

**שכבת שבע השכבות של OSI**, יישום, תצוגה, שיחה, תובלה, רשת, ערוץ, פיזית.

שכבת ה**יישום** אחראית על אינטראקציה עם המשתמש כפרוטוקול http וכו'.

שכבת ה**תצוגה** היא תרגום של aski פענוח למשתמש את מה שקיבלנו ממספרים לאותיות, פענוח וקידוד נתונים כדי שהנתונים יא' גילשו בטעות למשתמש אחר.

שכבת ה**שיחה**, הקמת שיחה, משך השיחה, ניהול השיחה, וסיום השיחה.

שכבת ה**תובלה**, יצרת חיבור לוגי בין שני מחשבים בכל מקום בעולם כאילו מחשבו אחד מחובר למחשב השני בכבל ואין שום מחשב מלבדם בעולם, גם אחראית על הזרמת המידע.

שכבת ה**רשת**, אחראית על חיבור בין מחשבים מרוחקים, ניתוב בדרך הנוכנה להגיע מנקודה לנקודה בדרך הקצרה ביותר.

שכבת ה**ערוץ** אחראית על החיבור בין המחשבים ברשת המקומית.

ה**שכבה הפיזית** זה הדברים הפיזיים של המחשב, כרטיסי שרת,כבל רשת, וכו'.

**שולח** המידע יורד בשכבות וזה נקרא כימוס, מקבל המידע עולה בשכבות וזה נקרא קילוף.
לכל שכבה מתווסף נתוני בקרה, והשכבה הבאה שמקבלת אותה מקבלת הכל כתבילה אחת ומוסיפה נתוני בקרה של השכבה עצמה.
PDU שם הנתונים שנשלחו בכל שכבה, יישום תצוגה ושיחה=הודעה, תובלה=מקטעים, רשת=מנות, ערוץ=פריימיים או מסגרת, פיזית=סיביות.

**ארבע** השכבות הראשונות אני עדיין מבחשש שלי כשאני מגיע לשכבת הרשת אני יוצא מהמחשב שלי.
**ראטור** תפקידו לחבר בין רשתות, שייר לשכבת הרשת (אינטרנט זה קבוצה של רשתות מקומיות שמתברכות בינם לבין עצמם וזה באמצעות ראטור למשל).
ה**סוויץ** מחבר בין מחשבים בתוך הרשת עצמה והוא שייר לשכבת הערוץ (ככלי **HAB** והוא שייר לשכבה הפיזית),
ה**מודם** ממיר אותות דיגיטליים לאלגונים וההפך.
**פרוטוקול** פירושו שפה שאיתה אני מתנהל מול הצד השני כדי ששני הצדדים יבינו אחד את השני (סל של כללים והוראות).

**פרוטוקול DHCP** תפקידו להקצות באופן אוטומטי כתובות IP (כדי לתקשר עם מחשב חייב לדעת מי הוא והיכן הוא נמצא כמו כאשר רוצים לתקשר עם בן אדם ולהגיע אליו לבית חייבים לדעת את הכתובת שלו) בנוסף הפרוטוקול גם מגדיר עוד כמה דברים כגון הקצאת כתובת הציאה שלנו מהרשת שמקראת דיפונט gateway וכן את שרת ה- dns.

ה**פרוטוקול עובד בארבעה שלבים** שנקראים DORA,
הודעת ברודקאסט ack request offer discover בשלב ה**ראשון** הלקוח שולח הודעת דיסקבר מסוג ברודקאסט לכל המחשבים ברשת ושואל מי הוא שרת dhcp (הודעת יוניקסט היא הודעה ממחשב למחשב ספציפי, הודעת ברודקאסט היא הודעה ממחשב אחד למספר מחשבים ספציפיים ברשת) בשלב ה**שני** שרת dhcp שולח הודעת offer שולח מציג לי כתובת ip שיש לי, בשלב השלישי הלקוח שולח הודעת request שהוא מקבל את הצעת כתובת ה IP , בשלב ה**רביעי** השרת שולח הודעת ack המאשרת את הקצאת כתובת ip (חייב את האישור הסופי הזה מכיוון שיכול להיות שבין השלב השני לשלב הרביעי כבר מחשב אחר שהוא אולי יותר רזיז תפס את כתובת ה ip הזו)
**חכירת כתובת ip** אם לדוגמא הכתובת הוכרזה לשעה, בחצי מהזמן הלקוח עובד רגיל לאחר חצי מהזמן הלקוח שולח בקשת חיידוש של כתובת ה ip אם עד 85% מהזמן הלקוח לא קיבל תשובה משרת dhcp כנראה ששרשת נפל והלקוח צריך לעשות מחדש את כל תהליך הDORA.
**כתובת APIPA 168.254.0.0** כתובת זמנית במקרה ששרת dhcp נפל או אינו מחובר למחשב או לא תקלה אחרת הכתובת הזו נועדה לתקשרות זמנית בין המחשבים בתוך הרשת.

ה**ראטור** הביתי שלנו הוא גם ראטור וגם סוויץ וגם מודם.
מראה האם זה הודעת request או reply וכן מהביט 32 זה מראה איזה סוג הודעה נשלחה וזה בגודל 32 ביטים וכן הלאה.

**שרת dns** **ממפה** כתובות רשת לכתובות ip וי' תהליך של **8 שלבים**, אבל לא עושה את זה כל פעם, אם כבר גלשתי פעם אחת זה נשמר ב dns cash, שלב 1 אני יוצא ולספקית האינטרנט ושולח שאילתת dns תביא לי את הכתובת של גוגל, שלב 2 השרת שולח הודעה למחשב root server (יש בדיק 13 כאלה בעולם) תפקידו לבדוק איזה tld אני מבקש (.com וכו') (כל המבנה של כתובת האינטרנט נקרא tld), שלב 3 אח מזחיר לי כתובת של tld שאחראי על כל ה dns או coil וכו, שלב 4 אני מגיע ל tld הזה, שלב 5 ה tld שולח ל dns את הכתובת של גוגל לדוגמא, שלב 6 ה dns נוגש לשרת dns המחשב הראשי של גוגל למשל ומבקש ממנו את הדף אינטרנט שהוא לדוגמא www, שלב 7 הוא שולח את הדף המבוקש ל dns, שלב 8 ה dns מחזיר למחשב שלי את הדף ורק כעת אני יכול לגלוש לדף המבוקש, תהליך זה לוקח לפעמים כמה שניות.

**פרוטוקול HTTP** פרוטוקול של דפי האינטרנט, שולחים אתר מקבליים אתר, פרוטוקול עובד באמצעות requests, get בקשת דף אינטרנט, post יצירת משאב חדש, put העורך משאב קיים, delete מחיקת משאב קיים. ובאמצעות response בקוצת 200 הבקשה מולאה בהצלחה, בקוצת 300 ריל דייקט, בקוצת 400 שיגאות לוקח כדוגמת המבוקש לא קיים, בקוצת 500 שיגאת שרת.
**הפורטים מחולקים לשלוש קבוצות**, קבוצה 1 מ 0 עד 1023 נקראת well known שזה קבוצת הפורטים שכולנו משתמשים בהם, קבוצה 2 registers זה רשומים, אם ארגון רוצה לרשום איזה פורט על שמו וזה מתחיל מ 1024 עד 49,150, קבוצה 3 ephemeral זה קבוצה של פורטים זמניים שמערכת ההפעלה שלנו משתמשים בהם, קבוצה 4 שיש להם שם שהוא אינטרנטי וזה נשמר ב dns cash, שלב 1 אני יוצא ולספקית האינטרנט ושולח שאילתת dns שלב 2 השרת שולח הודעה למחשב root server (יש בדיק 13 כאלה בעולם) תפקידו לבדוק איזה tld אני מבקש (.com וכו') (כל המבנה של כתובת האינטרנט נקרא tld), שלב 3 אח מחזיר לי כתובת של tld שאחראי על כל ה dns או coil וכו, שלב 4 אני מגיע ל tld הזה, שלב 5 ה tld שולח ל dns את הכתובת של גוגל לדוגמא, שלב 6 ה dns נוגש לשרת dns המחשב הראשי של גוגל למשל ומבקש ממנו את הדף אינטרנט שהוא לדוגמא www, שלב 7 הוא שולח את הדף המבוקש ל dns, שלב 8 ה dns מחזיר למחשב שלי את הדף ורק כעת אני יכול לגלוש לדף המבוקש, תהליך זה לוקח לפעמים כמה שניות.

**בשכבת התובלה** יש שני פרוטוקולים עיקריים פרוטוקול udp ופרוטוקול tcp, פרוטוקול udp לא מקושר ולא אמין ופרוטוקול tcp מקושר ואמין, בפרוטוקול udp אין שום הבטחה או התחייבות שהמקטעים שאני שולח יגיעו לצד השני או שיגיעו בלי שיגאות, בפרוטוקול tcp נשתמש כשנעדיף מהירות על פני אמינות (יש לציון שברוב המקרים אנז משתמשים ב tcp).
**הכוונת של פרוטוקול udp** בנויה מארבעה שדות כל שדה 16 ביט שביחד הכל 64 ביט והם שמונה בתים, השדה הראשון הוא source port פירוש מספר מפתח של יישום הרשת מצד השולח, השדה השני הוא destination port מפתח של יישום הרשת מצד המקבל, השדה השלישי הוא length שהוא גודל המברק, השדה הרביעי הוא checksum שזה קבוצת שיגאות. כעת גם מובן למה מספר הפורט יכול להגיע ל-65,536 כי אם נתרגם את המספר הזה לבינארי זה 16 ביטים. לרוב אנחנו נשתמש בפרוטוקול tcp כפייון שברוב הזמן אנחנו לא נשים לב לשיגאות הקטנות אם נוצרו במהלך השליחה של המקטעים בעין אנושית, וגם במקרה שהייתה שיגאה גדולה תמיד אפשר לשלוח שוב.

**פרוטוקול** הוא מקושר ואמין והגודל שלו יכול להיות בן 20 בתים ל-60 בתים. מקושר פירושו שתמיד הוא ישאל אם אפשר ליצור קשר לפני שהוא שולח נתונים (דהיינו תהליך מקדים לפני יצירת הקשר),ואילו udp אינו מקושר והוא ישר שולח את מבקש שמונה.

**פרוטוקול udp** נחשב למהיר יותר מ 3 סיבות. סיבה ראשונה, אין מנגנוני אמינות. סיבה שניה, אין מנגנון קשירת קשר דהיינו שאינו יוצר תהליך מקדים לפני שידור הנתונים עצמם. סיבה שלישית, היא שהכוונת של ה udp מכילה 8 בתים לעומת tcp שהוא בין 20 ל- 60 בתים, ואם לדוגמא כל מקטע שאני שולח הוא מוגבל לגודל של 65 קילו ביט ואני רוצה לשלוח תמונה ששוקלת הרבה יותר אז ברור שב udp מכיוון שנתוני הבקשה שלו קטנים יותר אז התמונה תישלח בפתות מקטעים מאשר בפרוטוקול tcp שנתוני הבקרה שם הרבה יותר גדולים והתמונה תישלח בהרבה יותר מקטעים ואם במקרה יהיה תקלה במקטע מסוים זה ייצור בעיה בשליחת התמונה.

ה **checksum** הוא מנגנון בדיקה של תקינות הנתונים דהיינו שהוא בודק אם הנתונים הגיעו לצד השני באופן תקין או שהיו שיגאות בדרך.(שיגאות פירושו ש 0 הופך ל 1 או 1 הופך ל 0 אולי דרך גלים אלקטרומגנטיים או כל תקלה אחרת באוויר), הוא אינו חובה אלא אופציונלי ושכבת היישום תחליט אם לבצע אותו או לא.
**כיצד עובד מנגנון ה checksum ? עיין מעבר לדף**

**פרוטוקול tcp** הוא פרוטוקול אמין ומקושר. מקושר פירושו שקיים תהליך קשירת קשר לפני שידור הנתונים עצמם וכך זה עובד, **ביצירת קשר**, הלקוח שולח בקשת syn שהיא בקשת התחברות לשרת לאחר מכן השרת שולח ack שזה אומר אישור שקיבלתי את הבקשה שלך + בקשת syn שגם אני רוצה להתחבר אליך, ובפרוטוקול tcp כל מקטע חייב להיות מאושר לכן בכל פעם תשלח הודעת ack ולאחר שהשרת שולח גם בקשת syn זה אומר שהוא רוצה גם להתחבר ללקוח, נשלחת הודעת ack שהתחברות נעשתה בהצלחה, תהליך זה נקרא לחיאת יד משולשת, ורק כעת בשלב זה.נוצר הקשר בין השרת ללקוח.
**בביתוק קשר**, יכול להיות שיהיה פתאומי ניתוק מידי באמצעות הדגל res, לצורך הענין עומס על שרתים או מסיבות אבטחה כל שהן, ויכול להיות שיהיה חינוי, פירוש שהלקוח רוצה לנתק את הקשר מכל סיבה שהיא, ישלח הלקוח הודעת fin והשרת יחזיר לו ack לאשר את קבלת המקטע ועדיין לא מסיים את הקשר לאחר מכן ישלח לו הודעת fin בנפרד מה ack ובגלל שהלקוח קיבל מקטע אז הוא יאשר לו בהודעת ack את קבלת המקטע, ורק כאן הקשר מתסיים, לפעמים מתייחסים לזה לחיאת יד מובעת.
מצב הבקשה רוצה לנתק את הקשר אבל השרת עדיין לא סיים לשדר את מה שהוא רצה, לכן הוא יודיע שהוא קיבל את ההודעה אבל ימשיך לשדר וכשסיים ישלח גם הוא בקשת ניתוק ואז הלקוח יאשר באמצעות ack את קבלת הבקשה. כל זה לא קיים ב tcp אלא הוא ישר שולח הכל.

**אמינות של tcp**
**מספרי רצף(seq number)** מספר מזהה למקטעים, מספר הרצף הראשון הוא נרדומלי מ 0 עד 2 בחזקת 32, המספר הזה נקרא init seq number, המספר הבא של המקטע הבא יהיה ה seq number הקודם + ה length, לדוגמא שלחתי מקטע ראשון והמספר הנרדומלי של היה 10 וגודל המקטע היה 500 בתים, המקטע הוצא אחריו יקבל את המספר 510,ואם שלחתי מקטע הבא בגודל של 200 בתים, המקטע הבא אחריו יקבל את המספר 710, וכן הלאה חיבור של המספר הקודם + גודל המקטע.
tcp הוא פרוטוקול ישן, בזמן אף אחד לא תיאר שמישהו יוריד קובץ בגודל כזה, כיום מורידים הרבה יותר מזה, מה שקורה היום שכאשר הוא מסיים סיבוב הוא חוזר ל 0 וממשיך להוסיף.

**מספרי אישור ack number**, מספר מזהה למקטעים, גם כאן המספר הראשון הוא נרדומלי, לדוגמא 20 וגודל המקטע שנשלח הוא 200,לאחר מכן הודעת ה ack שתישלח תהיה 221, ואם נשלח עוד מקטע בגודל 300, הודעת ה ack שתישלח תהיה 522, פירוש שהצד השני מודיע אני מצפה לקבל כעת את מספר הבית הבא, עצם העובדה שהוא מצפה לקבל מספר מסוים שמעם שעד לאותו הבית הכל התקבל בצורה תקינה.

ה**דגלים**, urg כאשר חלק במקטע מסוים דורש טיפול התחלתי לפני כולם, ack הודעת אישור קבלת מיקטע, res סיום קשר פתאומי, syn קשירת קשר, fin סיום קשר חינוי.

ה **windows size נקרא גם rwnd** מתייחס לגודל החלון של המקבל, לדוגמא אם גודל החלון הוא 4000 והמקטע שנשלח הוא בגודל 300, המקבל שולח הודעת 3700 + win 301 ack שזה המקום שנשאר לי לקבל מקטעים נוספים, המחשב המקבל מעדכן את גודל החלון כך שהשולח לא יוכל לשדר מקטעים גדולים יותר ממה שפנוי ואם בכלל לא פנוי השולח לא ישרד עד קבלת הודעת עדכון שהחלון התפנה.

**חלון ה checksum** זהה לחלוטין ל checksum ב udp, ההבדל ביניהם שב udp הוא לא מנדטורי לא חובה ואילו ב tsp הוא חובה.

**האינטרס של tcp** הוא לשלוח כמה שיותר מקטעים ברצף עד לקבל ack, לדוגמא אם יש לנו 20 מקטעים וכל מקטע 1000 בתים, ו 4 כבר התקבלו ואושרו, 4 יהיו באוויר עד שאיושרו, וכל השאר עדיין לא ישלחו, ברגע שאחד המקטעים מתוך ה 4 שבאוויר יקבל אישור מיד יתווסף ל 3 הנותרים מקטע נוסף מהמקטעים שעדיין לא נשלחו שיהיה באוויר וימתין לקבלת אישור, זאת אומרת שבכל זמן נתון יהיה באוויר מספר מקטעים גודל החלון שיכול להכיל המקבל.

**מה קורה במצב של שידורים חוזרים**, דהיינו אם יש שיגאה באחד המקטעים, יש שני דרכים. א, go back n (gbn) כל מה שבאוויר נשלח מחדש. ב, selective rturen (sr) המקטע השגוי נשלח שוב וכל המקטעים התקינים ממתינים גם במחשב המקבל.
**יתרון ב gbn** לא דרוש מקום מיוחד בזיכרון לשמור מקטעים כי בכל מקרה אקבל אותם שוב, שזה **חיסרון ב sr** ששם דרוש מקום בזיכרון לשמירת המקטעים שהתקבלו עד לקבלת המקטע השגוי מחדש.

**כל זה**, הוא הקדמה לבקרת עומסים שזה חלק מהאמינות של tcp.

**בקרת עומסים**. שתי בעיות יכולות להיות כאשר מדברים על עומסים, מקרה של duplicate ack ומקרה של time out. מקרה של duplicate ack לדוגמא, אם נשלח מקטע בגודל 200 והמקבל שולח הודעת 201 ack כי הוא מצפה לקבל כעת את הבית ה 201 ומשום מה המקטע הבא שנשלח הוא 50 אזי המקבל ישלח שוב שהוא מצפה לקבל מהבית ה 201, וכן אם שוב פעם המקטע הבא שנשלח מהבית מספר 300, המקבל ישלח שוב שהוא מצפה ל 201 ack, כאשר המערכת מזה 3 פעמים ack זהים, היא מבניה שיש בעיה וצריך להפחית עומס ולהאט את השליחה. ובמקרה של time out יש חלון זמן שאם עד אז לא נשלחה הודעת ack, מבחינת המערכת המקטע בכלל לא הגיע לצד השני וצריך לשלוח אותו שוב.

מספר המקטעים שניתן לשלוח עד לקבלת ack מוגבל ב 2 דברים: 1, גודל החלון של המקבל. 2, ssh threshold כך זה עובד ראה איר

ליסיכום: TCP ה-1 ומקושר, 2-אמין, תומך בבקרת עומסים באמצעות חלון עומסים, תומך בבקרת שיגאות באמצעות שדה ה checksum, תומך באובדן מקטעים באמצעות מספרי רצף ומספרי אישור, תומך בשידורים חוזרים באמצעות selective

בפרוטוקול tcp נשתמש בדרך כלל כאשר נעדיף אמינות על פני מהירות.



