עוברים את ה-bandwidth, ולכן בכל פעם מקטינים את החלון.

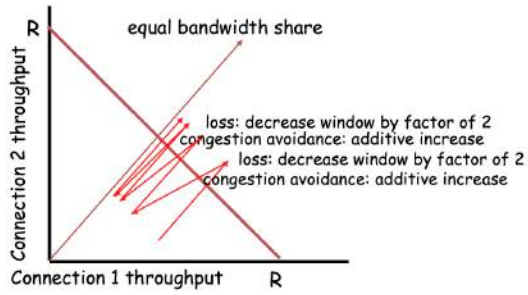
מכאן מתקבל האלגוריתם הבא:

```
for every ACK {
    W += 1/W // (Additive Increase )
        }
for every 3 dupacks {
        ssthreshold = W/2
        W = W/2
  // (Multiplicative Decrease)
}
for every timeout {
        ssthreshold = W/2
Enter Slow Start (W = 1)
}
```

נרצה לשאול, מהו ה-throughput של אלגוריתם זה כפונקציה של החלון וה-RTT? נבחין כי אם W הוא החלון כאשר פקטה אבדה היה throughput של $\frac{W}{\text{RTT}}$, החלון קטן ל-$\frac{W}{2}$ ואז מקבלים $\frac{W}{2\cdot\text{RTT}}$. לכן סך הכל נקבל $0.75\cdot\frac{W}{\text{RTT}}$.

כמו כן, נבחין כי אידאלית היינו רוצים שאם K ערוצי tcp חולקים את אותו ערוץ מידע, ה-bandwidth R של הלינק, יתחלק באופן שווה ל-$\frac{R}{K}$ בממוצע. האם הוא באמת הוגן?

נניח שיש לנו שני רכיבים שמנסים להתחבר, אחד התחבר וניהל שיחה כבר הרבה זמן, ולכן היה לו bandwidth R. לפתע הרכיב השני גם מתחבר. מה יקרה?



איור 35: תחילה יש עלייה ב-throughput של 1 וירידה בשל 2. כאשר בשלב מסויים, 1 מתחיל לחוות אובדן פקטות ולכן מקטין את גודל החלון פי 2, מה שמקטין מאוד את ה-throughput שלו. היות שהוא התחיל עם חלון הרבה יותר גדול, זה הרבה יותר דרמטי מהקטנת החלון כשאחנו חווים אבדן, שכן החלון שלנו קטן. על כן, אנחנו יכולים לגדול, ואז שוב יש אבדן, אבל אם החלון שלו גדול יותר, הוא יחווה הקטנה גדולה יותר בגודל החלון, ואם שלנו גדול יותר זה יקרה לנו. כלומר אנחנו נצליח להתאזן, ובפרט לקבל throughput שווה בממוצע.

! למעשה כשצריך שירותי multimedia לא משתמשים ב-tcp כדי שה-rate לא יישלט על ידי ה-congestion control ולכן הם משתמשים ב-udp.

7 פרוטוקול קישור בין רשתי BGP

עד כה הסברנו על העברת מידע בתוך רשת פנימית, כלומר intradomain – routing. אבל בפועל יש לנו רשתות גדולות, שצריכות לתקשר ביניהן, למשל, רשת פנימית של גוגל, ורשת של פייסבוק. לרשתות אלה אנו קוראים autonomous systems ויש כ-$50,000$ כאלה. לכן נוכל להתייחס אליהם כישויות ברשת גדולה יותר.

הטופולוגיה של הרשת שנקבל היא ספקי אינטרנט, ו-clients. לכל יישות יהיה מרחב כתובות שהיא חושפת לשאר הישויות, כדי שידעו על אלו כתובות היא אחראית. כמו כן, החיבורים בין הישויות הוא עסקי, שכן ספק אינטרנט יעדיף להעביר מידע דרך לקוח שלו מאשר דרך ספק אינטרנט שהוא משתמש בו, כי אחרת זה יעלה לו כסף. היחסים בין הישויות הם או באמת יחס של ספק-לקוח, אך גם של העברה שיתופית. למשל, ספק אינטרנט אחד יכול לאפשר לספק אחר להעביר דרכו מידע במחיר מוזל, ולהפך.

לרכיבים שאינם ספקי רשת, כלומר אינם ISPs, אנו קוראים stubs והם מהווים כ-85% מה-ASes. הם לא מאפשרים שירותי העברת מידע, והם מתקשרים דרך ספק האינטרנט. המזהה בפרוטוקול זה הוא ה-AS number שזה רצף של 16 ביטים (או 32).

הדרך שבה הרשת בוחרת כיצד להעביר מידע בין רשתות הוא באמצעות BGP – Border Gateway Protocol, שהוא פרוטוקול בשכבת האפליקציה, בפרט מעל TCP. בפרוטוקול זה, בגדול, כל רשת מפרסמת את מרחב הכתובות שלה לשאר הרשתות, ואת מרחב הכתובות של השכנים שלה, כך שהן יודעות כיצד להעביר את המידע. בשונה מתקשורת פנימית, המסלול שנבחר אינו בהכרח הקצר ביותר, שכן מסלול קצר יכול לעבור דרך ספק אינטרנט שהוא לא שלנו, ולכן זה יעלה לנו כסף. בפרט הוא עובד כך:

1. נקבל מסלולים משכנים ונפלטר מה המסלולים שאנחנו לא רוצים.
2. נבחר את המסלול הכי טוב לפי ההעדפה המקומית.
3. נפלטר איזה מסלולים אנחנו לא רוצים להעביר לשכנים.
4. נעביר את המסלולים שנבחרו לשכנים.

(א) לקוח יעביר מסלולים לכולם, אבל לא כאלה שכוללים את הספק שלו, לספקים אחרים. אחרת הם יעבירו דרכו מידע, והוא יצטרך לשלם על זה.

(ב) ספק יעביר מסלולים ללקוחות בלבד.

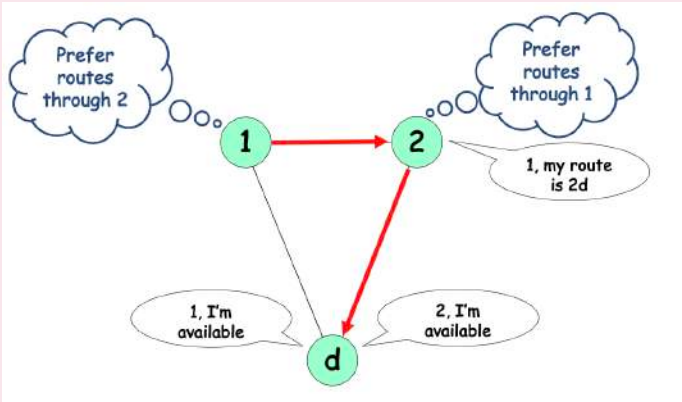
(ג) עמית יעביר מסלולים ללקוחות בלבד.

הדרך שבה מחושבים מסלולים ל-IP – prefixes הוא בלתי תלוי, כלומר אם רשת אחת פרסמה תחילית אחת, ואז עוד אחת, המסלולים יחושבו מחדש. כמו כן המסלולים המפורסמים כוללים את כל הרשתות, כדי למנוע מצב של לולאה.

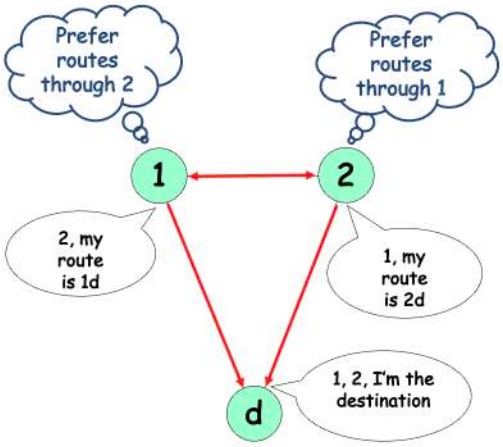
7.1 BGP Safe

כאשר מעבירים מסלולים בין רשתות, הרשת הכוללת צריכה להתייצב בשלב מסויים. אך ייתכנו מקרים שזה לא יקרה.

התייצבות BGP



איור 36: במקרה זה d מספר ל-2 שהוא זמין ולכן יגדיר את המסלול שלו ל-d להיות ישיר. לאחר מכן 2 יודיע ל-1 על המסלול שלו, ו-1 יבחר מסלול ישיר ל-2. עכשיו, גם כאשר d יכריז ל-1 שהוא זמין, המצב לא ישתנה ולכן קיבלנו מצב יציב.



איור 37: עכשיו d מכריז שהוא זמין ל-$1,2$ והם בוחרים במסלול ישיר ל-d. עכשיו שניהם יכריזו אחד לשני שיש להם מסלול ישיר ל-d, ופי ההעדפות שלהם, הם יגדירו מסלול ישיר אחד לשני, כלומר יעדיפו את המסלולים $12d, 21d$ במקום $1d, 2d$. הם יכריזו על המסלולים הנ"ל, יראו שנוצר מעגל ויחזרו למסלולים הישירים ל-d, וחוזר חלילה, כלומר נקבל מצב לא יציב. למעשה ייתכנו גם מצבים בהם לא יכול להיות בכלל מצב יציב.

הבעיה באוסילציות האלה היא שהן גורמות לתעבורה עודפת ברשת. על כן צריך להבטיח יציבות.

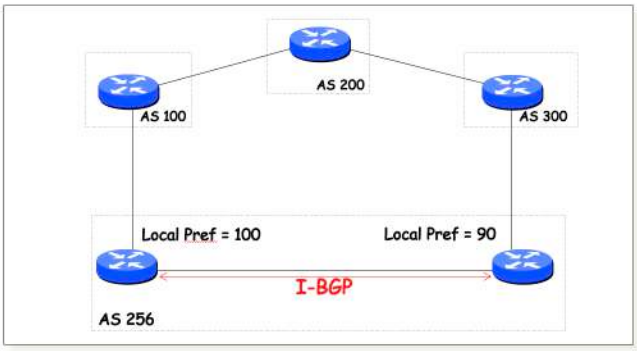
הדרך שבה נעשה זאת היא לפי העקרונות הבאים:

1. אין מעגלים בין ספקים, כלומר ספק קטן הוא לא לקוח של ספק גדול שהוא לקוח של ספק קטן.
2. ספקים יעדיפו להעביר מידע בין לקוחות ולא דרך ספקים.
3. אי הכרזה על מסלולים - ספק קטן לא יכריז על מסלול שלו לספק גדול A, עבור ספק גדול B, על מנת למנוע ניתוב של ספקים גדולים בין ספקים קטנים.

תחת התנאים אלה, הוכח ש-BGP יכול להתייצב, כלומר הוא בטוח.

7.2 BGP פנים-רשתי

נתבים בתוך רשתות צריכים להעביר מידע על BGP לרשתות אחרות. כאשר רשתות רוצות לתקשר, נפתח עבורן $\mathrm{E_{External}BGP-Session}$, בין הרווטרים (שנקראים גם default gateway) ואילו כאשר יש נתבים בתוך הרשת שרוצים לתקשר נפתח עבורם $\mathrm{I_{Internal}BGP-Session}$. ל-IBGP מטרה לניהול הוצאה של בקשות בין רשתיות מתוך הרשת:



איור 38: אם אנחנו AS256, והגיעה הודעה לנתב הימני, אנחנו נעביר אותה לנתב השמאלי שיוציא אותה החוצה, כי יש על זה רווח גדול יותר. זה למעשה מטרת ה-IBGP - החלטה דרך איזה נתב להוציא את ההודעה.

יכול להיות מצב בו כל הנתבים הרלוונטים בעלי אותה עדיפות מקומית, ובמקרה זה הנתב יבחר את נקודת היציאה הקרובה ביותר אליו, שנקבעת לפי $IGP - Interior\ Gateway\ Protocol$, BGP פנים רשתי. דבר זה גם נקרא $Hot - Potato - Routing$.
על כן סדר העדיפויות הוא כדלקמן:

1. עדיפות מקומית.
2. מסלול AS הקצר ביותר בארגונים בדרך ליעד, שנקבע בין השאר לפי IBGP.
3. נקודת היציאה הקרובה ביותר (Egress Point) - לזה אחראי ה-IGP, שמחשב את המסלולים פנימית בתוך הרשת.
4. שבירת שוויון ארביטרית.

7.3 BGP Security

מתקפות שאפשר לעשות הוא לנתר תעבורה בין רשתות, על ידי גישה לחיבור פיסי - אלה מתקפות משעממות יחסית כי לרוב אין לנו גישה פיסית, ובהנתן גישה פיסית אפשר לעשות הרבה מאוד דברים, כלומר זה מצב אידאלי מדי.

מתקפה אחרת היא מניפולציה על הפרוטוקול. למשל ארגון אחד יכול לקחת מרחב כתובות של ארגון אחר, ולהכריז על עצמו כאחראי עליו על ידי הפיכת המרחב לקצת יותר ספציפי, וכך תעבורה עם כתובות במרחב זה ינותבו אליו ולא אל הרכיב האמיתי, כי המרחב שלו ספציפי יותר. למעשה ב-2008 ממשלת פקיסטן החליטה להגביל את הגישה ליוטיוב. מה שהיה הגיוני שהיא תעשה, היה למנוע העברת מידע לרשת של יוטיוב, אמנם במקום זאת, היא הכריזה על מרחב כתובות של יוטיוב כשלה, כדי שכל בקשה מתוך פקיסטן ליוטיוב תנותב אליה. בפועל, ההכרזה הגיעה לשאר העולם ויוטיוב לא היה זמין יותר. זוהי טעות חריגה, שכן הדבר לא רק הופך את יוטיוב ללא זמין, אלא יוצר מתקפת DDos על הרשת שהכריזה על עצמה כיוטיוב, בניגוד לכוונת המשורר.
כדי לפתור בעיות אלה משתמשים בקריפטוגרפיה, אך זה איטי.

מתקפות נוספות שאפשר לבצע, הם במקום לתקוף את ה-session, או לבצע מניפולציה על הפרוטוקול, לתקוף את ה-$data - plane$, כלומר לשלוח מידע דרך נתיבים שונים ממה שיש לנו בטבלאות, לזרוק פקטות שאנחנו מקבלים וכו'. אין שום דבר שמבטיח שרשת אחת תענה לדרישות של ה-control plane שנוצר על ידי BGP וזו פרצת אבטחה.

חלק II

תרגולים

8 תרגול 1 - הסתברות

הערה 8.1. בשבוע הראשון לא היה תרגול.

ביצענו חזרה על מרחב מדגם, הסתברות מותנית, אי תלות, חוק בייס, נוסחת ההסתברות השלמה, משתנים מקריים, תוחלת, שונות, לינאריות התוחלת, נוסחת השונות לסכום.

דוגמה 8.1. פקטה טובה k_g היא פקטה שמקיימת כי ההסתברות שהיא שרדה לפחות T שניות, היא $\mathbb{P}(k_g \text{ time} > T) = e^{-T}$. פקטה רעה k_b מקיימת כי $\mathbb{P}(k_b \text{ time} > T) = e^{-1000T}$. מה ההסתברות שפקטה שרדה לפחות t שניות? מתקיים מנוסחת ההסתברות השלמה

$$\mathbb{P}(k \text{ time} = t) = \mathbb{P}[k \text{ time} = t \mid k = k_b]\,\mathbb{P}[k = k_b] + \mathbb{P}[k \text{ time} = t \mid k = k_g]\,\mathbb{P}[k = k_g]$$

נסמן $p = \mathbb{P}[k = k_g]$ אזי בהכרח

$$\begin{aligned}\mathbb{P}(k \text{ time} > t) &= (1-p)\,\mathbb{P}[k_b \text{ time} = t] + p\mathbb{P}[k_g \text{ time} = t] \\ &= (1-p)\,e^{-1000t} + pe^{-t}\end{aligned}$$

עתה נניח שנתון לנו שפקטה שרדה T שניות. מה ההסתברות שזו פקטה טובה? מתקיים כי

$$\begin{aligned}\mathbb{P}[k = k_g \mid k \text{ time} > T] &\overset{\text{Bayes}}{=} \frac{\mathbb{P}[k \text{ time} > T \mid k = k_g]\,\mathbb{P}[k = k_g]}{\mathbb{P}[k \text{ time} > T]} \\ &= \frac{pe^{-T}}{(1-p)\,e^{-1000t} + pe^{-t}}\end{aligned}$$

דוגמאות למשתנים מקריים בדידים

1. <u>משתנה מקרי ברנולי</u>: נסמן $X \sim \mathrm{Ber}(p)$. המשתנה בודק הצלחה של ניסוי הטלת מוטה p בודד. מתקיים כי

$$\begin{aligned}&\mathbb{P}(X = 1) = p \\ &\mathbb{P}(X = 0) = 1 - p \\ &\mathbb{E}[X] = p \\ &\mathrm{Var}(X) = p(1-p)\end{aligned}$$

2. <u>משתנה מקרי בינומי</u>: נסמן $X \sim \mathrm{Bin}(n, p)$. המשתנה מונה מספר הצלחות של n הטלות מטבע מוטה p. ומתקיים

כי

$$\mathbb{P}(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$$
$$\mathbb{E}[X]=np$$
$$\mathrm{Var}(X)=np(1-p)$$

3. <u>משתנה מקרי גאומטרי</u>: נסמן $X\sim\mathrm{Geo}(p)$. המשתנה מונה את מספר הניסיונות עד לקבלת הצלחה ראשונה. מתקיים כי

$$\mathbb{P}[X=k]=(1-p)^{k-1}p$$
$$\mathbb{E}[X]=\frac{1}{p}$$
$$\mathrm{Var}(X)=\frac{1-p}{p^2}$$

4. <u>משתנה מקרי פואסוני</u>: נסמן $X\sim\mathrm{Poa}(\lambda)$. המשתנה בודק את המקרה הגבולי של ההתפלגות הבינומית $(n\to\infty,p\to 0)$ ולכן מסתכלים על הממוצע λ במקום על np. מתקיים

$$\mathbb{P}(X=k)=\frac{\lambda^k}{k!}e^{-\lambda}$$
$$\mathbb{E}[X]=\lambda$$
$$\mathrm{Var}(X)=\lambda$$

דוגמה 8.2. נתונה שרשרת של $n-1$ נתבים, כאשר בכל נתב ההסתברות שפקטה תזרק היא p באופן בלתי תלוי, כולל ההגעה לנתב הראשון מהמקור. מה ההסתברות שפקטה תגיע ליעד? ההסתברות היא $(1-p)^n$. כמה פקטות נצטרך לשלוח עד שהפקטה תגיע ליעד? היות שההסתברות להצלחה היא $(1-p)^{n-1}$ נקבל כי מספר הפקטות $X\sim\mathrm{Geo}\left((1-p)^n\right)$. עתה נשנה את ההנחה, ונניח כי כל נתב שולח פקטות לנתב הבא עד שהפקטה מגיעה, מלבד המקור, שההסתברות שהפקטה שהוא שולח תגיע לנתב הראשון היא $1-p$. במקרה זה ההסתברות להצלחה היא $1-p$ ו-$X\sim\mathrm{Geo}(1-p)$ וכן $\mathbb{E}[X]=\frac{1}{1-p}$.

דוגמה 8.3. (תהליך פואסוני) בממוצע מגיעים $\lambda=50\frac{\text{meteors}}{\text{hour}}$. ההגעה שלהם ניתנת לתיאור כתהליך פואסוני עם λ. נסמן את מספר המטאורים שהגיעו ב-t שעות ב-M, ונקבל כי $M\sim\mathrm{Poa}(\lambda t)$. מבחינת יחידות - הממוצע צריך להיות חסר יחידות, והוא אכן כזה, כי מטאורים זו לא יחידה. מה ההסתברות לקבלת 2 מטאורים, במהלך 15 שניות? מתקיים כי

$$\mathbb{P}[M=2]=\frac{(50t)^2}{2!}e^{-50t}$$

נציב $t=15\text{sec}=\frac{15}{3600}\text{hours}$ ונקבל $\mathbb{P}[M=2]\approx 0.018$. עתה נשאל, מה ההסתברות לקבל לפחות 2 מטאורים? נחשב

$$\begin{aligned}\mathbb{P}[M>1]&=1-\mathbb{P}[M\le 1]=1-(\mathbb{P}[M=0]+\mathbb{P}[M=1])\\&=1-\left(\frac{(50\cdot t)^2}{0!}e^{-50\cdot t}+\frac{(50\cdot t)^2}{1!}e^{-50\cdot t}\right)\\&\approx 0.019\end{aligned}$$

שיפור של 0.001. זה אומר שההגעה של פקטות לאחר השתיים הראשונות תורמות מעט מאוד להסתברות לקבלתן. מדוע זה רלוונטי עבורנו? אנחנו נראה בקרוב כיצד לבנות התפלגות פואסונית על הרשת, כאשר המטאורים יהיו הפקטות.

משתנים מקריים רציפים

למשתנה מקרי רציף פונקציית התפלגות $\mathbb{P}[a \leq X \leq b]$. אם הוא רציף בהחלט אז יש לו פונקציית צפיפות f שמקיימת

$$\mathbb{P}[a \leq X \leq b] = \int_a^b f(x)\ \mathrm{dx}$$

$$\int_{-\infty}^{\infty} f(x)\ \mathrm{dx} = 1$$

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x)\ \mathrm{dx}$$

$$\mathrm{Var}[X] = \int_{-\infty}^{\infty} (x - \mathbb{E}(X))^2 f(x)\ \mathrm{dx}$$

דוגמאות למשתנים מקריים רציפים

1. <u>משתנה מקרי אחיד</u>: נסמן $X \sim \mathrm{Unif}[a,b]$. מתקיים

$$f(x) = \begin{cases} \frac{1}{b-a} & x \in [a,b] \\ 0 & x \notin [a,b] \end{cases}$$

$$\mathbb{E}[X] = \frac{a+b}{2}$$

$$\mathrm{Var}[X] = \frac{(b-a)^2}{12}$$

2. <u>משתנה מקרי אקספוננציאלי</u>: נסמן $X \sim \mathrm{Exp}(\lambda)$. אמתקיים

$$f(x) = \begin{cases} \lambda e^{-\lambda} & x > 0 \\ 0 & x \leq 0 \end{cases}$$

$$\mathbb{E}[X] = \frac{1}{\lambda}$$

$$\mathrm{Var}(X) = \frac{1}{\lambda^2}$$

3. <u>משתנה מקרי נורמלי (גאוסייני)</u>: נסמן $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$. מתקיים כי

$$f\left(x\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$
$$\mathbb{E}\left[X\right] = \mu$$
$$\mathrm{Var}\left(X\right) = \sigma^2$$

9 תרגול 2 - פרוטוקול ALOHA

מושגים

הגדרה 9.1. רוחב פס (bandwidth) הוא כמות המידע שאפשר להעביר בפס העברת המידע. יחידות: $\frac{\text{bits}}{\text{second}}$. נסמן אותו ב-B.

הגדרה 9.2. ה-Goodput הוא החלק היחסי מה-bandwidth ששולח מידע בצורה מוצלחת. נסמן אותו ב-η. אין לו יחידות.

הגדרה 9.3. ה-Throughput הוא החלק היחסי מה-bandwidth ששולח מידע. אין לו יחידות או סימן.

בהרצאה ראינו הגדרות יותר אינטואיטיביות.

נבחין כי מתקיים

$$\eta = \frac{\text{אפקטיביות כוללת של השימוש ב-bandwidth}}{\text{bandwidth}} = (\star)\ \frac{\text{זמן אפקטיבי כולל ממוצע}}{\text{total time}}$$
$$0 \leq \eta \leq 1$$

כאשר יש רעש, מתקיים כי $\text{Throughput} \neq \text{Goodput}$.

דוגמה 9.1. נתון $\mathrm{B} = 5\left[\frac{\text{bit}}{\text{sec}}\right]$ ופקטה בגודל 30bits. אזי קצב העברת הפקטה הוא $\mathrm{T} = \frac{|\text{packet}|}{\mathrm{B}} = \frac{30}{5} = 6\text{sec}$. עתה נתון שחייבים להעביר 3 פקטות בגודל 30bits במשך 2sec. מה רוחב הפס שצריך? מתקיים כי

$$\mathrm{T} = \frac{30 \cdot 3}{\mathrm{B}} = 2$$

ולכן $\mathrm{B} = 45$.

פרוטוקול ALOHA

בפרוטוקול זה יהיו לנו שני משתנים אותה נשנה בכל מודל:

1. זמן: חלוקה למקטעים קבועים (Slotted), או קו רציף (Pure).
2. התפלגות שליחת ההודעות X_p, כלומר מספר הפקטות שהועברו במרווח זמן T: זהו משתנה מקרי שצריך לקבוע כיצד מתפלג.

9.1 קביעת פרמטר הזמן

Slotted ALOHA בפרוטוקול זה יש לנו מרווחי זמן קבועים. כאשר כל מרווח מספיק בדיוק לשליחת פקטה אחת. כל רכיב שולח פקטה בתחילת מרווח זמן ולא באמצע. מרווח זמן זה מחשבים באמצעות $\mathrm{B}, \mathrm{PacketSize}$.

Pure ALOHA בפרוטוקול זה לכל רכיב יש שעון, אבל השעונים לא מסונכרנים. אנו נניח שיש אינסוף יחידות קצה. כלומר אינסוף רכיבים ששולחים הודעות.

9.2 קביעת ההתפלגות

דרך אחת היא להסתכל על הפקטות בצורה דיסקרטית עם התפלגות לשליחת פקטה p. במקרה זה מספר הפקטות שנשלחו במרווח זמן, יתפלג בינומית. כלומר עבור m פקטות,

$$X_p \sim \mathrm{Bin}\left(n,p\right): \qquad \mathbb{P}\left[X_p = n\right] = \binom{m}{n} p^n \left(1-p\right)^{m-n}$$

דרך אחרת היא להסתכל על זה מנקודת מבט כוללת. אם ידוע שבממוצע נשלחות g פקטות, אז בפרק זמן T ישלחו $g \cdot T$ פקטות. על כן במקרה זה זמן ההגעה של הפקטות יתפלג פואסונות עם פרמטר $\lambda = g \cdot T$, כלומר ההסתברות ש-i פקטות יגיעו בזמן t תקיים:

$$X_p \sim \mathrm{Poa}\left(g \cdot T\right) : \mathbb{P}_{K=i}\left[X_p = t\right] = \frac{\left(gT\right)^i}{i!} e^{-gT}$$

דוגמה 9.2. נניח שיש n יחידות קצה, שהמודל הוא Slotted וש-X_p מתפלג בינומית. נרצה לחשב את ה-Goodput. נבחין כי

$$\mathbb{P}\left[X_p = 1\right] = n \cdot p \left(1-p\right)^{n-1}$$

נגדיר $T_{\text{success}} = \begin{cases} T & \text{שלחנו הודעה בהצלחה} \\ 0 & \text{else} \end{cases}$. נבחין כי $\mathbb{P}\left[T_{\text{success}} = T\right] = \mathbb{P}\left[X_p = 1\right] = \mathrm{P}_{\text{success}}$ וכי $\mathbb{E}\left[T_{\text{success}}\right] = T\mathrm{P}_{\text{success}}$ על כן

$$\text{Goodput} = \frac{\text{successfull average}}{\text{total time}} = \frac{T\mathrm{P}_{\text{success}}}{T} = \mathrm{P}_{\text{success}} = n \cdot p\left(1-p\right)^{n-1}$$

כדי לקבל מקסימום נגזור ונשווה לאפס ונקבל ש-$p = \frac{1}{n}$ נותן η מקסימלי. מעניין. אנחנו מקבלים שאנטרופיה מינימלית נותנת Goodput מקסימלי. במקרה זה נקבע קיבולת של $\frac{1}{e}$ כאשר $n \to \infty$. כפי שראינו בהרצאה. זה מעט מדי.

דוגמה 9.3. עתה נרצה לטפל במקרה בו המודל הוא Pure. במקרה זה כל יחידת קצה מכילה Slots פשוט אין סנכרון. לכן הזמן האפקטיבי בו אנו רוצים שרק יחידה אחת תשלח מידע, הוא $2T$. לכן נקבל $\mathrm{P}_{\text{success}} = n \cdot p\left(1-p\right)^{2(n-1)}$ שכן אנו רוצים שיחידה תשלח מידע ושכל היחידות לא ישלחו דבר בשני Slots. במקרה זה המקסימום שנקבל הוא $\frac{1}{2e}$ שזה גרוע עוד יותר.

דוגמה 9.4. נניח שמספר ההודעות לשנייה ב-Slot כלשהי הוא g, שהמודל הוא Slotted וש-X_p מתפלג פואסונית. על כן gT מספר ההודעות הממוצע שהגיעו בזמן T ל-Slot כלשהו. בגלל שזו התפלגות פואסונית, אם יש קצב שונה לכל Slot כלומר $X_i \sim \lambda_i$ נקבל כי סך כל ההודעות שהגיעו $X_p = \sum X_i$ מקיימות כי $X_p \sim \mathrm{Poa}\left(\sum \lambda_i\right)$, שכן יש אי תלות ל-$X_i$. במקרה בו $\lambda_i = \lambda$ לכל i נקבל כי $X_p \sim \mathrm{Poa}\left(m\lambda\right) = \mathrm{Poa}\left(mgT\right)$ כאשר יש m Slots. נחשב אם כך

$$\mathrm{P}_{\text{Success}} = \mathbb{P}_{K=1}\left[t = T\right] = gTe^{-gT}$$

$$T_{\text{success}} = \begin{cases} T & \mathrm{P}_{\text{success}} \\ 0 & 1 - \mathrm{P}_{\text{success}} \end{cases}$$

$$\mathbb{E}\left[T_{\text{success}}\right] = T\mathrm{P}_{\text{success}}$$

דוגמה 9.5. נמשיך עם הדוגמא הקודמת למקרה ה-Pure ונחשב את ההסתברות להצלחה. מהאי תלות של העברת המידע בזמנים שונים, נקבל כי

$$\mathrm{P}_{\text{Success}} = \mathbb{P}_{K=0}\left[t=T\right] \cdot \mathbb{P}_{K=1}\left[t=T\right] = e^{-gT} gTe^{-gT} = gTe^{-2gT}$$

וסיימנו.

דוגמה 9.6. נניח כי המודל הוא Slotted פואסוני $X_p \sim \mathrm{Poa}\left(gT\right)$.

נניח שיש לנו הודעות ארוכות L קצרות S כאשר L, S מסמנים את הזמן הדרוש להעברת הפקטה. כאשר הודעה מסוג L מגיעה בהסתברות p. מה ההסתברות ל-L מוצלח? נרצה כי בזמן T תשלח הודעה אחת, כלומר

$$\begin{aligned}\mathrm{P}^{\mathrm{L}}_{\text{success}} &= p\mathbb{P}_{K=1}\left[t=L\right] = p\left(gL\right)e^{-gL}\\ \mathrm{P}^{\mathrm{S}}_{\text{success}} &= \mathbb{P}_{K=1}\left[t=S\right] = \left(gS\right)e^{-gS}\end{aligned}$$

כדי לחשב את ה-Goodput נגדיר $T_{\text{success}} = \begin{cases} L & \mathrm{P}^{\mathrm{L}}_{\text{success}} \\ S & \mathrm{P}^{\mathrm{S}}_{\text{success}} \\ 0 & 1-\mathrm{P}^{\mathrm{L}}_{\text{success}} - \mathrm{P}^{\mathrm{S}}_{\text{success}} \end{cases}$. על כן $\mathbb{E}\left[T_{\text{success}}\right] = L\mathrm{P}^{\mathrm{L}}_{\text{success}} + S\mathrm{P}^{\mathrm{S}}_{\text{success}}$ ולכן

$$\eta = \frac{L\mathrm{P}^{\mathrm{L}}_{\text{success}} + S\mathrm{P}^{\mathrm{S}}_{\text{success}}}{L}$$

! למה מותר לנו להסתכל על כל מקרה לגופו, ולסכום הכל? הרי יש תלות בפועל בין הרכיבים, ולכן צריך להתחשב בה. התשובה היא שעל אף שיש תלות בין מספר הפקטות ששלח כל רכיב, אנחנו יכולים להסתכל על X_i כמספר הפקטות ששלח הרכיב ה-i. אזי מספר הפקטות הכולל שנשלחו בהצלחה הוא $X = \sum X_i$. מלינאריות התוחלת, $\mathbb{E}\left[X\right] = \sum \mathbb{E}\left[X_i\right]$. על כן, מספיק להסתכל כמה הודעות מוצלחות כל רכיב אחד שלח בממוצע, למרות שהרכיבים תלויים.

דוגמה 9.7. עתה נסתכל על המקרה ה-Pure. גם כאן נרצה $\mathbb{P}_{K=1}\left[t=L\right]$. אבל זה לא מספיק. לפני זמן t צריך לחלק למקרים. אם נשלחה הועדה S בזמן $t-L$, היא לא תפריע לנו. אבל אם נשלחה הודעה S בזמן $t-S$ אז היא כן תפריע. כלומר נרצה כי בטווח $t-S$ לא נשלחו הודעות קצרות, ובטווח $t-L$ לא נשלחו הודעות ארוכות. היות שיש חפיפה. נוכל להסתכל על כל קבוצה של הודעות בנפרד כאשר בטווח $t-L$ נשלחו הודעות L בקצב gp ובטווח $t-S$ נשלחו הודעות S בקצב $g\left(1-p\right)$ על כן

$$\begin{aligned}\mathrm{P}^{\mathrm{L}}_{\text{success}} &= \mathbb{P}_{K=0}\left[t=L-S\right] \cdot \mathbb{P}_{K=0}\left[t=S\right] \cdot \mathbb{P}_{K=1}\left[t=L\right] \cdot p\\ &= e^{-g(p)(L-S)} \cdot e^{-g(1-p)S} \cdot gLe^{-gL}p\end{aligned}$$

עתה נרצה לעשות דבר דומה עבור הודעות קצרות. במקרה זה נרצה בטווח זמן S את $\mathbb{P}_{K=1}\left[t=S\right] \cdot \left(1-p\right)$ בשאר הטווחים נרצה התנהגות דומה ולכן נקבל

$$\begin{aligned}\mathrm{P}^{\mathrm{S}}_{\text{success}} &= \mathbb{P}_{K=0}\left[t=L-S\right] \cdot \mathbb{P}_{K=0}\left[t=S\right] \cdot \mathbb{P}_{K=1}\left[t=S\right] \cdot \left(1-p\right)\\ &= e^{-g(p)(L-S)} \cdot e^{-g(1-p)S} \cdot gSe^{-gS}\left(1-p\right)\end{aligned}$$

על כן

$$\begin{aligned}\eta &= \frac{L \cdot \mathrm{P}^{\mathrm{L}}_{\mathrm{success}}}{L} + \frac{S \cdot \mathrm{P}^{\mathrm{S}}_{\mathrm{success}}}{S} \\ &= e^{-g(p)(L-S)-gL-g(1-p)S} gp + (1-p)\, g e^{-g(p)(L-S)-gS-g(1-p)S}\end{aligned}$$

ניתוח פרוטוקול ALOHA

כדי לחשב את η עבור פרוטוקול ALOHA ביצענו את ההליך הבא:

1. חישבנו את $\mathrm{P}_{\mathrm{success}}$ שתלויה בהתפלגות של ההודעות (Bin/Poa), ובסוג הפרוטוקול (Pure/Slotted).
2. חישבנו את $\mathbb{E}\left[\mathrm{T}_{\mathrm{success}}\right]$.
3. הסקנו כי $\eta = \frac{\mathbb{E}[\mathrm{T}_{\mathrm{success}}]}{\mathrm{Total}}$.

10 תרגול 3 - רוחב פס, פרוטוקולים ALOHA, EC

10.1 רוחב פס

הסיב האופטי עליו עוברים הביטים נותן לכל רכיב רוחב פס B. אם הזמן היחסי בו הוא שולח מידע הוא t נקבל כי הניצולת שלו היא $t \cdot \mathrm{B}$. על כן, נקבל קשר "מדוייק" יותר ל-goodput:

$$\eta = \mathrm{B} \cdot \frac{\mathbb{E}\left[\mathrm{T}_{\text{success}}\right]}{\text{Total Time}}$$

היות ש-B נקבע משיקולים פיסיקליים, אנחנו תמיד נקבל אותו.

> **!** יחד עם זאת, אנחנו יכולים להגדיר את η כחלק היחסי מה-Bandwidth המוקצה לרכיב הנ"ל. אם הוקצה לו B נקבל כי $\eta_{\mathrm{B}} = \frac{\mathbb{E}[\mathrm{T}_{\text{success}}]}{\text{Total Time}}$. **חשוב להבדיל בין שני הביטויים.**

10.2 שאלות על פרוטוקול ALOHA

שאלה נתונים 9 כפרים על מאדים באופן הבא:

1	2	3
4	5	6
7	8	9

כאשר כל כפר מדבר עם עצמו בפרוטוקול כלשהו. נתון כי כל כפר יכול לתקשר עם 4 השכנים שלו: למעלה, למטה, ימינה שמאלה. נניח שהכפרים האי זוגיים מתקשרים בפרוטוקול Slotted ALOHA עם g_0 וכי הכפרים הזוגיים מתקשרים ב-Pure ALOHA עם g_e. התקשורת היא פנימית ולא חיצונית. נניח תחילה כי רק הכפרים הזוגיים קיימים, כלומר מאדים נראה כך:

	2	
4		6
	8	

מה הוא ה-goodput של כפר זוגי?

פתרון 1. נקבל כי ההסתברויות להצלחה הן

$$\mathrm{P}_{\text{pure}} = \mathbb{P}_{K=0}\left[t=T\right] \cdot \mathbb{P}_{K=1}\left[t=T\right] = e^{-g_e T} g_e T e^{-g_e T} = g_e T e^{-2 g_e T}$$
$$\mathrm{P}_{\text{slotted}} = \mathbb{P}_{K=1}\left[t=T\right] = g_o t e^{-g_o T}$$

על כן, בגלל שכל כפר במקרה זה מתקשר ב-Pure, נקבל כי עבור $\mathrm{T}_{\text{success}} = \begin{cases} \mathrm{T} & \text{success} \\ 0 & \text{fail} \end{cases}$ על כן

$$\eta_{\text{pure}} = \frac{\mathbb{E}\left[\mathrm{T}_{\text{success}}\right]}{\mathrm{T}} = \frac{\mathrm{P}_{\text{pure}} \cdot \mathrm{T}}{\mathrm{T}} = \mathrm{P}_{\text{pure}} = g_e T e^{-2 g_e T}$$
$$\eta_{\text{slotted}} = \frac{\mathbb{E}\left[\mathrm{T}_{\text{success}}\right]}{\mathrm{T}} = \frac{\mathrm{P}_{\text{slotted}} \cdot \mathrm{T}}{\mathrm{T}} = \mathrm{P}_{\text{slotted}} = g_o T e^{-g_o T}$$

וזה ה-Goodput לכל כפר בנפרד. היות שכל כפר מהווה $\frac{1}{4}$ מהרשת, נקבל כי ה-goodput הכולל הוא

$$4 \cdot \frac{1}{4} \cdot g_e T e^{-2g_e T} = g_e T e^{-2g_e T}$$

עתה נעשה תהליך דומה למקרה האי זוגי. נקבל באופן דומה כי $\eta_{\text{total}} = g_o T e^{-g_o T}$.

שאלה (המשך) בהמשך לשאלה הקודמת, מה יקרה אם נשלב את שני המקרים? כלומר עתה מאדים נראה כך:

1	2	3
4	5	6
7	8	9

פתרון 2. נבחין כי יש לנו שלושה סוגי רכיבים:

1. כפר פינתי $(1,3,7,9)$.
2. כפר במרכז צלע $(2,4,6,8)$.
3. כפר במרכז הגריד (5).

ננתח אם כך את ה-goodput של כל אחד מסוגי הרכיבים. נתחיל עם כפר מספר 1. כדי שהוא יצליח, אנחנו צריכים שבטווח הזמן T, הוא ישלח הודעה אחת, וכל השכנים שלו $(2,4)$ לא ישלחו הודעות גם בטווח $[T,T+1]$ אבל גם בטווח $[T-1,T]$, היות שהם לא מסונכרנים. על כן כדי ש-$2,4$ לא ישלחו הודעות ולא יפריעו ל-1 נרצה את

$$\underbrace{\mathbb{P}^{\text{Pure}}_{K=0}[t=T] \cdot \mathbb{P}^{\text{Pure}}_{K=0}[t=T]}_{[T-1,T]} \cdot \underbrace{\mathbb{P}^{\text{Pure}}_{K=0}[t=T] \cdot \mathbb{P}^{\text{Pure}}_{K=0}[t=T]}_{[T,T+1]}$$

כאשר אנחנו מסתכלים על אורך הזמן של הטווח, ולא על הזמן של תחילת הטווח, לכן הביטוי הראשון הוא בזמן $t=T$ ולא $t=T-1$. על כן

$$\begin{aligned}\mathrm{P}^{\text{corner}}_{\text{success}} &= \mathbb{P}^{\text{Slotted}}_{K=1}[t=T] \cdot \mathbb{P}^{\text{Pure}}_{K=0}[t=T] \cdot \mathbb{P}^{\text{Pure}}_{K=0}[t=T] \cdot \mathbb{P}^{\text{Pure}}_{K=0}[t=T] \cdot \mathbb{P}^{\text{Pure}}_{K=0}[t=T] \\ &= \mathbb{P}^{\text{Slotted}}_{K=1}[t=T] \cdot \left(\mathbb{P}^{\text{Pure}}_{K=0}[t=T]\right)^4\end{aligned}$$

עתה נסתכל על כפר במרכז הצלע, למשל 2. נרצה כי $1,3,5$ ישלחו אפס הודעות מזמן t. היות שהם Slotted מספיק להסתכל על הטווח $[T,T+1]$, כלומר $\mathbb{P}^{\text{Slotted}}_{K=0}[t=T]^3$. נרצה כי 2 ישלח הודעה בטווח $[T,T+1]$ אבל שהוא גם לא ישלח ב-$[T-1,T]$ כי הוא Pure, כלומר $\mathbb{P}^{\text{Pure}}_{K=1}[t=T] \cdot \mathbb{P}^{\text{Pure}}_{K=0}[t=T]$. סך הכל

$$\mathrm{P}^{\text{mid}}_{\text{success}} = \mathbb{P}^{\text{Pure}}_{K=1}[t=T] \cdot \mathbb{P}^{\text{Pure}}_{K=0}[t=T] \cdot \left(\mathbb{P}^{\text{Slotted}}_{K=0}[t=T]^3\right)$$

יחד עם זאת, עולה מכאן הבעיה הבאה. אנחנו הנחנו שה-Slot של ה-Pure ממוקם בדיוק בסופו של ה-Slot של ה-Slotted, וקיבלנו כי הרכיבים לא יפריעו בטווח $[T-1,T]$. אבל האם בממוצע זה מה שקורה? התשובה היא לא. בממוצע, ה-Pure הנ"ל יחפוף ל-Slotted, ולכן המקרה שתיארנו כאן זניח. לכן צריך לכפול ב-$\left(\mathbb{P}^{\text{Slotted}}_{K=0}[t=T]^3\right)^2$. לחילופין, אפשר לכפול ב-$\left(\mathbb{P}^{\text{Slotted}}_{K=0}[t=2T]^3\right)$. זה שקול.

עתה נטפל ברכיב המרכזי (5). מי שמשפיעים עליו הם $(2,4,6,8)$, שהם כל הרכיבים הזוגיים. היות שהם Pure, נקבל כי ההסתברות שהם לא שלחו הודעה בטווחים $[T-1,T],[T,T+1]$ היא $\left(\mathbb{P}_{K=0}^{\text{Pure}}[t=T]\right)^8$ וסך הכל

$$\mathrm{P}_{\text{success}}^{\text{center}}=\mathbb{P}_{K=1}^{\text{Slotted}}[t=T]\left(\mathbb{P}_{K=0}^{\text{Pure}}[t=T]\right)^8$$

סך הכל נקבל ש-$\eta_{\text{total}}=\frac{1}{9}\left[4\mathrm{P}_{\text{success}}^{\text{corner}}+4\mathrm{P}_{\text{success}}^{\text{mid}}+\mathrm{P}_{\text{success}}^{\text{center}}\right]$.

עולה השאלה למה אנחנו יכולים לבחור את נקודת ההתחלה של ה-Slot? הרי הרכיב מדבר ב-Pure ולכן הוא יכול להתחיל מאיזה נקודה שהוא רוצה. חשוב לזכור שהרכיב עצמו עובד בפרוטוקול Slotted, ההבדל היחיד הוא שהשעון לא מסונכרן עם האחרים. לכן מבחינתו, הוא התחיל מתחילת Slot, ומבחינתו, כדי להצליח, דרוש הניתוח שביצענו. אם נסתכל על זה מנקודת המבט של הרכיבים ה-Slotted, שלהם שעונים מסונכרנים, היינו יכולים להסתכל על המקרה הממוצע, כלומר לחשב את כל התוחלת לפי הזמן בו מתחילה ה-Pure שהוא $\alpha T+(1-\alpha)(T+1)$ כאשר $\alpha\in(0,1)$. מכאן אנו מסיקים שביכולתנו למקם את ה-Slot, בזהירות, היות שצריך להזהר ממקרים שלא מתכתבים עם הממוצע (כמו הטעות שעשינו קודם). זה נותן לנו אפשרות לבצע בחירות מקלות.

שאלה (Slotted) נתונים שני רכיבים וכי ה-Slots בגודל S.

1. רכיב מסוג A השולח הודעות באורך S, בהסתברות p. רוחב הפס שלו הוא $\frac{1}{4}B$.
2. רכיב מסוג B השולח הודעות באורך αS בהסתברות $1-p$ כאשר נתון ש-$\alpha>1$. רוחב הפס שלו הוא $\frac{3}{4}B$.

מה הוא ה-Goodput אם נתון כי שליחת ההודעות בפרק זמן T מתפלגת $\mathrm{Poa}(gT)$?

פתרון 3. כפי שתיארנו קודם, מלינאריות התוחלת אפשר להסתכל על כל אחד מהרכיבים כתהליך נפרד, כאשר אנחנו משקללים את חלקו היחסי בתהליך הכולל.

הרכיב A שולח הודעות בהסתברות p ולכן בממוצע ישלח בקצב gp כלומר $X_A\sim\mathrm{Poa}(gp)$. מצד שני, הרכיב B ישלח בקצב $(1-p)g$ ולכן $X_B\sim\mathrm{Poa}(g(1-p))$.

נתחיל עם הודעה ארוכה ששולח B. ב-Slot הראשון נרצה שתשוגר ההודעה, כלומר $\mathbb{P}_{K=1}^{\mathrm{B}}[t=S]$. מצד שני, נרצה ש-$B$ לא ישגר הודעות בזמן שהפקטה הקודמת נשלחה. היות שלוקח לה להגיע αS, צריך ש-B לא ישלח הודעות החל **מהטווח הבא** שיחפוף עם $[S,S+\alpha S]$, היות ש-α אינו בהכרח מספר שלם, הטווח הוא $[2S,S+\lceil\alpha\rceil S]$, שכן בטווח $[S,2S]$ כבר שלחנו מידע. ולכן נקבל $(\lceil\alpha\rceil-1)S$. כלומר $\mathbb{P}_{K=0}^{\mathrm{B}}[t=(\lceil\alpha\rceil-1)S]$.

לזאת נרצה להוסיף את המקרה שבו B שלח הודעה לפני ההודעה הנ"ל, כלומר בטווח $[S-\lceil\alpha\rceil S,S]$, ולכן תתווסף $\mathbb{P}_{K=0}^{\mathrm{B}}[t=(\lceil\alpha\rceil-1)S]$.

לבסוף נדרוש ש-A לא שלח הודעות בטווח $[S,S+\lceil\alpha\rceil S]$ ונקבל $\mathbb{P}_{K=0}^{\mathrm{A}}[t=\lceil\alpha\rceil S]$.

סך הכל קיבלנו

$$\mathrm{P}_{\text{success}}^{\mathrm{B}}=\mathbb{P}_{K=0}^{\mathrm{B}}[t=(\lceil\alpha\rceil-1)S]\cdot\mathbb{P}_{K=1}^{\mathrm{B}}[t=S]\cdot\mathbb{P}_{K=0}^{\mathrm{A}}[t=\lceil\alpha\rceil S]\cdot\mathbb{P}_{K=0}^{\mathrm{B}}[t=(\lceil\alpha\rceil-1)S]$$

עתה נסתכל על הודעות קצרות ששולח A. נרצה ש-A ישלח הודעה $\mathbb{P}_{K=1}^{\mathrm{A}}[t=S]$. הוא לא יפריע לעצמו כי ה-Slots באורך S. עתה, נרצה ש-B לא יפריע לו, כלומר שלא ישלח הודעות בטווח $[S,2S]$ כלומר $\mathbb{P}_{K=0}^{\mathrm{B}}[t=S]$ ובטווח $[S-(\lceil\alpha\rceil S),S]$ כלומר $\mathbb{P}_{K=0}^{\mathrm{B}}[t=(\lceil\alpha\rceil-1)S]$. סך הכל

$$\mathrm{P}_{\text{success}}^{\mathrm{A}}=\mathbb{P}_{K=0}^{\mathrm{B}}[t=(\lceil\alpha\rceil-1)S]\cdot\mathbb{P}_{K=1}^{\mathrm{A}}[t=S]\cdot\mathbb{P}_{K=0}^{\mathrm{B}}[t=S]$$

נגדיר עתה את משתני הזמן $\mathrm{T_A} = \begin{cases} S & \mathrm{P}^{\mathrm{A}}_{\mathrm{success}} \\ 0 & \mathrm{else} \end{cases}, \mathrm{T_B} = \begin{cases} \alpha S & \mathrm{P}^{\mathrm{B}}_{\mathrm{success}} \\ 0 & \mathrm{else} \end{cases}$. לכן

$$\mathbb{E}\left[\mathrm{T}_{\mathrm{success}}\right] = \mathbb{E}\left[\mathrm{T_B} + \mathrm{T_A}\right] = \alpha S \mathbb{P}^{\mathrm{B}}_{\mathrm{success}} + S\mathbb{P}^{\mathrm{A}}_{\mathrm{success}}$$

סך הזמן עליו הסתכלנו עבור A הוא S היות שהסתכלנו על Slot בגודל S. עבור B הסתכלנו על $\lceil\alpha\rceil S$ ולכן

$$\begin{aligned}\eta &= \frac{3}{4}\cdot\frac{\alpha S\mathbb{P}^{\mathrm{B}}_{\mathrm{success}}}{\lceil\alpha\rceil S} + \frac{1}{4}\cdot\frac{S\mathbb{P}^{\mathrm{A}}_{\mathrm{success}}}{S} \\ &= \frac{3}{4}\cdot\frac{\alpha}{\lceil\alpha\rceil}\mathbb{P}^{\mathrm{B}}_{\mathrm{success}} + \frac{1}{4}\cdot\mathbb{P}^{\mathrm{A}}_{\mathrm{success}}\end{aligned}$$

הכפל בקבועים $\frac{1}{4}, \frac{3}{4}$ הוא כפל ב-bandwidth היחסי של כל אחד מהם.

10.3 קוד לתיקון שגיאות

כשאנחנו שולחים פקטה, אנחנו בעצם שולחים רצף ביטים, ולכן הוא יכול להשתנות כתלות בפיסיקה של הסביבה. נרצה לספק מנגנון שייתן מענה לשתי בעיות:

- **זיהוי**.
- **תיקון**.

ה-Overhead של אלגוריתמים מסוג זה מוגדר על ידי $\frac{\text{Actual Message}}{\text{Original Message}} - 1 = \frac{\text{\#Redundant Bits}}{\text{\#Original Bits}}$

Repetition Code

אחת השיטות שבשימוש היא Repetition Code. לכל ביט שנשלח, אנחנו נוסיף עוד שני ביטים שיהיו זהים אליו. ככה, אם אחד הביטים ניזוק, נוכל לקבוע מה הביט הנכון לפי הכרעת הרוב. למשל, $0 \to 000 \to 010 \to 0$. במקרה זה נקבל

$$\text{overhead} = \frac{3N}{N} - 1 = 2$$

יחד עם זאת, אם ניזוקו שני ביטים, האלגוריתם יטעה. לאלגוריתם זה יש משמעות גאומטרית. היות שהוספנו שני ביטים עברנו ממימד 1 למימד 3. לכן ניתן להסתכל על קודקודי הקוביה התלת מימדית כתוצאות האפשריות של שלושת הביטים לאחר שהגיעו ליעד. תחילה, יהיה לנו ווקטור שיצביע על הווקטור הנכון. למשל ווקטור ממרכז הקוביה לקודקוד $(0,0,0)$ או ל-$(1,1,1)$. הרעש, יסובב את הווקטור לקודקוד אחר. כדי לקבוע מי הווקטור הנכון, נטיל אותו על $(0,0,0), (1,1,1)$ ונבחר את הווקטור שההטלה עליו היא הגדולה ביותר.

קוד זה הוא לינארי, היות שהוא מבצע פעולות לינאריות על המידע. בפרט, העלאת המימד היא כפל של הביט $[a]$ במטריצה $[1,1,1]$.

Parity Code

לקוד זה יש שתי צורות - חד מימדית ודו מימדית.

בפרוטוקול החד מימדי, אנחנו לוקחים את רצף הביטים ומוסיפים לו בסוף ביט שמייצג את הזוגיות של סכום כל הביטים. כלומר $0101 \to 01010$ ו-$100 \to 1001$. כשנקבל את ההודעה נשווה את הזוגיות שלה לביט ה-parity, ואז נדע אם הייתה שגיאה. הפרוטוקול לא יתקן את השגיאה, ואם היו שתי שגיאות, הוא עלול לא לדעת עליהן בכלל. נבחין כי

$$\text{overhead} = \frac{N+1}{N} - 1 = \frac{1}{N}$$

בפרוטוקול הדו מימדי אנחנו הופכים את ההודעה למטריצה בגודל $i \times j = N$ ומוסיפים לכל שורה ועמודה את ביט ה-parity החד מימדי שלה. לבסוף, אנחנו מוסיפים את ביט ה-parity של מה שהוספנו. כלומר

$$\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \\ \hline 1 & 1 & 0 \end{array}\right]$$

בצורה זו, אם הייתה שגיאה אחת, נוכל לתקן אותה על ידי זיהוי העמודה והשורה שבהן הייתה השגיאה. למעשה, הוא יכול לזהות עד 3 שגיאות, ולתקן רק שגיאה אחת. נבחין כי

$$\text{overhead} = \frac{i \times j + i + j + 1}{i \times j} - 1 = \frac{i + j + 1}{N}$$

4 שגיאות הוא לא יכול לזהות, למשל בדוגמא הבאה:

$$\left[\begin{array}{ccc|c} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 \end{array}\right] \rightarrow_{\text{errors}} \left[\begin{array}{ccc|c} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 \end{array}\right]$$

נראה מה האלגוריתם יזהה. העמודה הראשונה לא תשים לב, וגם לא השורה הראשונה. העמודה השנייה גם לא, והשורה השנייה גם לא.

נוכל לסכם את כל האלגוריתמים שראינו בטבלא הבאה:

קוד	מספר הביטים שמזהה	מספר הביטים שמתקן	תקורה
Repetition Code (3 times)	2	1	2
1D Parity	1	0	$\frac{1}{N}$
2D Parity	3	1	$\frac{i+j+1}{N}$

11 תרגול 4 - פרוטוקולים CSMA/CD

בהרצאה ראינו שפרוטוקול ALOHA לא מתמודד עם התנגשויות. כפתרון לבעיה זו הצגנו את פרוטוקול ה-CSMA, שהוסיף את התנאי לערוץ פנוי בעת שליחת ההודעה. על כן, נוכל להציע את שלושת האלגוריתמים הבאים.

1. $1-\text{presistent}$. אם הערוץ פנוי, נשלח את ההודעה. במקרה שהערוץ תפוס נחכה עד שהוא יתפנה ונשלח את ההודעה מיד לאחר מכן. **בעיה**: הסתברות גבוהה להתנגשויות כשהערוץ עמוס.
2. $\text{non}-\text{presistent}$ - אם הערוץ פנוי, נשלח את ההודעה. במקרה שהערוץ תפוס, נמתין זמן מסויים. **בעיה**: ניצול לא טוב בזמן שהערוץ יחסית פנוי.
3. $\text{p}-\text{presistent}$ - אם הערוץ פנוי נשלח את ההודעה בהסתברות p. אם הערוץ תפוס, נחכה עד שהוא יתפנה ואז נשלח את ההודעה בהסתברות p. זוהי למעשה קומבינציה ממשוקלת של שני הפתרונות הקודמים.

יחד עם זאת, ראינו כי $1-\text{presistent}$ לא מספיק, היות שיתכנו התנגשויות מהודעות שנשלחו בזמנים שונים. לכן הוספנו את זיהוי ההתנגשויות, שהוא פרוטוקול ה-CSMA/CD. עולות מכאן כמה שאלות. כיצד לידע את כולם על ההתנגשות? מי צריך לידע על ההתנגשות? כמה זמן לוקח לכל שאר הרכיבים לדעת על ההתנגשות? מתי נשלח את המידע שוב?

כפי שראינו בהרצאה, כל רכיב צריך לחכות לפחות 2T_{prop} לאחר שהוא שלח הודעה אחת - לכן תנאי הכרחי ברשת CSMA/CD הוא ש-$\text{T}_{\text{trans}} \geq 2\text{T}_{\text{prop}}$.

במקרה בו אנו ממתינים $(\text{non}-\text{presistent})$ אנו נרצה להשיג המתנה עבורה ככל שהיו יותר התנגשויות, ככה נמתין יותר. אבל נרצה שהזמן ייבחר להמתנה ייבחר ראנדומלית כדי שהרכיבים ברשת לא יתנהגו באותו אופן, וכל אחד יחכה בזמן אחר. הדרך בה עושים זאת ברשת, הוא באמצעות Exponential Backoff. בפרט, עבור קבוע c, לאחר שכשלנו להעביר frame בדיוק k פעמים, נמתין זמן $j \cdot T$, כאשר $j \sim \text{Unif}\left\{0, 1, \ldots, c^k - 1\right\}$. כאן T הוא הזמן הדרוש להעברת ביט אחד ברשת.

פרוטוקול CSMA/CD

פרוטוקול ה-CSMA/CD נותן מענה לשאלות אלה תחת מגבלות הרשת, ונותן גם המתנה אקספוננציאלית.

- במקרה של התנגשות, נשלח $\text{jam}-\text{signal}$ לרשת, ככה שהיא תדע על ההתנגשות.
- נמתין לפי ה-Exponential Backoff, ובדרך כלל נבחר $c = 2$.
- כאשר רכיב הצליח להעביר מידע בהצלחה, הוא מאפס את כמות הפעמים שכשל, כלומר את k.

ניתוח ה-goodput

כדי לנתח את $\eta_{\text{CSMA/CD}}$, נניח כמה הנחות מקלות.

Packets Switch

הנחות הפרוטוקול

- נניח כי אנחנו במקרה ה-$p-$presistent.
- אין המתנה אקספוננציאלית.
- נניח כי גודל כל Slot הוא $2\mathrm{T}_{\mathrm{prop}}$.
- לרכיבים תמיד יש מידע להעביר.
- בעת זיהוי ערוץ: אם הוא פנוי, נשלח הודעה בהסתברות p אחרת נחכה ל-slot הבא.
- בעת זיהוי התנגשות: אם נמצאה התנגשות, נפסיק להעביר מידע, וננסה שוב ב-slot הבא.

הנחות הרשת

- End to End Propagation $= \mathrm{E2EP} = \mathrm{T}_{\mathrm{prop}}$.
- מספר הרכיבים הוא N.
- זמן העברת Frame הוא $\mathrm{T}_{\mathrm{trans}}$.
- במקרה של התנגשות - כל ה-slot מבוזבז.

מכאן אנו מבינים שייתכנו שלושה מקרים. הראשון הוא מצב בו שני רכיבים התחילו לשלוח מידע בזמנים שונים, והם התנגשו לאחר זמן מסויים, והם ידעו על ההתנגשות רק לאחר $2\mathrm{T}_{\mathrm{prop}}$. המקרה השני הוא שהערוץ פנוי לגמרי, ואף רכיב לא מנצל אותו. המקרה האחרון הוא שרכיב זיהה ערוץ פנוי ומעביר בו מידע בהצלחה.

על כן, ההסתברות להעברה מוצלחת של מידע היא כמו בפרוטוקול ALOHA:

$$f(p) = \mathbb{P}[X_p = 1] = np(1-p)^{n-1}$$

היות שאנחנו מניחים אי תלות. על כן ההסתברות המקסימלית היא $\max_{p\in[0,1]} f(p) = \left(1-\frac{1}{n}\right)^{n-1} = S$, שזו ההסתברות להגעת פקטה אחת ב-Slot.

עתה, כדי לחשב את η עלינו להתחשב בשני המקרים. במקרה של הצלחה, לוקח לנו $\mathrm{T}_{\mathrm{trans}}$, במקרה של כשלון, לוקח לנו $2\mathrm{T}_{\mathrm{prop}}$. לכן היות שמספר ה-Slots עד הצלחה, מתפלג $\mathrm{Geo}(S)$, נקבל כי מספר ה-Slots עד הצלחה, כולל, יהיה $\frac{1}{S}$. על כן, מספר ה-Slots שבוזבזו הוא $\frac{1}{S}-1$ ולכן הזמן שבוזבז הוא $2\mathrm{T}_{\mathrm{prop}}\left(\frac{1}{S}-1\right)$. סך הכל, נקבל כי

$$\begin{aligned}\eta_{\mathrm{CSMA/CD}} &= \frac{\mathrm{T}_{\mathrm{trans}}}{\mathrm{T}_{\mathrm{trans}} + \left(\frac{1}{S}-1\right)2\mathrm{T}_{\mathrm{prop}}} \\ &= \frac{1}{1+2\cdot\left(\frac{1}{S}-1\right)\cdot\frac{\mathrm{T}_{\mathrm{prop}}}{\mathrm{T}_{\mathrm{trans}}}}\end{aligned}$$

נבחין כי כאשר $n\to\infty$ מתקבל כי $S=\frac{1}{e}$ ולכן $\eta_{\mathrm{CSMA/CD}} = \frac{1}{1+2\cdot\frac{\mathrm{T}_{\mathrm{prop}}}{\mathrm{T}_{\mathrm{trans}}}\cdot(e-1)}$. בפרט, כאשר $\mathrm{T}_{\mathrm{prop}} \ll \mathrm{T}_{\mathrm{trans}}$ נקבל כי $\eta_{\mathrm{CSMA/CD}}$ מאוד גדול, וזה הגיוני, היות שנקבל מעט זמן מבוזבז.

מסקנה 11.1. נשתמש ב-CSMA/CD כאשר $\mathrm{T}_{\mathrm{prop}} < \mathrm{T}_{\mathrm{trans}}$.

שאלות

שאלה נניח שנתונה רשת CSMA/CD עם רוחב פס של $\mathrm{B} = 10\mathrm{Mbps}$ ו-$\mathrm{T}_{\mathrm{prop}} = 2\cdot 10^8\mathrm{m/s}$. הוספנו רשת נוספת כך שהמרחק המקסימלי בינה לבין הרשת שלנו הוא 20Km, כאשר החיבור ביניהן בעל אותם פרמטרים. נניח שלרשת הראשונה קוראים A ולרשת השנייה B. האם ניתן להריץ CSMA/CD ברשת הכוללת, כאשר גודל הפקטה הוא 64bytes?

פתרון 4. עלינו לבדוק האם זמן השידור מקיים כי $\mathrm{T}_{\mathrm{trans}} \geq 2\mathrm{T}_{\mathrm{prop}}$. אם כן, אפשר. אחרת, אי אפשר. נחשב את הזמן

שלוקח לביט להגיע בין שתי הנקודות הכי רחוקות ברשת:

$$\tau = \frac{\text{Distance}}{\text{prop. speed}} = \frac{20\text{km}}{2 \cdot 10^8\text{m/s}} = 10^{-4}\text{s} = 100\mu\,\text{sec}$$

לכן, אנחנו רוצים שזמן השידור, יהיה לפחות $2\tau = 200\mu\,\text{sec}$. אבל הזמן להעברת ההודעה יהיה

$$\frac{\text{msg size}}{\text{bandwidth}} = \frac{8 \cdot 64}{10\text{mb/s}} = \frac{8 \cdot 64}{10^7\text{b/s}} = 51.2\mu\,\text{sec} < 2\tau$$

לכן התשובה היא שלא ניתן.

שאלה בנתוני השאלה הקודמת, מה השינוי הכי קטן שצריך לעשות כדי שזה יעבוד.

פתרון 5. היות שרכיבים פיסיים כמו מרחק, רוחב פס וזמן קשה לשנות, נשנה את גודל ההודעה. נרצה כי

$$\text{msg size}/\text{bandwidth} \geq 2\tau$$

כלומר

$$\begin{aligned}\text{msg size} &\geq 2\tau \cdot \text{bandwidth}\\ &= 2 \cdot 10^{-4}\text{s} \cdot 10^7\text{b/s}\\ &= 2 \cdot 10^3\text{bits}\\ &\approx 2 \cdot 2^{10}\text{bits} = 2 \cdot 2^7\text{bytes} = 256\text{bytes}\end{aligned}$$

אינטואיטיבית, נכפול ב-4 את הפקטה מהחישוב הקודם, ונקבל זמן העברה של $51.2 \cdot 4\mu\,\text{sec} \geq 2\tau$.

שאלה אם היינו מגדילים את רוחב הפס פי 10, איך הייתה משתנה התשובה שלנו?

פתרון 6. היינו מכפילים את גודל הפקטה פי 10.

שאלה נניח עתה שנתונה רשת של שלושה רכיבים A, B, C. הנמצאים אחד אחרי השני כל אחד במרחק 5km מהשני, כלומר $A \rightarrow_{5\text{km}} B \rightarrow_{5\text{km}} C$. נתון כי $\text{v}_{\text{prop}} = 6 \cdot 10^7 \frac{\text{m}}{\text{s}}, B = 3\frac{\text{mb}}{\text{s}}$. כדי שפרוטוקול ה-CSMA/CD יעבוד טוב, מה גודל הפקטה המינימלי ש-A צריך לשלוח ל-C?

פתרון 7. מתקיים כי

$$\text{T}_{\text{prop}} = \frac{\text{Distance}}{\text{v}_{\text{prop}}} = \frac{10\text{km}}{6 \cdot 10^7 \frac{\text{m}}{\text{s}}} = \frac{1}{6} \cdot 10^{-3}s$$

נסמן את גודל הפקטה ב-x ונדרוש כי $\text{T}_{\text{trans}} = \frac{x}{B} \geq 2\text{T}_{\text{prop}}$ אזי $x \geq \frac{2\text{T}_{\text{prop}}}{B} = 10^3\text{bit}$.

שאלה נניח ש-A שולח הודעה ל-B. כיצד זה היה משנה את התשובה?

פתרון 8. היות שגם C יכול לשלוח הודעות, A צריך לדעת על התנגשויות שמתרחשות עם C, ולכן T_{prop} נשאר אותו דבר, וסך הכל $x = 10^3\text{bit}$ נשאר אותו דבר.

שאלה נניח שיש רשת Slotted ALOHA עם $\text{T}_{\text{trans}} \geq 2\text{T}_{\text{prop}}$ כאשר כל Slot בגודל T_{trans}. נניח שאנו מוסיפים לרשת CMSA. איך ישתפר ה-goodput?

פתרון 9. היות שאנחנו ב-Slotted ALOHA, כל פעם שרכיבים ירצו לשדר ויבדקו אם הערוץ פנוי, הם יגלו שהערוץ פנוי, ולכן לא ישתנה שום דבר.

שאלה נניח שאנחנו מוסיפים CMSA/CD למערכת מהשאלה הקודמת. איך ישתפר ה-goodput?

פתרון 10. במקרה זה גם אם הייתה התנגשות, הם יחכו ל-slot הבא, ולכן ה-slot הנוכחי בכל מקרה בוזבז. כלומר גם עם המתנה, וגם בלי המתנה, אין באמת הבדל.

שאלה נניח שיש רשת Pure ALOHA עם $T_{trans} \geq 2T_{prop}$ כאשר כל Slot בגודל T_{trans}. נניח שאנו מוסיפים לרשת CMSA. איך ישתפר ה-goodput?

פתרון 11. במקרה זה אם מישהו מנסה לשדר, הוא יבדוק שאין כבר אנשים שמשדרים, ולכן ימנע התנגשות. כלומר זה שיפור.

שאלה נניח שהוספנו CMSA/CD. איך ישתפר ה-goodput ביחס לשאלה הקודמת?

פתרון 12. במקרה זה אם היו התנגשויות, אנחנו נפסיק לשלוח מידע, אבל גם נעבור למצב המתנה, וככה נשחרר את הערוץ להעברות מידע אחרות. לכן גם כאן יש שיפור.

12 תרגול 5 - פרוטוקול Spanning Tree (STP)

בהרצאה התחלנו לדבר על פרוטוקול ה-STP. עתה נרצה להציג כיצד בפועל בוחרים את מבנה העץ. ראשית אנחנו בוחרים את השורש, לפי ה-id המינימלי שהרשת מגלה בהודעות HELLO. לאחר מכן, אנחנו מחשבים את המרחק של כל switch מהשורש.

נסמן ב-RP צלע היוצאת מ-switch אל LAN שמחברת אותה לשורש וב-FP בפורט זמין שמחבר את ה-LAN ל-switch האידאלי. נבחין כי RP, FP ירכיבו את העץ.

ה-LAN שמחוברת ל-FP היא באחריות ה-Switch שממנו היא יוצאת. כאשר רכיב רוצה לשלוח הודעה, ייתכן שה-LAN אליה מגיעה ההודעה מכילה חיבורים לכמה switches, והם יתחרו מי מהם יוביל ל-RP, כלומר אל מי מהם יעביר הרכיב ברשת מידע. הדרך שבה הוא ייבחר, היא לפי שלושה קריטוריונים.

1. מרחק.
2. אם המרחקים שווים, אז לפי ה-id - הנמוך מנצח.
3. אם גם המזהים שווים וגם המרחקים שווים, אז לפי הפורט - פורט נמוך יותר מנצח.

הדרך שבה הם מעבירים את המידע של האלגוריתם אחד לשני היא באמצעות פקטת HELLO שמכילה את ה-RID (Root) , SID (ID) , DTR (Distance).

כלומר כדי לבנות את העץ, בהנתן מרחקים ושורש, אנחנו מחשבים FP לכל LAN לפי הפרוטוקול הנ"ל, ואז מחשבים את ה-RP לכל switch באופן דומה.

13 תרגול 6 - שכבה 3 ופרוטוקול ARP

אפשר לחשוב על פרוטוקול ה-IP שראינו בהרצאה כהפרדה היררכית של הרשת, עכשיו כשאנחנו רוצים לשלוח הודעה, אנחנו שולחים אותה למרכזיה שאחראית עלינו (נתב), שתעביר אותה הלאה. אם הנתב אחראי על ה-subnet של הכתובת, הוא יעביר את ההודעה ישירות ליעד, ואחרת, יעביר אותה ל-router הבא שאחראי על subnet אחר. על כן, אנחנו קוראים לנתב שלנו ה-Gateway שלנו, שכן הוא מקשר אותנו לתתי רשתות אחרות.

בנוסף, יש מוסכמה על כתובות IP מסויימות:

- 127.0.0.1 מסמן את המחשב שלנו ברשת ה-LAN.
- הכתובת של הנתב שלנו היא אחת מבין 192.168.1.1/10.0.0.1/172.16.x.x.
- 0.0.0.0 משמעותה - אני לא מחובר עדיין לרשת.
- 255.255.255.255 משמעותה - להעביר את ההודעות שלי ב-broadcast. זו כתובת יעד.

כתובת IP יכולה להנתן על ידי מנהל רשת. אך לרוב, כתובת מתקבלת בפרוטוקול DHCP, בו, כפי שהוצג בהרצאה, יש שרת המחלק לנו כתובות, ומראש מוגדר לו מרחב כתובות עליו הוא אחראי. הדרך בה הוא מחלק כתובת היא כדלקמן:

1. הלקוח ישלח הודעת DHCP DISCOVER שמשמעותה - אני רוצה כתובת IP, איזה שרת DHCP יכול להביא לי אותה?

(א) בחלק זה, ללקוח אין עדיין כתובת IP ולכן הוא צורח, כלומר מעביר את ההודעה לכל הרכיבים ברשת. את זאת הוא עושה על ידי קביעה של $src = 0.0.0.0$ ו-$dst = 255.255.255.255$.

(ב) כדי שהשרת ידע לזהות אותו, הוא כולל בהודעה transaction id.

2. השרת יענה לו ב-DHCP OFFER שמשמעותה - שמעתי, הנה כתובת שאתה יכול לקבל.

(א) הוא יעביר ל-$dst = 255.255.255.255$ עם transaction id דומה למה שקיבל, כך הלקוח ידע שההודעה מכוונת אליו.

(ב) השרת יכול להיות באותה subnet של הלקוח, בפרט באותו LAN, ולכן הודעת ה-broadcast מגיעה לשרת. אבל הוא יכול להיות גם בשרת DHCP בתת רשת אחרת, שאחראי על כמה מרחבי כתובת. במקרה זה, הנתב יעביר בהודעה שדה שיסמן איזה מרחב כתובות רלוונטי, והשרת יעביר כתובת בהתאם, כלומר התקשורת תהיה בין-רשתית.

3. הלקוח ישלח בקשת DHCP REQUEST - בה הוא יבקש את הכתובת שקיבל, וכך יאשר לשרת שהוא קיבל את ההודעה.

(א) בשלב זה, הלקוח עדיין מתקשר כ-$src = 0.0.0.0, dst = 255.255.255.255$.

4. השרת יענה לו ב-DHCP ACK שמשמעותה - קיבלתי, זו הכתובת שלך מעכשיו, עד שיעבור פרק זמן T.

(א) בשלב זה, הכתובת בפועל כבר של הלקוח, ולכן ההודעה תוחזר לכתובת dst של הלקוח, שהוא יודע אותה, כי קיבל אותה בשלב 2.

שלבים $1, 2$ אלה שלבים אופציונלים ויש פרוטוקולים שמדלגים עליהם (על ידי עדכון שני השלבים האחרים).

שרת ה-DHCP מהווה את המדריך שלנו לרשת בעת ההתחברות. הוא מספק לנו גם כתובת לשרת ה-DNS, את ה-$subnet - mask$ של ה-LAN ואת ה-default gateway (הנתב). ייתכן שיהיו לנו כמה שרתי DHCP, וככל הנראה המחשב ייבחר בהצעה של השרת הראשון.

פרוטוקול ARP (Address Resolution Protocol)

עד כה גילינו כיצד משיגים כתובת IP, אך דבר אחד לא ברור - כיצד נשיג כתובות MAC. הרי בסופו של דבר, מי שיעביר את ההודעה לרכיב ברשת הוא switch בשכבה 2 ברשת LAN, ולכן עלינו לתת לו כתובת MAC כדי שידע למי להעביר אותה - הוא לא מודע בכלל לכתובת IP, שכן הוא מסתכל על הבתים הרלוונטים רק לשכבה 2. הדרך שבה נעשה זאת הוא באמצעות פרוטוקול משכבה 2, ARP.

היות שאנחנו יודעים מהי כתובת ה-IP של היעד, נוכל לנצל זאת - כל רכיב ברשת יחזיק טבלא שתמפה כתובות IP לכתובת MAC. היות שכתובות IP משתנות, לכל רשומה בטבלא יהיה שדה ttl, ואם הרשומה עדיין חוקית, במקום לשלוח שאילתת ARP הוא יוכל להשתמש במידע בטבלא. אחרת, אם הרשומה איננה חוקית, או לא קיימת:

1. נשלח בקשת broadcast מסוג ARP Request שתכלול את ה-IP של היעד, ומשמעותה ברשת היא: "מהי כתובת ה-MAC של הרכיב המחזיק בכתובת ה-IP הזו?"
2. היעד ישלח לנו ARP Response שתכלול את כתובת ה-MAC.
3. במידה והרכיב לא נמצא ברשת הנ"ל, מי שיבצע את השאילתת ARP הוא בכלל ה-gateway ברשת של היעד, ואנו נצטרך להסיק את ה-MAC של הנתב באופן דומה.

כדי לצפות בטבלת ה-ARP במחשב שלכם (בלינוקס), הריצו $\text{arp} - \text{a}$. אם אנחנו שולחים הודעה למחשב שלא נמצא ברשת המקומית שלנו, אנחנו נצטרך את כתובת ה-MAC של הנתב, ונתחיל בזה שנבצע שאילתת ARP עם כתובת ה-IP של המחשב הזה. הנתב שיראה את זה יידע שהמחשב נמצא ברשת אחרת וימשיך בתהליך הלאה. מבחינתנו, כתובת ה-MAC של המחשב היא כתובת המחשב של הנתב.

Hop By Hop

כאשר אנחנו רוצים להעביר פקטה בין LAN אחת לאחרת, מה שקורה הוא שאנחנו שולחים את הפקטה ל-Router שלנו. הוא ישנה את ה-src mac של ההודעה, ושל היעד, אבל ישמור את ה-src ip, dst ip, וכך הוא יוכל להעביר את ההודעה הלאה לנתבים של הרשת המתאימה. הוא יודע לאן להעביר את ההודעה לפי ה-subnet. כאמור, יש לו כמה פורטים ולכל אחד subnet אחרת. בכל שלב, ייתכן שישלח שאילתת arp.

הערה 13.1. כדי שלא נצטרך לזכור כתובות IP של כל רכיב ברשת, נשתמש בפרוטוקל DNS שראינו בהרצאה.

14 תרגול 7 - העברה אמינה של מידע

המטרה שלנו היא לספק תקשורת אמינה (Reliable Transport), במובן שאם הודעה לא הגיעה ליעד, נדאג לשלוח אותה שוב עד שהיא תגיע. נציע את האלגוריתם Stop And Wait:

Stop And Wait	
המקור	**היעד**
1. נשלח פקטה ונפעיל טיימר.	1. נשלח הודעת ACK אם הגיעה פקטה תקינה.
2. נחכה שיעבור זמן T_{out}, או להגעת פקטת ACK/NACK.	2. נשלח NACK (או שנתעלם) אם הפקטה לא תקינה.
3. אם הגיעה NACK או שעבר הזמן, נאפס את הטיימר ונשלח את הפקטה שוב.	
4. אם התקבל ACK, נאפס את הטיימר ונשלח את הפקטה הבאה.	

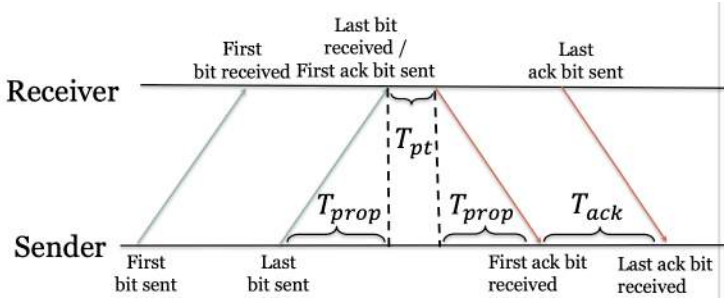
בעיה מרכזית בפרוטוקול זה, הוא שבמצב בו פקטת ACK נשלחה אך לא הגיעה, אנחנו נשלח פקטה חדשה, למרות שהיעד מצפה לקבל את פקטה מספר 2.

הפתרון לבעיה הוא מספור של הפקטות של המידע.

אך גם כאן יש בעיה, מה קרה עם ה-ACK הגיעה, אך לאחר ה-timeout, לפני שהרכיב הספיק לשלוח את הודעה מספר 1 פעם נוספת (כלומר היא בדרך ליעד, אך עוד לא הגיעה)? במקרה זה נשלח כבר את הודעה מספר 2 לפני שקיבלנו באמת אישור.

הפתרון לבעיה הוא הוספת מספור של פקטות ה-ACK, שהגיעו מהיעד, שכן אז נדע האם להתייחס אליהן או לא, בהתאם למספור של הפקטה ששלחנו עתה.

כמו כן, עלינו לדעת כיצד לקבוע את T_{out}. כדאי להתחיל בלמצוא את הזמן המינימלי הדרוש להמתנה. ראשית עלינו להמתין להגעה של הביט האחרון ששלחנו ליעד, שזה T_{prop}, לאחר מכן נצטרך להמתין להגעה של הביט הראשון של פקטת ה-ack שזה גם T_{prop} וסך הכל קיבלנו $2T_{prop}$. אבל במקרה שלנו, זה לא יספיק, שכן נצטרך להמתין לזמן שייקח לפקטת ה-ack להגיע במלואה. כלומר נצטרך להמתין $2T_{prop} + T_{ack}$. יחד עם זאת, ייתכן שנצטרך להתחשב בזמן בין שליחת הביט הראשון של ה-ack לבין ההגעה של הביט האחרון של ההודעה שלנו, זמן זה יסומן ב-T_{pt}:



איור 39: פרקי הזמן בעת שליחת הודעה וקבלת ack

סך הכל אנחנו מקבלים כי

$$T_{out} \geq 2T_{prop} + T_{ack} + T_{pt}$$

היות שאת T_{prop} אנחנו יודעים לחשב בהנתן פרמטרים של הרשת, נוכל לחשב זמן זה. יהיו פעמים שנזניח את T_{pt}.

שאלות

שאלה נתונות שתי יחידות קצה המתקשרות בפרוטוקול Stop And Wait, כאשר נתון שההסתברות שפקטה תכשל היא p. מהו ה-good put במערכת? נתון ש-$\mathrm{T_{ack}}$ זניח.

פתרון 13. נסמן ב-X את מספר הפעמים הדרושות לשליחת פקטה עד למצב של הצלחה, היות שההסתברות להצלחת פקטה ספציפית היא $(1-p)^2$, שכן דרושה הצלחה לפקטה עצמה, ול-ack, נקבל כי $X \sim \mathrm{Geo}\left((1-p)^2\right)$. מכאן $S = \mathbb{E}[X] = \frac{1}{(1-p)^2}$, ולכן הזמן הכולל הוא $S\left(\mathrm{T_{packet}} + \mathrm{T_{pt}}\right)$ (נתון לנו ש-$\mathrm{T_{ack}}$ זניח) ולכן

$$\eta = \frac{\mathrm{T_{packet}}}{S\left(\mathrm{T_{packet}} + \mathrm{T_{pt}}\right)}$$

נעיר כי במקרה זה ביצענו ניתוח הסתברותי ל-total time, בעוד ביצענו ניתוח דטרמינסטי לזמן ההמתנה. זה שונה ממה שעשינו עד כה - $\eta = \frac{\mathbb{E}[\mathrm{T_{success}}]}{\mathrm{T_{total}}}$, אך זה שקול (לא נראה למה).

שאלה בנתוני השאלה קודמת, מהי ההסתברות לכשלון של פקטה באורך L, כאשר במקרה של כשלון של חלק אחד מבין L החלקים, כל L החלקים ישלחו שוב? מהו ה-goodput?

פתרון 14. במקרה זה, אנחנו לא צריכים להתייחס ל-ack, אלא רק לפקטה עצמה, כי הכשלון הוא של הפקטה הנתונה. לכן $1-(1-p)^L$, שכן ההסתברות להצלחה היא $(1-p)^L$. במקרה זה, נבחין כי

$$\mathrm{T_{success}} = \begin{cases} L \cdot \mathrm{T_{packet}} & (1-p)^L \\ 0 & 1-(1-p)^L \end{cases}$$

מכאן, נקבל כי ה-goodput הוא

$$\begin{aligned}
\eta &= \frac{\mathbb{E}\left[\mathrm{T_{success}}\right]}{L\mathrm{T_{packet}} + \underbrace{2\mathrm{T_{prop}}}_{\mathrm{T_{out}}}} \\
&= \frac{\mathbb{P}\left[\text{window failed}\right] \cdot 0 + \mathbb{P}\left[\text{window succeded}\right] \cdot L \cdot \mathrm{T_{packet}}}{L\mathrm{T_{packet}} + 2\mathrm{T_{prop}}} \\
&= \frac{L \cdot \mathrm{T_{packet}}\,(1-p)^L}{L\mathrm{T_{packet}} + 2\mathrm{T_{prop}}}
\end{aligned}$$

שאלה ננסה לשפר את הפרוטוקול מהשאלה הקודמת, על ידי כך שנשלח רק את ההודעות לאחר ההודעה שנכשלה, ולא את מי שנשלח לפני כן. כלומר, B (היעד) יחזיק buffer שהוא ימלא בפקטות, כאשר ה-buffer מלא עם פקטות לפי הסדר, הוא יעביר אותו. על כן, אם הגיעו פקטות $1, 2, 4$, הוא יחיכה לפקטה מספר 3 ורק אז ישלח. נגדיר את זמן המחזור להיות הפרש הזמנים בין תחילת החלון הנוכחי לתחילת החלון הבא. מהו זמן המחזור?

פתרון 15. העברה של כל חלק בפקטה היא $\mathrm{T_{packet}}$ לכן סך הכל $L \cdot \mathrm{T_{packet}} + \mathrm{T_{out}}$.

שאלה בנתוני השאלה הקודמת, מה ההסתברות שבזמן המחזור, אף פקטה לא תתקבל, כלומר אף פקטה לא תועבר ל-B?

פתרון 16. זה יקרה אך ורק אם הראשונה תיפול, ולכן בהסתברות p.

שאלה בנתוני השאלה הקודמת, מהו ה-goodput?

פתרון 17. נבחין כי אם פקטה מספר i נכשלה, אנחנו נמתין $(i-1)\,\mathrm{T}_{\text{packets}}$, שכן נמתין לכל הפקטות שהצליחו. זה יקרה בהסתברות $(1-p)^{i-1}\,p$, שכן זה אומר ש-$i-1$ הפקטות הקודמות הצליחו, בעוד הפקטה ה-i נכשלה, ולכן נקבל כי

$$\mathrm{T}_{\text{success}} = \begin{cases} L\mathrm{T}_{\text{packet}} & (1-p)^{L} \\ (L-1)\,\mathrm{T}_{\text{packet}} & (1-p)^{L-1}\,p \\ \vdots & \vdots \\ 2\mathrm{T}_{\text{packet}} & (1-p)^{2}\,p \\ \mathrm{T}_{\text{packet}} & (1-p)\,p \end{cases}$$

מכאן

$$\begin{aligned} \mathbb{E}\left[\mathrm{T}_{\text{success}}\right] &= \mathrm{T}_{\text{packet}} \sum_{j=1}^{L} j\,(1-p)^{j}\,p \\ &= p \cdot \mathrm{T}_{\text{packet}} \sum_{j=1}^{L} j\,(1-p)^{j} \end{aligned}$$

ומכאן $\eta = \frac{p\cdot \mathrm{T}_{\text{packet}} \sum_{j=1}^{L} j(1-p)^{j}}{L\cdot \mathrm{T}_{\text{packet}} + \text{‘}\mathrm{T}_{\text{out}}}$.

שאלה בנתוני השאלה הקודמת, כמה פקטות נצטרך לשלוח כדי שפקטה $L-1$ תגיע בהצלחה?

פתרון 18. נסמט ב-X_i את מספר הפעמים הנ"ל עבור פקטה i. נסמן ב-Y את מספר הפעמים הדרוש להצלחה עבור $L-1$. מכך שאנחנו שולחים כל פעם הודעות רק לאחר הכשלון של ההודעה, ולא את כולן. נקבל כי כדי שהודעה $i+1$ תשלח, דרוש שהודעה i נשלחה. כלומר $Y = \sum_{i=1}^{L-1}\left[X_i - 1\right]$, שכן בניסיון האחרון של כל הודעה, יש הצלחה, ולכן נשלחה ההודעה הבאה. מכאן $\mathbb{E}\left[Y\right] = \sum_{i=1}^{L-1}\mathbb{E}\left[X_i\right]$. נבחין כי $X_i \sim \mathrm{Geo}\left(1-p\right)$ ולכן $S = \mathbb{E}\left[X_i\right] = \frac{1}{1-p}$. על כן

$$\mathbb{E}\left[Y\right] = (L-1)\,(S-1) + 1 = (L-2)\,(S-1) + S$$

שכן בניסיון האחרון של $L-1$ אנחנו כן רוצים להחשיב את ההצלחה.

15 תרגול 8 - GBN (Go Back N) ו-SR (Selective Repeat)

פרוטוקול GBN

כדי לשפר את פרוטוקול ה-S&W שראינו בתרגול הקודם, נציע את התוספת הבאה. נגדיר $\mathrm{T_{out}}$ כמו קודם, ונשמור חלון. החלון יהיה בגודל k יחידות זמן, ונקדם אותו לפי הודעות ack שמגיעות. כלומר נבצע את התהליך הבא:

1. נשלח הודעה בכל יחידת זמן ונפתח סטופר ל-$\mathrm{T_{out}}$ לאחר כל שליחה.
2. אם הגיע ack, נקדם את החלון לפקטות שעוד לא הגיע עבורן ack.
3. אם עבר ה-timeout ולא הגיע ack, אנחנו נשלח את ההודעה המתאימה, ואת כל ההודעות שהגיעו אחריה.
4. כל הודעה שנשלחת מכילה שני רכיבים - המספר הסידורי שלה בתוך החלון, והמספר הסידורי שלה מבין כל ההודעות שאנחנו רוצים לשלוח. למשל אם גודל החלון הוא 5, ומספר ההודעות הכולל הוא 20, שלחנו את ההודעות $1,2,3,4$, קיבלנו אשרור ל-1, אבל עבר ה-timeout של 2. אז נשלח שוב את הודעה 2 עם המספר הסידורי בתוך החלון 1, ועם המספר הסידורי בתוך סדרת ההודעות 2. כאשר השימוש במספר הסידורי בתוך הסדרה נועד למנוע מקרה בו כל ה-ack נכשלו, ושלחנו שוב את כל הסדרה, אך היות שלא כללנו את המספר הסידורי בסדרה כולה, זה נראה כאילו שלחנו סדרה חדשה, שכן המספורים mod גודל החלון, יתחילו שוב מ-0, והיעד קיבל את כל הסדרה הקודמת.
5. כאשר היעד יקבל שוב את כל ההודעות, הוא לא ישלח הודעת ack לכל הודעה בנפרד, אלא ישלח הודעת ack להודעה אחת יותר מההודעה האחרונה שנשלחה, כאינדיקציה לכך שהוא קיבל את כל ההודעות החדשות. לכן במקרה זה נקדם את החלון כמה צעדים. כלומר הוא אומר לנו - אני מצפה לקבל את ההודעה הבאה.

על כן בכל פעם שהודעה לא מקבלת ack, אנחנו שולחים מספר קבוע של הודעות שוב, ומכאן השם GBN. זה אומר שאם יש לנו חלון בגודל k, נצטרך $\lceil \log_2 k \rceil$ ביטים בכל הודעה בשביל המספר הסידורי של החלון, אך אנחנו רוצים כמה שפחות, כדי שההודעות יהיו קטנות. בפרט, אנחנו צריכים שגודל החלון יהיה לכל הפחות $\mathrm{T_{out}}+1$, כדי שנוכל לעלות על הודעה שנפלה, לפני שהחלון עבר אותה. נבחין כי אנחנו לא באמת צריכים לשלוח את המספר הסידורי של כל פקטה בין כל הפקטות, אלא מספיק לשמור במקום k מספרים סידוריים, $k+1$ מספרים סידוריים, כאשר המספר ה-$k+1$ יהווה אינדיקציה ליעד שלא הגיע כל החלון, ולכן עליו לזרוק את הפקטות הקודמות. כלומר עלינו לשמור $\lceil \log_2 (k+1) \rceil$ ביטים.

שאלה נניח שלאחר שפקטה נפלה אנחנו שולחים אותה L פעמים. נניח שההסתברות לכשלון של פקטה היא p. מה ההסתברות להצליח לשלוח אותה כאשר נשלח אותה שוב בדיוק k פעמים?

פתרון. ההסתברות לאי הצלחה בניסיון הראשון היא p. לאחר מכן, אנחנו שולחים את ההודעה L פעמים, וההסתברות שכולן נכשלו היא p^L. אנחנו חוזרים על זה $k-1$ פעמים, ואז רוצים שבניסיון ה-k לפחות הודעה אחת הגיעה ומקבלים $p\cdot\left(p^L\right)^{k-1}\cdot\left(1-p^L\right)$. מכאן תוחלת הזמן היא $\sum_k \mathrm{T_{wait,k}}\cdot p_k$. נחשב את $\mathrm{T_{wait,k}}$. מתקיים כי

$$\begin{aligned}
\mathrm{T_{wait,0}} &= \mathrm{T_{pkt}} \\
\mathrm{T_{wait,1}} &= (\mathrm{T_{out}} + L\cdot \mathrm{T_{pkt}}) \\
\mathrm{T_{wait,k}} &= (\mathrm{T_{out}} + L\cdot \mathrm{T_{pkt}})\cdot k
\end{aligned}$$

ומכאן $\mathrm{T_{avg}} = \sum_{k=0}^{\infty} \mathrm{T_{wait,k}}\cdot p_k = \mathrm{T_{pkt}} + \frac{p\cdot(\mathrm{T_{out}}+L\cdot\mathrm{T_{packet}})}{1-p^L}$. על כן $\mathrm{goodput} = \frac{\mathrm{T_{packet}}}{\mathrm{T_{avg}}} = \frac{1-p^L}{1-p^L+p(a+L-1)}$ כאשר $a = \frac{\mathrm{T_{out}}+\mathrm{T_{pkt}}}{\mathrm{T_{pkt}}}$.

פרוטוקול SR (Selective Repeat)

החסרון המרכזי של GBN הוא שהוא זורק כל פקטה שהגיעה אחרי פקטה שנשלחה שוב. כדי למנוע זאת, נאפשר לו להחזיק buffer, שישמור את הפקטות שהגיעו, וכאשר פקטה קודמת שלא הגיעה נשלחה שוב, הוא ידחוף אותה אל תוך ה-buffer

במקום המתאים. במקרה שבו הפקטה כבר נמצאת ב-buffer הוא יתעלם ממנה, וישלח ack, רק שהודעה זו תסמן את הגעתה של הודעה i בלבד, ולא של הודעות נוספות, בשונה מ-GBN.

במקרה זה, גם ליעד יהיה חלון באותו גודל. כאשר בכל פעם הוא יקדם את החלון, רק כאשר החלון הקודם התמלא. במקרה זה הוא יקדם את החלון ל-sequence number הצפוי הבא.

כמו קודם, עולה השאלה כיצד למספר את הפקטות. אם נמספר רק לפי גודל החלון $k = 4$, עם $2k - 1 = 7$ מספרים סידוריים, אנו עלולים להגיע לבעיה. למשל עבור $\mathrm{T_{out}} = 8$ שלחנו את הודעות $1, \ldots, 4$, וה-ack של 1 נכשל. לכן אנחנו נשלח שוב את 1 אבל לא בהכרח עכשיו, אלא רק לאחר ה-timeout. נעביר את פקטות $5, 6, 7$. עתה, השולח יכול לבחור אם להעביר את 1 או את 8. היות שהמספר הסידורי של $8 \mod 7 = 1$, היעד לא ידע לומר אם מדובר בשליחה מחדש של פקטה, או בפקטה חדשה. מכאן אנחנו מסיקים כי דרושים $2N$ מספרים סידוריים, ו-$\lceil \log_2 2N \rceil$ ביטים בכל פקטה.

נחשב את ה-goodput של הפרוטוקול. מתקיים כי $\text{goodput} = \frac{\mathrm{T_{pkt}}}{\mathbb{E}[\mathrm{T_{waste}}]}$ נבחין כי הזמן המבוזבז הוא $\mathrm{T_{pkt}}$ שכן ב-timeout שאנחנו מחכים, שולחים פקטות אחרות. לכן הזמן היחיד שאנחנו מבזבזים הוא בפקטה הראשונה שאנחנו שולחים. היות שההסתברות שפקטה נכשלה היא $1 - p$ (נניח שההסתברות להצלחה היא p), היות שמספר הניסיונות להצלחה זה משתנה מקרי גאומטרי, אנחנו מקבלים כי $\mathbb{E}\left[\mathrm{T_{waste}}\right] = \frac{1}{1-p} \cdot \mathrm{T_{pkt}}$ לכן $\text{goodput} = \frac{\mathrm{T_{pkt}}}{\frac{1}{1-p} \cdot \mathrm{T_{pkt}}} = 1 - p$. אנליזה אחרת שיכולנו לעשות, הוא להסתכל על חלון זמן עבור פקטה יחידה, שכן כפי שאמרנו קודם, זה מספיק. אזי הזמן המבוזבז הוא $(1-p)\,\mathrm{T_{pkt}}$ ולכן $\text{goodput} = \frac{(1-p)\mathrm{T_{pkt}} + p \cdot 0}{\mathrm{T_{pkt}}} = 1 - p$.

16 תרגול 9 - Routing

כשלמדנו על שכבה 2 אמרנו שנתבים מסוגלים להעביר מידע אחד לשני על ידי $hop-by-hop$, אך התעלמנו מהמקרה בו יש לנו מעגלים. בפרט, אנחנו צריכים העברת מידע אופטימית, ובשביל זה צריך גם מסלולים אופטימלים, כלומר לכל נתב נרצה להחזיק טבלא שתמפה subnet ל-hop אופטימלי. לשם כך יש לנו שני אלגוריתמים, שראינו בהרצאה:

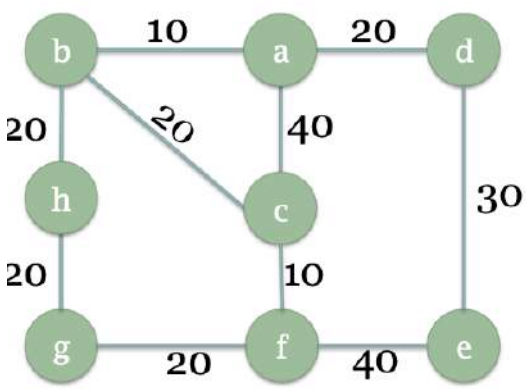
1. גלובלי: LS, בו יש תעבורה גבוהה יותר בעדכון המידע, שכן כל קודקוד מודע לכל המשקולות בגרף ולכן צריך להעביר מידע לכל הקודקודים. פרוטוקול שמשתמש באלגוריתם זה הוא ה-OSPF. הדרך בה מוצאים את המסלולים האופטימלים כאן היא באמצעות Dijkstra.
2. לוקאלי: DV, בו יש תעבורה נמוכה יותר בעדכון מידע, שכן כל קודקוד מודע למשקלים של השכנים שלו. יש לכך של $count-to-infinity$. פרוטוקול שמשתמש באלגוריתם זה הוא ה-RIP. הדרך בה מוצאים את המסלול האופטימליים כאן היא באמצעות האלגוריתם של Bellman Ford בצורה מבוזרת, כפי שראינו בשכבה 2.

> **!** אנקטודה לא קשורה: Dijkstra הוביל מאבק כנגד הפקודה goto line שעוד קיימת ב-c בטענה שהיא מובילה לתכנות גרוע ולבאגים רקורסיביים.

אלגוריתם ה-Distance Vector ב-RIP

כל קודקוד יחזיק טבלא שממפה מרחקים מקודקודים אחרים ברשת, כאשר היא מאותחלת להיות ∞ עבור קודקודים שאינם שכנים שלו, והמשקל של הצלע המחברת ביניהם, במקרה שמדובר בקודקוד שכן.

בכל איטרציה, אנחנו מקבלים מידע מהשכנים שלנו, ורושמים זאת בטבלא, ואז מעדכנים את המרחקים שלנו לפי העדכון ואלגוריתם Bellman Ford. כשנסיים, נשלח זאת לשכנים שלנו, עד שלבסוף נגיע להתכנסות. התכנסות מתרחשת כאשר לא השתנה שום דבר מהטבלא הקודמת.



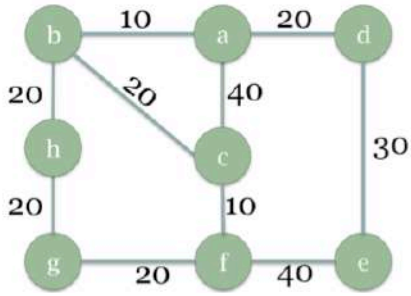
Node a	a	b	c	d	e	f	g	h
a	0	10	40	20	∞	∞	∞	∞
b	∞	∞	∞	∞	∞	∞	∞	∞
c	∞	∞	∞	∞	∞	∞	∞	∞
d	∞	∞	∞	∞	∞	∞	∞	∞

Node b	a	b	c	d	e	f	g	h
b	10	0	20	∞	∞	∞	∞	20
a	∞	∞	∞	∞	∞	∞	∞	∞
c	∞	∞	∞	∞	∞	∞	∞	∞
h	∞	∞	∞	∞	∞	∞	∞	∞

איור 40: דוגמא לגרף עם משקלים וטבלאות הניתוב ההחלתיות של a, b. התהליך עצמו של מילוי הטבלאות הוא מייגע.

לבסוף כשאנחנו מקבלים טבלת מרחקים, נמיר אותה לטבלת העברות באופן הבא:

Node a	a	b	c	d	e	f	g	h
a	0	10	30	20	50	40	50	30
b	10	0	20	30	60	30	40	20
c	30	20	0	50	50	10	30	40
d	20	30	50	0	30	60	70	50

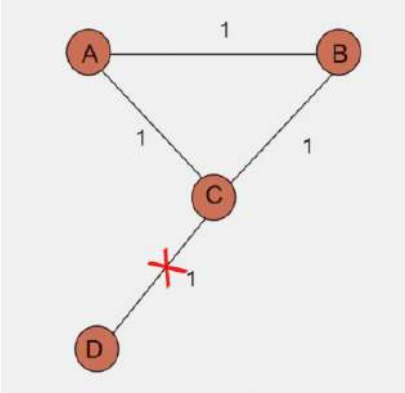


Dest.	Via	Cost
b	b	10
c	b	30
d	d	20
e	d	50
f	b	40
g	b	50
h	b	30

איור 41: טבלת העברות - מעבירים לשכן שהוא הקודקוד הבא במסלול הקצר ביותר

Poisoned Reverse

ראינו את בעיית ה-$count-to-infinity$. דרך אחת לפתור אותה היא באמצעות Poisoned Reverse, בו כאשר יש לנו מסלול מ-a ל-c שמסתמך על b, כשאנחנו שולחים את v_a ל-b, נציין ב-v_a של-b אסור להסתמך על a כדי להגיע ל-c. בפרט, נשים בכניסה v_a^c אינסוף כאשר אנחנו שולחים אותו ל-b, ואז הוא יידע לא להסתמך על a בשליחת מידע ל-c. כך נפטר מהמעגל המיותר. יחד עם זאת, זה לא יפתור את כל הבעיות.



איור 42: נתון הגרף שבאיור כאשר כל הרשת התכנסה. לפתע נופלת הצלע $\{D,C\}$, ונדרש עדכון ברשת.
כאשר החיבור נשבר, C מסמן את D כלא נגיש ומודיע על כך ל-A ו-B. נניח ש-A לומד על כך ראשון. אזי הוא חושב שהמסלול הכי טוב שלו ל-D, עובר דרך B, כי דרך C הוא לא נגיש. A מסמן את D כבלתי נגיש ומודיע ל-B, היות שהמסלול שלו ל-D מסתמך על B, וכך עובד ה-poisoned reverse. כמו כן, הוא מסמל מסלול בגודל 3 ל-D דרך B.
עכשיו C חושב ש-D נגיש דרך A במסלול באורך 4, ומודיע על כך ל-B.
B חושב ש-D נגיש במסלול באורך 5 דרך C ומודיע על כך ל-A, שמודיע על כך ל-C.
קיבלנו לולאה אינסופית שלא תעצר. הבעיה כאן שהתלות בין המסלולים היא לא רק בין שכנים, אלא בעוד רמה, ולכן poisoned reverse לא מספיק.

אלגוריתם ה-Linked State ב-OSPF

ביצענו הרצה של דייקסטרה על הגרף מהדוגמא הקודמת. למדנו עליו בהרחבה בקורסים קודמים ולכן מצאנו לנכון לא לצרף הרצה זו.

17 תרגול 10 - Traffic Engineering

ראינו שתי שיטות לקבלת זרימה "טובה" ברשת:

1. Minimum Congestion, שמוגדר על ידי $\min \max_e \frac{f_e}{c_e}$.
2. Maximim Flow, שמוגדרת על ידי $\max \sum_v f_v$.

הדרך שבה אנחנו עושים זאת, היא באמצעות תכנון לינארי. למשל עבור $\mathrm{Max-Flow}$:

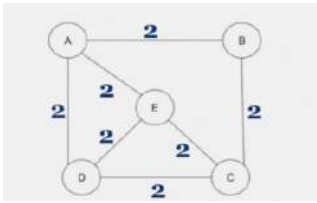
$$\max \sum_{v:(s,v)\in E} f(s,v)$$
$$\text{s.t}$$
$$\forall v \in V \setminus \{s,t\} : \sum_{u:(u,v)\in E} f(u,v) = \sum_{w:(v,w)\in E} f(v,w)$$
$$\forall u,v \in E : f(u,v) \le c(u,v)$$
$$\forall u,v \in E : f(u,v) \ge 0$$

שיפור לאלגוריתם - ECMP

כפי שראינו בהרצאה, נחלק את העומס על ה-nexthops, כאשר פקטות עם Header זהה ישלחו לאותו nextHop, לפי ה-Hash. נדגיש שה-Hash לא ייתן בהכרח ערך שמתאים לאינדקס של nextHop, לכן אפשר להפעיל מודולו.

שאלות

שאלה נניח שיש מקור, יעד ואילוצים $(A,C,5),(C,E,3)$. בגרף הבא. מהי הזרימה אופטימלית?



איור 43: הגרף בשאלה

פתרון. נניח תחילה שהקיבולים אינסופיים. אזי הזרימה האופטימלית שעונה על האילוצים היא

$$(A,B,2.5),(A,D,2.5),(D,C,2.5),(B,C,2.5),(E,C,3)$$

עתה נפתור את השאלה. בפרט חתך מינימלי הוא $(B,C),(A,E,D)$, עם זרימה 6. לכן זו גם הזרימה המקסימלית ברשת, לכן נדאג ש-A ישלח 4 ל-B,D ו-C ישלח 2 ל-E.

שאלה בנתוני השאלה הקודמת, האם קיימות משקולות לצלעות, כך שהפתרון ל-$\mathrm{Max-MCF_{OSPF/ECMP}}$ הוא הפתרון האופטימלי? אם כך, מצאו השמה כזו והסבירו איך זה מגיעים לפתרון האופטימלי, אם אין, הסבירו למה.

פתרון. אינטואיטיבית, אנחנו צריכים להכריח את Dijkstra לתת את המסלולים דרכם עברה הזרימה מהשאלה הקודמת. לכן נבחר את המשקולות הבאות:

$$w(A,B)=1, w(A,D)=1, w(A,E)=\infty$$
$$w(D,E)=\infty, w(D,C)=1, w(B,C)=1$$

ואכן נקבל את אותם מסלולים, והזרימה שתתקבל היא אכן אותה אחת מהשאלה הקודמת.

שאלה עכשיו נניח שהאילוצים הם $(B,D,5),(C,E,3)$. מהו הפתרון האופטימלי עבור בעיית ה-$\mathrm{MinCong}-\mathrm{MCF}_{\mathrm{OPT}}$? מצאו זרימה שמשרה פתרון זה.

פתרון. נשתמש בשיקולים של חתך, עם הקיבול הקטן ביותר. ראינו בשאלה קודמת שהחתך המינימלי בעל קיבול 6. לכן כל הזרימה עוברת דרך החתך, לכן כדי לקבל עומס מינימלי, הזרימה תצטרך לעבור באופן שווה בין צלעות החתך. חתך אופטימלי למשל הוא (B,C), והזרימה העוברת דרך החתך היא 6, והזרימה הכוללת שאנחנו צריכים היא 8, לכן סך הכל נקבל שהפתרון האופטימלי הוא $\frac{8}{6}$, כלומר הצלע הכי עמוסה בגודל $\frac{4}{3}$. בפרט, נקבל את הזרימה $\left(B,A,\frac{4}{3}\right),\left(C,E,\frac{4}{3}\right),\left(B,C,\frac{4}{3}\right)$.

18 תרגול 11 - TCP & NATs

TCP

הכרנו שני וריאציות של TCP - TCP Reno, TCP Tahoe. נסכם בקצרה את האלגוריתם לפיהם הם פועלים:

```
State = SS; W = 1;
SSThreshold = ? // threshold for enter CA

while True:
        for every ACK:
                if State == SS:
                        W += 1
                else: // state == CA
                        W += 1/W // (additive increase)

        for every 3 dupacks:
                SSThreshold = W/2
                State = SS, W=1   // if TCP Tahoe
                State = CA, w=W/2 // IF TCP Reno

        // multiplicative decrease
        for every timeout:
                SSThreshold = W/2
                State = SS, W = 1

        if W>=SSThreshold: State = CA
```

Network Address Translation (NAT)

בעבר דיברנו על בעיית החלוקה הלא הוגנת של מרחב הכתובות, ואמרנו ש-IPv6 נותן מרחב כתובות גדול יותר. יחד עם זאת, יקח זמן עד שהרשת תעבור ל-IPv6, וכפתרון אלטרנטיבי הוצעה טכנולוגיית ה-NAT. הרעיון הוא, שיהיה שרת אחראי על רשת מקומית, ויחלק לה כתובות שזמינות לרשת החיצונית, אך היא לא תדע שהוא חילק אותן, וכן יאפשר תקשורת בין רכיבים מתוך הרשת לרכיבים חיצוניים. הסיבה שלא עוברים ל-IPv6 היא שצריך לעדכן נתבים לקרוא את ה-header המתאים לפרוטוקול, וזה דבר שצריך לעשות ידנית. יש הרבה נתבים, ורכיבים כאלה, ולכן זה לא קרה עד היום.
הדרך שנעשה זאת היא שכאשר רכיב פנימי ירצה לתקשר עם רכיב חיצוני, הוא יפתח ערוץ עם כתובת ה-IP שלו ופורט מתאים, אבל בכל פעם שישלח פקטה, היא תגיע לשרת ה-NAT, שיפתח ערוץ עם כתובת ה-IP שלו עצמו, ופורט ייעודי, וכך ישנה את הפקטות שהוא מקבל מהרכיב הפנימי להכיל את ה-IP של עצמו ואת הפורט, ככה שהרכיב החיצוני יוכל באמת לזהות את הפקטה, ולהחזיר תשובה. זה יראה כך:

נבחין כי יש לזה גם פונקציונליות של אבטחה, שכן נעדיף לעיתים לא לחשוף את כתובת ה-IP הפנימית של שרת מסויים, לשאר הרשת, כדי למנוע התקפות כלשהן. למעשה, היום מתייחסים ל-NAT ככלי אבטחה לא פחות מכלי לפתרון בעיית הכתובות.