פרויקט מסכם – חלק ג

1. נניח ויש לקוח בשם BOB שמחובר כבר לצ'אט, וכעת ALICE מעוניינת להתחבר לצ'אט ולשלוח הודעה ל BOB. גם נניח שכל הלקוחות והשרת באותה תת רשת.

אני מזכיר שלקוח מתקשר עם השרת דרך שני סוקטים – אחד עבור שליחת בקשות וקבלת תגובות מהשרת, והשני עבור עדכונים חמים מהצ'אט (מישהו נכנס, יצא, שלח הודעה אלי או לכולם) וזה חוץ מהסוקטים להעברת הקבצים אבל הם לא רלוונטים לשאלה.

נניח את הנתונים הבאים:

- אצל אליס כתובת ה MAC של כרטיס הרשת הוא AAA, (מקצר פה במקום לכתוב 12 תווים כל פעם. גם עבור FF:FF:FF:FF:FF:FF אקצר ל- FFF)
- אצל אליס כתובת ה IP היא 10.0.0.1
- אצל אליס הפורט מקור של הסוקט "המרכזי" הוא 1111
- אצל השרת כתובת ה MAC של כרטיס הרשת הוא CCC
- אצל השרת כתובת ה IP היא 10.0.0.3
- אצל השרת פורט האזנה הוא 12345
- אצל בוב כתובת ה MAC של כרטיס הרשת הוא BBB
- אצל בוב כתובת ה IP היא 10.0.0.2
- אצל בוב פורט היעד של סוקט-העדכונים-מהצ'אט הוא 55000 והמקור הוא 1212

- אליס מחברת בכבל את המחשב שלה ל switch.
- עכשיו יש לה חיבור פיזי, והיא צריכה לתקשר עם שרת DHCP בשביל לקבל כתובת IP.

- discover: אליס מפרסם בקשה למציאת שרת DHCP

Discover: AAA → FFF, 0.0.0.0:68 → 255.255.255.255:67, UDP, "Discover msg"

- offer: שרת ה DHCP הקרוב מעביר הצעה עם פרטים עבור אליס:
נגיד שכתובת הראוטר היא 10.0.0.111 בכתובת MAC: DDD

Offer: DDD → FFF, 10.0.0.111:67 → 255.255.255.255:68, UDP, "Offer msg"

- request: אליס בוחרת את ההצעה שנשלחה ומבקשת להמשיך את השיחה עם שרת ה DHCP הנבחר ובעצם לקבל ממנו את ה IP והפרטים שהציע. זה נעשה ב broadcast כדי לעדכן את כל השרתים האחרים בסביבה שלא הם נבחרו להמשך השיחה:

Request: AAA → FFF, 0.0.0.0:68 → 255.255.255.255:67, UDP, "Request msg"

- Ack: השרת הנבחר מעדכן אותה (ולמעשה את כל הסביבה) שהוא נתן לה סופית את הפרטים:

Ack: DDD → FFF, 10.0.0.111:67 → 255.255.255.255:68, UDP, "Ack msg"

- וכעת לאליס יש כתובת IP (10.0.0.1), כתובת default gateway, וכתובת שרת DNS מקומי. אז היא יכולה להפעיל את תוכנת הלקוח.

- תוכנת הלקוח אצל אליס (זאת שקיבלה את הפרמטרים על השרת, שאליס ידעה מראש) יוצרת חיבור TCP עם השרת. בשביל זה היא צריכה לדעת את כתובת ה MAC אליה היא רוצה לפנות. כאן נגלה לפי ה subnet mask שהשרת והלקוח באותה רשת ולכן הודעת ה ARP תהיה עבור השרת ישירות:

ARP request: AAA → FFF, "How has 10.0.0.3? Tell 10.0.0.1"

נניח שהשרת ישירות עונה לה (בעקרון כל מי שיודע את התשובה יכול):

ARP response: CCC → AAA, "10.0.0.1 is at CCC"

- זהו, אליס יודעת כל מה שהיא צריכה, ונתחיל להעביר הודעה בצ'אט:

- ניצור חיבור TCP:
  TCP SYN: AAA → CCC, 10.0.0.1:1111 → 10.0.0.3:12345, TCP segment with SYN flag
  הערה: אני דילגתי על הקטע של ARP שמצא את CCC, ואני מדלג על 2 הפקטות הבאות של לחיצת היד.
- עכשיו אליס מחוברת ב TCP לשרת, אבל עוד לא לצ'אט. עבור כניסה לצ'אט היא תשלח את ההודעה הבאה, שבשכבת האפליקציה מדובר בהודעת בקשה להצטרף לצ'אט וזה מכיל את השם משתמש של אליס.
  GET-CONNECT: AAA → CCC, 10.0.0.1:1111 → 10.0.0.3:12345, TCP, "10005ALICE"
- כעת השרת עונה לאליס (נניח שהוא מחזיר הודעה 200 ולא 400) ומספר לה שהוא פתח עבורה פורט עבור הסוקט-עדכונים-מהצ'אט, נניח 55001 אז ההודעה תיראה כך
  CONNECT-OK: CCC → AAA, 10.0.0.3:12345 → 10.0.0.1:1111, TCP, "20005ALICE55001"
- עכשיו על מנת לקבל הודעות משאר המשתתפים בצ'אט, אליס צריכה ליצור סוקט TCP נוסף ולהתחבר איתו לשרת בפורט 55001, ובת'רד נפרד להקשיב להודעות שמגיעות ממנו.
- היא יוצרת חיבור TCP מול השרת בפורט 55001, (אני מדלג על לחיצת היד) ופשוט מקשיבה בלופ להודעות מהצ'אט.
- בינתיים, נשלחת הודעת עדכון לכלל המחוברים בצ'אט (רק בוב) שאליס נכנסה, וזה יראה כך:
  ALICE-JOINED: CCC → BBB, 10.0.0.3:55000 → 10.0.0.2:1212, TCP, "30005ALICE"
- עכשיו, עוד לפני שאליס יכולה טכנית לשלוח הודעות לבוב, נשלחת הודעה ממנה המבקשת את רשימת המשתמשים בשרת, וזה נראה ככה:
  GET-CONNECTED-USERS: AAA→CCC, 10.0.0.1:1111 → 10.0.0.3:12345, TCP, "102"
- עכשיו נשלחת הודעה חזרה מהשרת המכילה את רשימת המשתמשים המחוברים חוץ מאליס:
  SEND-USERS-LIST: CCC → AAA, 10.0.0.3:12345 → 10.0.0.1:1111, TCP, "2021103BOB"
- ורק עכשיו אליס רואה שהמשתתפים בצ'אט אליהם היא יכולה לשלוח הודעות זה רק בוב, אז היא בוחרת בממשק הגרפי את בוב (הממשק מונע נסיון לשלוח למשתמש שלא קיים, אבל למעשה גם לזה דאגנו בפרוטוקול) ואליס תכתוב בממשק למשל "hey bob" ותלחץ "שלח", מאחורי הקלעים תישלח ההודעה הבאה לשרת, שמכילה את ההודעה, ואת הנמען (בוב):
  SEND-MSG-TO-USER: AAA → CCC, 10.0.0.1:1111 → 10.0.0.3:12345, TCP, "10303BOB07hey bob"
- עכשיו השרת ירצה לשלוח תגובת אישור שליחה לאליס, אז קודם הוא ישלח לבוב את ההודעה, הוא ישלח לו את זה כמובן בסוקט המיועד להודעות מהצ'אט, ויציין שההודעה הגיע מאליס:
  MOVE-MSG-TO-USER: CCC → BBB, 10.0.0.3:55000 → 10.0.0.2:1212, TCP, "30305ALICE07hey bob"
- בנקודה זו, בוב שהקשיב להודעות מהסוקט המיועד להודעות בצ'אט קיבל הודעה, היא תלך לניתוח ותוצג בממשק הגרפי ההודעה שאליס שלחה לבוב.
- עכשיו כל שנותר לשרת הוא לשלוח תגובה לאליס שמאשרת לה שההודעה שלה הגיע לבוב:
  MSG-WAS-SENT: CCC → AAA, 10.0.0.3:12345 → 10.0.0.1:1111, TCP, "203"

.2 CRC הוא מן סוג של checksum על כל ה frame. הוא חלק משכבת הקו ומתווסף/יורד ברמת החומרה, כך שלא נראה אותו ב wireshark למשל.

.3 http 1.0 הוא עובד ללא הקבלת תהליכים, בעוד ש http 1.1 כן.
למשל, אם נרצה להוריד עמוד ב http 1.0 שמכיל הפניות ל 20 תמונות,
הוא יוריד את העמוד, ואז תמונה ראשונה, אחריה תמונה שניה, וכו' – מה שב http 1.1 היו נפתחים כמה סוקטים במקביל ומורידים כמה תמונות ביחד.
דבר נוסף, http 1.0 לא מכיל שדה host כי הוא חושב שבשרת יושב אתר אחד.
אבל במציאות על שרת אחד יכולים לשבת כמה אתרים, וצריך לציין את הדומיין בבקשה – זה קורה אצל http 1.1.
ההבדל בין Http 1.1 ל http 2.0 הוא שלגרסת 2.0 ההדרים הם בינאריים, מה שמאוד מקל על התעבורה – כי למשל בפרוטוקול טקסטואלי, עבור הערך 1 (למשל עבוד איזה דגל) נצטרך תו, שזה 8 ביטים, ובבינארי נסתפק בביט אחד. עבור הערך 30, בטקסטואלי נצטרך 16 ביטים (שני תווים), ובבינארי נצטרך 5 ביטים, וכו'.
ב http 2.0, יש גם multiplexing עבור האובייקטים שיורדים, מה שמדמה את עבודה שכבת התעבורה – רק עבור אובייקטים של html ודומיהם, וכך למרות המהירות של http 1.1 על פני http 1.0 (קצת כמו היחס בין stop & wait ל gbn, למרות שזאת לא אותה שכבה פה, אני מדבר רק

על צורת העבודה...) הבעיה של חסימת חבילות אחרות (דומה ל GBN) ב http 1.1 נפתרת על ידי מיתוג האובייקטים שיורדים (בדומה ל selective repeat) שזה כבר נעשה ב http 2.0

חוץ הפרוטוקולים הנ"ל, גוגל פיתחה את QUIC, שמספק העברת אתרים + אבטחה, מה שאין בגרסאות ה http ומה שעוטף אותם בפועל זה פרוטוקול אחר בשם TLS.
גוגל טוענת שזה יותר מהר ככה.

.4 פורטים נועדו על מנת לאפשר לשתי ישויות קצה לנהל שיחה על תווך פיזי אחד בו עוברות כמה הודעות מסוגים שונים (סוג == אפליקציה). לולא מספרי port היה ניתן לנהל בתקשורת עם מחשב אחד רק סוג הודעות אחד. לולא פורטים גם הייתה בעיה ל NAT להכניס חבילה לרשת פנימית. כי הוא עוזר "על הדרך" להבדיל למי מבין המחשבים ברשת שייכת הודעה כלשהי מבחוץ.

.5 subnet היא רשת המהווה חלק מרשת. היכולת לחלק רשת לכמה תתי רשתות עם הגדרת מזהה רשת באמצעות מסכת רשת, או CIDR (אותו דבר תכלס..) נותנת לי אופציה להגדיר הגדרות שונות עבור כל תת רשת, מבלי שאצטרך להקצות כתובות חדשות לבניית רשתות חדשות. מאוד חסכוני.

.6 כתובת MAC היא קודם כל ה id של כל כרטיס רשת. השימוש בה נעשה רק בתוך תת רשת ולא מחוץ לה. כרטיס רשת לוקח חבילות המיועדות לו או ל broadcast (אלא אם כן הוגדר לקחת הכל..) והוא עושה זאת לפי כתובת MAC. הוא טכנית לא מסוגל לקרוא כתובות ip, שבין ב ipv4 ובין ב ipv6 זה לא אותו מספר בתים הדרוש לכתובת MAC.. סיבה נוספת לנחיצות שלו – רכיב switch לא מכיר כתובות ip, הוא חייב mac, עוד סיבה – בהצטרפות לרשת על ידי DHCP המזהה המרכזי של המחשב הוא כתובת ה MAC (יש גם seq לכל שיחה כזאת אבל הוא מזהה השיחה ולא המחשב..)

.7 **ראוטר –** תפקידו לחבר בין רשתות על ידי השתתפות בכמה מהם בו זמנית על ידי לפחות 2 ממשקים.

Switch – תפקידו לחבר בין ישויות שנמצאות באותה תת-רשת

NAT – פותר את בעיית כמות הכתובות של ipv4, הוא בדרך כלל יהיה פונקציה בראוטר ביתי ולא רכיב פיזי. תפקידו להחליף לכל חבילה יוצאת את כתובת ה ip ממנו בא, לכזאת השייכת לספק, כלומר אחת הכתובות האמיתיות, כזאת שהלקוח שילם לספק עבורה. כשחבילה רוצה להיכנס לתת רשת נעשה תהליך הפוך.

.8 רכיב NAT, ההסבר בסוף שאלה 7, ופרוטוקול IPv6 שמספק טווח רחב יותר של כתובות (2 בחזרת 48)

.9 - הנתב 3c מחובר ישירות ל AS4 והוא חלק מה default gateways של AS3, ולכן הוא לומד על x על ידי eBGP בלבד.

- הנתב 3a שייך ל AS3, אבל אינו ה default המתאים במקרה הזה. לכן הוא לומד על x על ידי iBGP ועל ידי OSPF (OSPF אומר לו להביא חבילות ל x דרך 3b, כי iBGP מציין שזה אומר לעבור דרך 3c)

- הנתב 1c לומד על x על ידי eBGP (ולא גם iBGP ו-RIP, כי 1c הוא הדיפולט המתאים כאן של 1AS)

- הנתב 2c לומד על x דרך OSPF ו iBGP, כי iBGP אומר להעביר ל x דרך 2a, ו OSPF אומר ל 2c היכן לצאת עבור 2a.