בס״ד

אוניברסיטת אריאל

המחלקה למדעי המחשב

תקשורת ומחשוב

מס׳ הקורס : 7028310

מרצה : ד״ר דביר עמית זאב

העורך : משה חנוקוגלו

תאריך : תשע״ח - סמסטר א׳

Internet Addressing

לכל מחשב ישנה כתובת שמאפיינת אותו בתוך קבוצת המחשבים שהוא מחובר אליה, לקבוצה זו נקרא הרשת של המחשב (הכתובת שיש למכשיר ברשת שאליה הוא מחובר אינה בהכרח אותה כתובת שאיתה הוא מדבר עם מכשירים שמחוץ לרשת זו עקב ה Nat – נלמד על כך בהמשך). את הכתובת הוא יקבל בד״כ מהנתב (Router) בפרוטוקול DHCP (פרוטוקול – צורה מוסכמת לשימוש בדבר מסויים). הכתובת בנויה בתבנית xxx.xxx.xxx.xxx כאשר כל קבוצה xxx יכולה לקבל מספר בין 0 (כולל) ל 255 (כולל).

בעצם קבוצת מספרים זו היא רצף של 32 ביטים כך שכל אחד מארבעת הקבוצות מייצגת 8 סיביות.

כל כתובת מורכבת משני חלקים, מספר הרשת ומספר מחשב בתוך הרשת.

הערה: ע״מ לראות את כתובת ה IP של מחשב מסוים צריך להיכנס ל CMD של Windows (Terminal של Linux) ולכתוב ipconfig (in Linux write ifconfig). שם נוכל לקבל מידע כגון כתובת IP, Subnet Mask (יוסבר בהמשך) וכו׳.

ישנם שתי צורות לחלק את הכתובת לשני החלקים, מספר רשת(network number) ומספר מחשב ברשת (Host number).

1. Class
2. Classless Inter-Domain Routing (CIDR)

צורה 1, Class

מחלקת את כל הכתובות (מ 0.0.0.0 עד 255. 255. 255. 255) לחמש מחלקות.

Class	Leading Bits	Size of *Network Number* Bit field	Size of *Rest* Bit field	Number of Networks	Addresses per Network	Start address	End address
Class A	0	8	24	128 (2^7)	16,777,216 (2^{24})	0.0.0.0	127.255.255.255
Class B	10	16	16	16,384 (2^{14})	65,536 (2^{16})	128.0.0.0	191.255.255.255
Class C	110	24	8	2,097,152 (2^{21})	256 (2^8)	192.0.0.0	223.255.255.255
Class D (multicast)	1110	not defined	not defined	not defined	not defined	224.0.0.0	239.255.255.255
Class E (reserved)	1111	not defined	not defined	not defined	not defined	240.0.0.0	255.255.255.255

[1]

מחלקות A,B,C ניתנות לשימוש לכולם. D,E שמורות לשימושים מיוחדים (ישנם כמה כתובות מיוחדות כגון 127.0.0.1 שהם שמורות לשימושים אחרים).

ההבדל בין מחלקות A,B,C הוא בצורת החלוקה של הכתובת לשני החלקים.

מחלקה A, נותנת 8 סיביות למספר הרשת ואת שאר 24 הסיביות נותנים למחשבים באותה רשת.

מחלקה B, נותנת 16 סיביות למספר הרשת ואת שאר 16 הסיביות נותנים למחשבים באותה רשת.

מחלקה C, נותנת 24 סיביות למספר הרשת ואת שאר 8 הסיביות נותנים למחשבים באותה רשת.

החיסרון של שיטה זו הוא שאם רשת מסוימת צריכה, לדוגמא, 2000 מחשבים שזה יותר מ 2^8 ולכן אי אפשר להשתמש ב Class C אלא צריך להשתמש ב Class B אך שם יש לנו 2^{16} שזה הרבה יותר ממה שצריך ואז ישנם המון כתובות לא בשימוש.

[1] אנה פרבר

לכן המציאו צורה חדשה של חלוקת הכתובות לשני החלקים.

<u>וצורת זו היא צורה 2, CIDR</u>

בדרך זו יכולים להקצות כל מספר של סיביות (בין 1 ל 31 כולל) למספר הרשת ושאר הסיביות למספרי המחשבים באותה רשת. (לכן אם מישהו צריך 2000 מחשבים אז ניתן לו 11 סיביות).

בשיטה זו נצטרך לדעת כמה סיביות מייצגות את מספר הרשת, ישנם שתי דרכים לקבל מספר זה.

1. מספר טבעי בין 1 ל 31 כולל. ואז ירשמו xxx.xxx.xxx.xxx/k כאשר k זהו המספר שמיצג את מספר הסיביות לרשת.
2. תבנית של מספרים xxx.xxx.xxx.xxx שנקראת Subnet Mask (שאותה יקבל המחשב מאותו שרת שנתן לא את כתובת ה IP), כאשר כל קבוצה xxx יכולה לקבל מספר בין 0 (כולל) ל 255 (כולל). כל קבוצה xxx נהפוך לסיביות ואז כל סיבית ששווה 1 ב Subnet Mask אומרת שסיבית באותו מקום בכתובת שייכת למספר הרשת והשאר הסיביות שייכות למספר המחשב. (לדוגמא: אם נתון Subnet Mask בצורת 255.255.128.0 אזי בבינרית זה 11111111.11111111.10000000.00000000. 11111111 מה שאומר שה 1+8+8=17 סיביות הראשונות שמורות למספר הרשת והשאר למספר המחשב ברשת זו).

הערה: מבנה הכתובת כפי שהצגנו xxx.xxx.xxx.xxx הוא לפי IPv4, אך כיום קיימת בנוסף גרסה IPv6 שיש לה מבנה שונה קצת.

עכשיו אחרי שידוע לנו כיצד בנויה הכתובת, ע״מ לבדוק אם שתי כתובות שייכות לאותה רשת אז נבצע פעולת And (שער לוגי) בין כל אחת מהכתובות לבין ה Subnet Mask שלה, אם שתי התוצאות אותו הדבר אז משמע שהם באותה רשת.

עכשיו כאשר מחשב רוצה ״לדבר״ עם מחשב אחר דבר ראשון הוא בודק האם המחשב השני נמצא על אותה רשת שלו, ואז הוא ישלח לו בצורה ישירה. אחרת הוא שולח למכשיר שנקרא <u>נתב (router)</u> ואז הנתב מסתכל על כתובת היעד ומכוון את חבילת המידע הלאה לכיוון שהוא מבין מהכתובת שיכול להיות היעד. וכך מעבירים את החבילה עד שהיא בסוף מגיע ליעדה.

הסיבה שהנתב לא יודע ממש מה המחשב שאליו מיועדת החבילה היא שכדי לדעת זאת הוא צריך לשמור אצלו את כל הכתובות של כל המחשבים והרי זו רשמיה מאד גדולה וככל שתגדל כמות המחשבים תגדל הרשימה ולבסוף לא יהיה אפשרי לנתב לדעת על כל הכתובות של המחשבים. לכן הנתב שומר לעצמו רק ה Subnet Mask וכאשר הוא מקבל חבילה הוא בודק את מספר הרשת שלה וכך הוא יודע להעביר אותה להמשך ״טיפול״ של נתב אחר עד שהיא מגיע לנתב של הרשת אליה היא מיועדת ושם הוא מכניס אותה למחשב המתאים.

הנתב כפי שאפשר להבין בד״כ מחבר בין הרשת הפנימית לבין הרשת החיצונית, לכל חיבור כזה של הנתב קוראים ממשק (Interface)

The Network Edge

כל רשת בנויה ממכשירים שעומדים בקצוות הרשת והם מתקשרים ביניהם דרך הרשת.

ישנם כמה צורות של מודלים לרשת, לדוגמא Client-server או peer to peer.

<u>מודל Client-server</u>

במודל זה כל המכירים באותה רשת מחוברים לשרת (Server). והשרות שהמכשיר מקבל הוא ע״י השרת.

כלומר השרת מעביר נתונים בין מכשירים וכן תעבורת הרשת עוברת דרכו דבר שמגדיל את האבטחה על המכשירים ברשת. החסרון הוא שהוא יותר יקר לתפעול.

השרת תמיד פתוח (להבדיל מכשיר פרטי שלא תמיד), יש לשרת כתובת IP קבועה וכן הוא משמש כמרכז נתונים למכשירים ברשת.

והמכשיר הפרטי יכול להתחבר ולהתנתק מהשרת לסרוגין, הכתובת IP לא קבוע ולא יכול להתקשר ישירות עם מכשיר אחר אלא רק דרך השרת.

<u>מודל peer to peer</u>

במודל זה המכשירים יכולים לדבר בינם לבין עצמם ולא בהכרח דרך השרת והם יכולים להתחבר ולהתנתק אחד מהשני מתי שירצו בלי להיות תלויים בשרת, דבר זה גורם להורדת האבטחה כיוון שאין שום גורם (כגון השרת במודל Client-server). שיכול למנוע מעבר דבר ״רע״ ממכשיר אחד לאחר אך מודל זה זול יותר לשימוש. (אך הרבה מהמקרים את החיבור הראשוני יעשו הצדדים דרך server)

The Access Network

את חיבור האינטרנט למכשירים השונים אפשר לבצע בשני דרכים, חוטי ואל חוטי.

לכל אחד מהם יש יתרון וחיסרון.

חוטי, מעביר יותר מהר ויותר אמין את הנתונים אך יכול יותר להינזק. (כיום יש שימוש בשיטת Fiber to the Home – FTTH, הכוונה לסיב אופטי שמעביר עוד יותר מהר את הנתונים)

אל חוטי מעביר נתונים למרחק יותר גדול אך יותר לאט ויכול להיות יותר שגיאות בהעברה.

כל אחת מהשיטות משמשת גם למרחקים קצרים וגם לארוכים.

ועכשיו אנו מגיעים לנקודה שהיא כיצד להשתמש באמצעי החיבור כדי לקבל את המצב הקרוב ביותר להיות אידאלי.

ניקח לדוגמא את העולם, אם אנחנו רוצים שמחשבים ידברו בניהם אז אנחנו צריכים לחבר אותם, אם נמתח כבל בין כל המכשירים בעולם אזי נצטרך המון חוטים ומשאבים כדי לחבר את כולם. אז כדי לחסוך אנו נבנה מערכת שתקשר את כל במחשבים ואותה נבנה בשכבות כך שהשכבה הקרובה ביותר למכשיר תהיה באזור המכשיר משם הוא יוביל לתחנה של העיר ואז של המדינה בדרך יעבור דרך ספק האינטרנט של המדינה/אזור (ISP) וספקי האינטרנט מעבירים לשכבה מעליהם שהיא (IXP) ושיכבה זו מחוברת לספקי אינטרנט העולמיים. ובעצם כך המכשירים לא יהיה צריכים להיות מחוברים ממש בניהם אלא הם מחוברים לתחנות והתחנות מחוברות בניהם.

OSI Protocol Stack

ע״מ שנוכל לשלוח חבילה ממחשב אחד לאחר, אנחנו חייבים לצרף לחבילה שלנו עוד חלקים ״שיעטפו״ אותה כדי שהחבילה הזו תדע איך לעבור ברשת מה היעד שלה ומה היא אמורה לעשות כאשר תגיע ליעד.

וכמובן צריך שתהיה הסכמה בין המנהלי השיחה כיצד לארגן את החבילה הזו כדי שבזמן שאחד יקבל את החבילה הוא ידע כיצד לקרא אותה. לכן הארגון ISO הקים מודל OSI שכולם יקבלו אותו כדי להבין אחד את השני. במשך השנים שינו קצת את המודל אבל הרעיון המרכזי נשאר.

ישנם שתי שיטות ל״עטיפת״ חבילה, שיטת 7 השכבות (השיטה הראשונה) ושיטת 5 השכבות (הרעיון המשודרג).

<u>שיטת 7 השכבות</u>

מודל שבע השכבות, כשמו כן הוא, הוא כולל שבע שכבות שעוטפות את החבילה בשלב מסוים הפכו את המודל הזה לחמש שכבות.

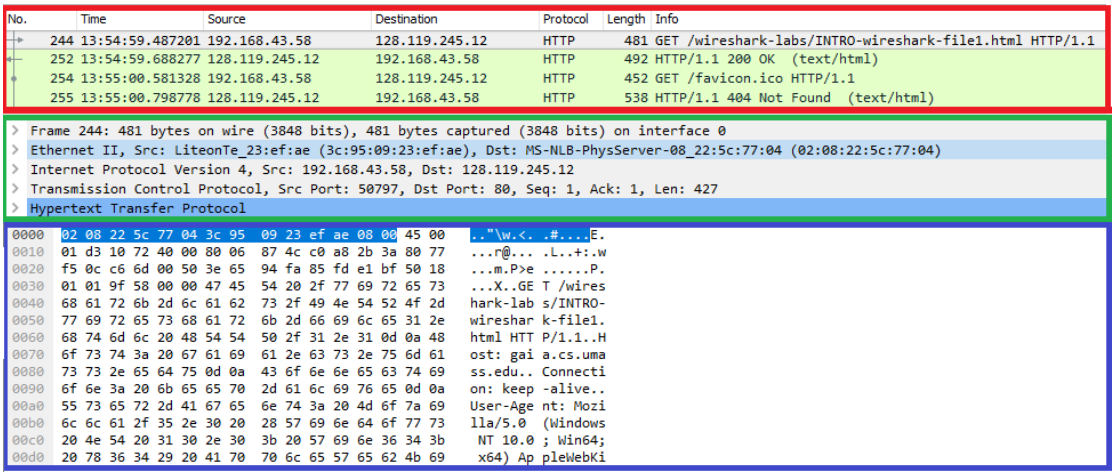
<u>מודל 5 השכבות</u>

חמשת השכבות הן מהפנימית החוצה:

1. Application (אפליקציה) – השכבה שהכי קרובה למשתמש, היא זו שתדע האם מדובר באימייל, אתר מבוקש או כל שירות אחר שביקשתי. זוהי השכבה שבה מזוהים שותפי התקשורת (האם יש מי לדבר?), קיבולת הרשת מוערכת (האם הרשת תיתן לי לדבר איתם עכשיו?), ליצור את הדבר שהשולח רוצה לשלוח או לפתוח את הדבר שהתקבל בצורה הנכונה (אם זה אימייל ,סתם דף אינטרנט, שירות העברת קבצים או כל שירות אחר). (שכבה זו אינה היישום עצמו, היא קבוצה של שירותים שהיישום צריך להיות מסוגל לעשות שימוש בהם, אם כי יישומים מסוימים עשויים לבצע פונקציות שכבת האפליקציה בעצמם.) פרוטוקולים נפוצים לשכבה זו הם HTTP,FTP,DNS, DHCP.
2. Transport (תעבורה) – שכבה זו אחראית על אופן שבו תעבור החבילה ברשת (שכבה זו **<u>לא</u>** אחראית ממש על העברת המידע) לדוגמא, האם יש שגיאות בחבילות שהגיעו וכן, האם המידע יעבור בפרוטוקול TCP ששם דגש על כך שהמידע יעבור בצורה אמינה בכך שהוא ידרוש הודעה על קבלת כל המידע או שהמידע התקבל בצורה שגויה אם בכלל הוא התקבל, אך כל הבדיקות האלו יעלו למשתמש במהירות התקשורת. או האם להשתמש בפרוטוקול UDP שפחות שם דגש על אמינות ולא יבקש הודעת משוב על קבלת המידע במחשב היעד אך הוא גורם לתקשורת מהירה יותר בכך הוא מוותר על הבדיקות.
3. Network (רשת) - שכבה זו מטפלת בפנייה וניתוב של הנתונים (שליחתו בכיוון הנכון ליעד הנכון על שידורים יוצאות וקליטת שידורים נכנסים ברמת החבילה).כמו כן, שכבה זו אחראית על הסוג המעבר – האם מאחד לאחד (Unicast), מאחד או כמה לקבוצה של אחרים(Multicast), או מאחד לכל האחרים (Broadcast) IP הוא פרוטוקול בשכבת הרשת עבור האינטרנט. בשכבה זו תמצא כתובת ה IP שבמהלך העברת החבילה הנתב ישתמש בה בכדי לדעת לאן להעביר את החבילה הזו.
4. Link (קו) – שכבה זו מכניסה את החבילות לתוך המסגרת שמתאימה להם וכן מקבלת הודעות על כך שהמידע התקבל למי שהיא יועדה. בשכבה זו תהיה כתובת ה MAC של המכשיר, וכשהחבילה תגיע למתג (Switch) הוא ישתמש ב MAC ע״מ שיוכל לדעת לאן לשלוח את החבילה.
5. Physical (פיזי) – שכבה זו מעבירה את זרם הביטים דרך הרשת ברמה החשמלית, האופטית או הרדיו, כמו כן היא מגדירה את האותות של התקשורת. היא מספקת את אמצעי החומרה של שליחת וקבלת הנתונים ברשת של ספק.

כל חבילה שנשלחת ברשת עוברת תהליך כזה פעמיים. פעם אחת ביציאה שלה מהמחשב, בשלב זה היא ״נעטפת״ בכל השכבות הנ״ל ואז היא יכולה לנתב בתוך הרשת עד ליעד שלה (שלב זה של ״עטיפה״ נקרא Encapsulation). פעם שניה, במעברה ברשת לכיוון היעד. בחלק מהנתונים משתמש הנתב בדרך, בחלק משתמש ה Switch וחלק מחשב היעד עצמו (שלב זה נקרא Decapsulation).

ניתן לראות את החבילות ואת מרכיבי כל אחת מחבילות לשכבות שלה ע״י WireShark



החלק האדום: החבילות שנתפסו. כל שורה מציגה את הזמן שהחבילה נתפסה את הכתובת המקור והיעד שלה את הפרוטוקול שבו היא מועברת ועוד.

החלק הירוק: הפרטים שמתקבלים ע״י לחיצה על אחת החבילות ושם אנו מקבלים מבט כללי על החבילה – על השכבות של החבילה. כל אחת מהשורות מייצגת שיכבה במודל חמשת השכבות. השורה הראשונה מייצגת את השכבה הפיזית וכו׳ עד שהשורה החמישית מייצגת את שכבת האפליקציה.

החלק הכחול: מציג את כל אחת מהשכבות בייצוג ההקסדצימלי שלה.

כעת ניכנס לעומק בכל אחת מהשכבות.

Application

שכבת האפליקציה היא השכבה הראשונה בתהליך לשליחת המידע היא פועלת בתוך המחשה עצמו. במכשיר המקבל, שכבה זו פוגשת אחרונה את החבילה ויודעת בסוף איזה יישום להפעיל בהתאם לחבילה.

שכבת זו בעצם מכינה את החבילה לשליחה ומצרפת אליו מידע כגון לאיזה פורט הוא שייך איזה סוג פרוטוקול תקשורת להשתמש, איזה רוכב פס צריך להשתמש, מגדירה את תהליך עיבודן של ההודעות

Port

כל חבילה מתחילה במחשב יוצאת לרשת הפנימית (LAN) ואם צריך יוצאת לרשת החיצונית על מנת להגיע למכשיר היעד שלה. כפי שהוסבר, בכדי שהחבילה תדע להתמצא ברשת החיצונית היא מקבלת כתובת IP של המקור והיעד שלה ובעצם ע״פ כתובת זו הנתבים יודעים לאיזה מקום לשלוח את החבילה. בתוך הרשת הפנימית ה Switch משתמש בכתובת ה MAC שנתנה לחבילה ובכך הוא יודע לאיזה מכשיר בתוך הרשת לשלוח את החבילה. אך ברגע שהגענו לתוך המכשיר כיצד נדע לאיזה תהליך שייכת החבילה, והרי זה יכול להיות אימייל, גלישה באינטרנט או מעבר קבצים? לכן הוסיפו פרמטר שנקרא Port, פרמטר זה הוא מספר שמייצג את התהליך, ועכשיו שתגיע חבילה המכשיר יחפש את המספר הזה וידע לאיזה תהליך לגשת.

מספרי הפורטים מתחלקים לשני חלקים האחד מ 0 עד 1023 (כולל) והשני זה כל השאר עד 2^{16}.

בחלק הראשון ייצג בד״כ פורטים ידועים שכולם יסכימו עליהם (לדוגמא פורט 80 מייצג את פרוטוקול http) וזאת על מנת שכאשר מכשיר ירצה לפתוח פניה לשרת הוא לקבלת שרות מסוים הוא ידע לאיזה פורט לפנות ע״מ לקבל שירות זה, אך כיוון שלמכשיר הפונה אין צורך להיות חשוף לפניות כל הזמן אז

כאשר הוא יפתח תהליך הוא יבחר בצורה אקראית מספר מהטווח השני. ומספר זה ייצג את התהליך שלו מול הצד השני (כדי שכשהצד השני יחזיר הודעה הוא ידע לאיזה תהליך לפנות) עד לסיום ההתקשרות.

אחד מהפרוטוקולים בשכבת האפליקציה הוא HyperText Transfer Protocol - HTTP – פרוטוקול זה משמש להעביר כמעט את כל הקבצים ונתונים אחרים על World Wide Web, בין אם הם קבצי HTML, קבצי תמונה, תוצאות שאילתה, או כל דבר אחר. בדרך כלל, HTTP מתרחש באמצעות TCP/IP socket (socket הינו צינור שמקשר בין שכבת האפליקציה לשכבת התעבורה).

כשאר מבקשים דף אינטרנט זה נעשה בחלק מהמקרים ע״י הכנסת כתובת ה URL בדפדפן

(לדוגמא: http://www.solver.com/integer-constraint-programming) כתובת זו מורכבת מהפרוטוקול ומה URN (לדוגמא: www.solver.com/:integer-constraint-programming) והמיקום של הקובץ באתר נקרא URI (לדוגמא: integer-constraint-programming). כאשר מבקשים דף מסוים פונים לשרת מקבלים את הדף HTML הבסיסי ואז המכשיר רואה שישנם בדף זה עוד אובייקטים כגון תמונות וכו׳ אז הוא מבקש כל אחד מהם מהשרת.

בין השרת והלקוח מתנהל בעצם דו שיח,

<u>הודעת ה HTTP של הלקוח לשרת</u> נראית כך:

```
GET /favicon.ico HTTP/1.1
Host: gaia.cs.umass.edu
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/                Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
```

שורה ראשונה מכילה את סוג ההודעה (POST/GET) אח״כ את מיקום ושם הקובץ המבוקש ואז את גרסת ה HTTP. (במקרה שלנו 1.1)

שורה שניה (HOST) מכילה את שם של האתר שמארח את הקובץ.

שורה שלישית (Connection) אומרת האם הקשר יישאר פתוח או לסגור אותו (Close).

שורה רביעית (User-Agent) מכילה את שם הדפדפן ומערכת ההפעלה שלו.

שורה חמישית (Accept) מכילה את הפורמט של המידע שמתקבל.

שורה שישית (Referer) את המקור של הקובץ.

שורה שביעית (Accept-Encoding) מכילה את הפורמט של השפה שכתובה באתר.

שורה שמינית (Accept-Language) איזו שפה יש כתובה באתר (עברית, אנגלית וכו׳).

<u>הודעת ה HTTP של השרת ללקוח</u> נראית כך :

```
HTTP/1.1 200 OK
Date: Fri, 27 Oct 2017 13:55:01 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3
Last-Modified: Fri, 27 Oct 2017 05:59:02 GMT
ETag: "51-55c80fc4ded2e"
Accept-Ranges: bytes
Content-Length: 81
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

שורה ראשונה מכילה את גרסת ה HTTP. (במקרה שלנו 1.1) וכן הודעת סטטוס על החבילה. (במקרה שלנו ההודעה התקבלה (OK) וקוד הסטטוס הוא 200, ראה הרחבה בהמשך)

שורה שניה (Date) מכילה את של שליחת ההודעה.

שורה שלישית (Server) את שם וגרסת השרת.

שורה רביעית (Last-Modified) מכילה את התאריך האחרון לשינוי הדף.

שורה חמישית (ETag) משמשת לאימות מטמון אינטרנט.

שורה שישית (Accept-Ranges) מכילה את היחידות שבהם מגדירים את הקובץ (אצלנו בביטים).

שורה שביעית (Content-Length) מכילה את מספר הביטים של הקובץ שהוחזר ע״י השרת.

שורה שמינית (Keep-Alive) מאפשרת לשולח לרמוז על אופן החיבור וניתן להשתמש בו לקביעת פסק זמן (לניתוק הקשר) וכמות מקסימלית של בקשות. (פסק הזמן הדיפולטיבי הוא 15 שניות, וזאת כדי לאפשר בקלות יותר למערכת לשלוח את כל מרכיבי דף האינטרנט בלי לצרוך משאבים רבים)

שורה תשיעית (Connection) קובעת האם חיבור הרשת נשאר פתוח לאחר סיום העסקה הנוכחית או שייסגר.

שורה עשירית (Content-Type) מכילה את סוג הקובץ שנשלח ע״י השרת.

(להרחבה : https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers)

כפי שהוזכר לעיל, בהודעות תגובה מהשרת יש שדה שמראה את סטטוס ההודעה – התקבל, לא נמצא וכו׳. הודעות אלו מיוצגות ע״י מספרים בין 100 ל 500. כאשר טווח זה מחולק לחמישה חלקים,

מספרים מהצורה 1xx מיצגים הודעות מידע.

מספרים מהצורה 2xx מיצגים הודעות הצלחה.

מספרים מהצורה 3xx מיצגים הודעות ניתוב מחדש.

מספרים מהצורה 4xx מיצגים הודעות שגיאת לקוח.

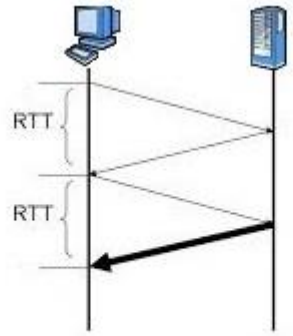
מספרים מהצורה 5xx מיצגים הודעות שגיאת שרת.

וע״י כך המקבל יוכל לדעת את טיב ההודעה ומידע נוסף על השיחה.

(להרחבה : https://developer.mozilla.org/en-US/docs/Web/HTTP/Status)

את ה״שיחה״ מול השרת אפשר לנהל בכמה שיטות שונות שתלויות בגרסת פרוטוקול HTTP אך כולם בעלי מחנה משותף של פתיחת קשר ובקשת הקבצים המתאימים.

נסמן ב RTT את הזמן שלוקח להודעה בודדת להישלח מהמכשיר לשרת וחזרה. ונסמן ב F/R את זמן העברת הקובץ



https://en.wikibooks.org/wiki/Communication_Networks/HTTP_Protocol

בתמונה לעיל משך הזמן הוא 2*RTT + F/R.

<u>גירסה 1.0 של HTTP</u>

לכל אובייקט שיבקש הלקוח הוא יצטרך לפתוח פניה לשרת (RTT) ואז לשלוח הודעת בקשה לאובייקט (RTT) ולסגור את השיחה ולא לבקש עוד אובייקט עד שהקודם יגיע, דבר זה גורר (לדוגמא) שאם הוא ירצה לקבל 100 אובייקטים הוא יצטרך לעשות סה״כ 100*(2*RTT + F/R) אך זה מאד ״יקר״.

אפשר ליעל את השיטה הזו ע״י כך שלא תיסגר השיחה עד קבלת כל האובייקטים וכן שיהיה אפשר לבקשת כמה אובייקטים בו זמנים, ולא לחכות עם בקשת אובייקט עד שהאובייקט הקודם יגיע.

<u>גירסה 1.1 של HTTP</u>

גרסה זו היא שיפור לעומת הקודמת כיוון שהשיחה לא תיסגר כל עוד לא סיים הלקוח לבקש כל מה שהוא רוצה. וזמן הפעולות שהוא (בהתייחס לדוגמא של 100 אובייקטים) 100*(RTT + F/R)+RTT (זמן חיבור + זמני בקשות אובייקטים), קטן יותר מהקודם. אך גם פה אפשר לשפר את זמן השיחה בכך שנוכל לשלוח בקשה לאובייקט חדש בלי לחכות לתגובה על הגעת האובייקט הקודם.

<u>גירסה 1.1 של Pipe HTTP</u>

בגרסה זו אפשר לבקש כמה אובייקטים במקביל מבלי לחכות לתגובה על בקשת האובייקט הקודם.

ולכן זמן השיחה הוא (בהתייחס לדוגמא של 100 אובייקטים) 100*(F/R)+RTT.

הערה: הדף HTML הזמנים יתחלקו לשלושה חלקים, פתיחת השיחה (RTT) בקשת דף ה HTML הבסיסי (RTT + F/R) ואז בקשת כל האובייקטים באותו דף כגון תמונות פרסומות וכו׳

(כמות-אוביקטים*(F/R) + RTT).

שיפורים אפשריים

<u>אבטחה</u>

על מנת לאבטח מידע הרבה פעמים מבקשים מהמשתמש להכניס שם משתמש וסיסמה.

כאשר המשתמש רוצה להשתמש במידע השרת מבקש ממנו את שם המשתמש והסיסמה ואז המשתמש שולח בחזרה את המבוקש.

במודלים הראשונים של בקשת ההזדהות, היה השרת מבקש והלקוח עונה לו בצורה גלויה. אך דבר זה היה פרוץ, וכל אחד יכל להכניס את עצמו לתקשורת בין השרת ללקוח (Man in the middle attack) וכך לשמוע את כל תעבורת הרשת ולראות מה שולחים וכך לדעת את פרטי ההזדהות של הלקוח. לכן נדרשה איזו הצפנה למידע ששולח הלקוח. את ההצפנה יבצע הלקוח לפי קוד ששולח לו השרת, לשדרוג זה יש כבר מעלה על פני קודמו כיוון שהמידע שעובר מהלקוח לשרת מוצפן אך אפשר לשמור את הקוד ששולח השרת ללקוח וע״י כך לדעת כיצד הוצפן המידע.

לכן שדרגו ופיתחו פרוטוקול שנקרא HTTPS שפועל כך:

1. השרת מבקש תעודת נאמנות מארגון מוסמך שמאשר שהוא אמין.
2. השרת מבקש זיהוי מהשרת.
3. השרת מחזיר את תעודת הנאמנות שלו.
4. הלקוח בודק מול הארגון המוסמך את תעודת הנאמנות של השרת.
5. אם התעודה אמיתית, אזי הלקוח יודע שהוא יכול לשלוח בבטחה את שם המשתמש והסיסמה.
6. השרת מאמת את פרטי ההזדהות.

אך גם שיטה זו אינה בטוחה לגמרי כיוון שאפשר לזייף את תעודת הנאמנות.

(ישנם עוד שיטות להתגבר על בעית האבטחה, אך עדין לא נכנסנו לפתרונות אלו)

Web Sessions

אחד מהדברים שמאטים את התעבורה ברשת זה פתיחת השיחה וסגירת. כמו כן היה עוזר לשרת להשאר פתוח כדי לזכור את הלקוח שלו, דבר שיעזור לשרת לנהל יותר טוב את המידע שלו. אך את החיבור אי אפשר להשאיר כל הזמן פתוח כיוון שהשרת יצטרך להחזיק המון שיחות פתוחות (דבר שעוד יותר יכביד עליו) וכן מכשירים מחליפים לפעמים כתובות IP ואז מתנתקת השיחה הזו.

ע״מ להפחית את התעבורה וכן כדי שהשרת יוכל לזהות בקלות את הלקוח המציאו את העוגיות (cookies). העוגיות הן קבצים שהשרת מציב על המכשיר של הלקוח שמאוכסן בהן מידע של השרת על הלקוח וכך בכל זמן שהלקוח ירצה לגשת לשרת הוא יצרף את העוגייה והשרת ידע מיד לזהותו.

נוכל לראות במסגרת האדומה כיצד נראית חבילה שמכילה עוגייה.

```
Frame 1528: 742 bytes on wire (5936 bits), 742 bytes captured (5936 bits) on interface 0
Ethernet II, Src: LiteonTe_23:ef:ae (3c:95:09:23:ef:ae), Dst: Zte_cc:12:18 (ec:1d:7f:cc:12:18)
Internet Protocol Version 4, Src: 192.168.0.197, Dst: 88.221.156.110
Transmission Control Protocol, Src Port: 50170, Dst Port: 80, Seq: 1, Ack: 1, Len: 688
Hypertext Transfer Protocol
> GET /DidYouKnow/Images/051807_cookies001.jpg HTTP/1.1\r\n
  Host: www.webopedia.com\r\n
  Connection: keep-alive\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36
  Accept: image/webp,image/apng,image/*,*/*;q=0.8\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9\r\n
```

Web proxy

דרך נוספת לשפר את מהירות העברת המידע בין השרת ללקוח היא ע״י שרתי proxy. מטרת שרתים אלו היא לאחסן את המידע הפופולרי והסטטי (שלא משתנה) קרוב ללקוח ובכך את המידע שמבקשים הכי הרבה פעמים לא יצטרכו לבקש מהשרת שממוקם רחוק, אלא יוכלו לשלוף אותו ממקום קרוב ללקוח.

כמו כן ברגע בקשת החבילה ע״י המכשיר, יבדק האם המכשיר ביקש את הבקשה הזו בעבר והאם יש לו את הגרסה העדכנית ביותר לאובייקט שהוא מבקש אם כן אין צורך לבקש שוב את החבילה. את תאריך העדכון האחרון של האובייקט בשרת אפשר יהיה לבדוק בשדה last-modified שמופיע בחבילה.

היתרון לשיטת של בדיקת התאריך הוא שלא יבקשו שוב את הדף אם כבר ביקשו אותו בעבר, דבר שמפחית את התעבורה ברשת.

Http לעזרת וידאו

עם הזמן פיתחו שימושים נוספים לפרוטוקול HTTP, אחד מהם הוא עזרה לצפייה בסרטונים ברשת. ע״מ שיוכלו לצפות בסרטונים ברצף בלי שיתקעו אך עם האיכות הטובה ביותר, השרת מאכסן כמה איכויות של הסרטון (MPD) ואז הלקוח בודק בכל פרק זמן איזה איכות מקסימלית הוא יכול לקבל בהתאם לרוחב הפס שלו. וכך הוא יוכל לקבל את הסרט בצורה שמתאימה לרוחב הפס שלו אך גם באיכות המקסימלית שמתאימה לו באותו רגע לפי העומס על הפס שלו.

DNS - Domain Name System

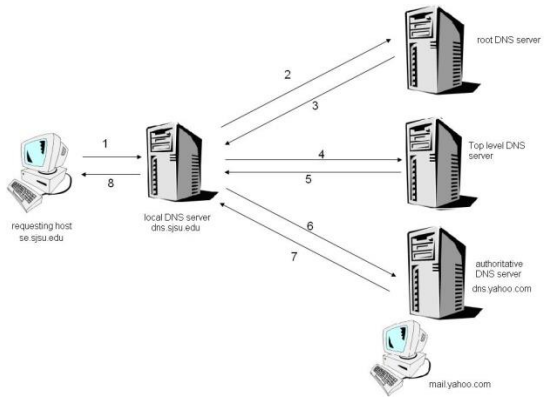
עד עכשיו דובר שהרשת מתקשרת ע״י כתובות IP, אך כאשר המשתמש נכנס לדפדפן ורוצה להיכנס לאתר כלשהו הוא לא מכניס את כתובת ה IP של האתר אלא מילה שבנויה מאותיות ומספרים (Domain name) שלרוב נוכל לקרא (כגון www. example.com). את ההתאמה בין כתובת זו לבין כתובת ה IP עושה שרת ה DNS. שרת ה DNS אינו שרת יחיד בעולם אלא רשת שרתים שבנויה כעץ. בשורש העץ נמצאים כ 13 שרתים שמחוברים בניהם, ונקראים Root domain, אליהם מגיעים פניות משרתי שמות מקומיים שלא יכולים להמיר את השמות לכתובת ה IP. מתחתיהם מגיעים שרתי ה Top Level Domain – TLD שרתים אלו אחראים על אתרים עם סיומות כגון com,org,net,edu וכן על סיומות של מדינות כגון il, uk, fr, ca, jp. מתחתיהם קיימים שרתי DNS לארגונים שמתוחזקים ע״י הארגון שנותנים שירות המרה זה. כמו כן יש שרתי שמות מקומיים שנותנים שירות להמרה זו. לקבוצת השרתים התחתונה קוראים שרתים מהימנים. (ישנה גם במכשיר הפרטי טבלת DNS קטנה שיודעת קצת לבצע את ההחלפות לכתובת IP.

ריבוי רמות זה נצרך מכמה סיבות, האחת היא הורדת העומס משכבה אחת וחלוקתו לכמה שכבות והשנייה היא שבמקרה ושרת אחד נופל הוא אינו מפיל יחד אתו את כולם. וכאשר משנים את כתובת ה IP של אתר מסוים אז לא צריך לשנות את כל השרתים שנמצאים בעץ זה אלא לרוב מספיק לשנות את השרתים שקרובים לשרתים שבהם השתנתה הכתובת.

אלגוריתם הבקשה לשירות DNS מתבצע בשתי צורות איטרטיבי ורקורסיבי.

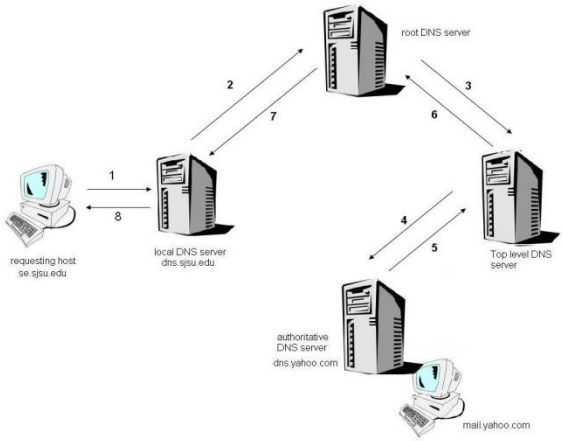
DNS איטרטיבי

המכשיר מבקש שירות DNS מהשרת DNS שלו. אם השרת יודע לבצע את ההמרה אז הוא מחזיר תשובה למכשיר. במקרה והוא לא יודע אז הוא שואל שרת אחר, השרת האחר מחזיר או את התשובה או איזה שרת הוא חושב שיודע את התשובה ואז שוב השרת של המכשיר שואל ע״פ התשובה שניתנה לו את השרת המתאים ושוב מחזירים לו תשובה עד שמגיעים לשרת שיודע לבצע את ההמרה. (כאשר בתמונה לקמן השרת שיודע את התשובה הסופית מוסר אותה בשלב 7 ואז בשלב 8 זה הגיע למכשיר הפרטי)



DNS רקורסיבי

המכשיר מבקש שירות DNS מהשרת DNS שלו. אם השרת יודע לבצע את ההמרה אז הוא מחזיר תשובה למכשיר. במקרה והוא לא יודע אז הוא שואל שרת אחר, השרת האחר שואל את השרת הבא בתור אחריו והוא שואל את הבא בתור אחריו עד שמחזירים תשובה ואז התשובה עוברת את כל השרתים (בסדר הפוך להגעת) עד שמגיע חזרה למכשיר שביקשת את השירות (כאשר בתמונה לקמן השרת שיודע את התשובה הסופית מוסר אותה בשלב 5 ואז בשלב 8 זה הגיע למכשיר הפרטי)



לגל רשומה של DNS יש זמן שאחריו הרשומה הזו נמחקת וזאת בין השאר ע״מ להוריד עומס מהשרת של ה DNS. זמן זה מסומן כ TTL – Time To Live.

מבנה בקשת DNS כולל כמה שדות, בין השאר שדה זיהוי של 16 סיביות שנוצר על-ידי המכשיר שיוצר את שאילתת ה- DNS. זה מועתק על ידי השרת לתוך התגובה, כך שניתן להשתמש בו על ידי המכשיר כי כדי להתאים את השאילתה לתשובה המתאימה שהתקבלה משרת DNS. דבר שני הוא דגל מבחין בין שאלות ותשובות. מוגדר כ- 0 כאשר השאילתה נוצרת ; השתנה ל -1 כאשר שאילתה זו השתנתה לתגובה על ידי שרת משיב. וכן השאלה עצמה. (ישנם עוד שדות שמרכיבים את ההודעה אך לא נכנסנו אליהם)

את שאלות ה DNS מבצעים בדרך כלל ע״י פרוטוקול UDP כדי למנוע עיכובים בבקשות.

יחס כתובות שרתים

על אותו שרת יכולים להיות כמה כתובות, מה שאומר שכל בקשות ה DNS יקבלו את אותה כתובת IP ובתוך השרת יעשו את ההפרדה לכל אתר ואתר.

וכן יכול להיות שלאותה כתובת אתר (Domain name) יש מספר שרתים עם כתובות שונות. את החלוקה הזו יעשו לרוב אתרים גדולים מאד (כגון Google). כאשר יפנו לאתר זה הוא יכול לתת כל פעם כתובת של שרת שונה שלו וזאת ע״מ שהוא יוכל לחלק את העומס שלא יהיה על שרת בודד.

התקפות DNS

ישנן מספר התקפות נפוצות שבאות על שירות ה DNS.

- DDoS – מתקפה זו מבוצעת ע״י שליחת כמות רבה מאד של בקשות בו זמנית לאותו שרת וכך השרת קורס תחת עומס הבקשות ולא יכול להגיב. מתקפות אלו מתבצעות לרוב ע״י גופים שרוצים להשבית אתר מסוים או שרת.
- DNS Hijacking – מתקפה זו היא מתבצעת כאשר גוף שלישי נכנס לשיחה בין הלקוח לשרת ה DNS (כאיש שבאמצע) וכך הוא יכול לענות ללקוח במקום שרת ה DNS ולתת לו תשובות שגויות לשאלות DNS של הלקוח. בצורה זו יוכל ה״פורץ״ להפנות את הלקוח למקומות שה״פורץ״ רוצה ולהכניס ללקוח מידע שלא ביקש. כמו כן, לא חייבים להיות ה״האיש שבאמצע בכדי לבצע זאת אלא אפשר גם לשנות את טבלת ה DNS שנמצאת בתוך המכשיר לשנות אותה וכך לבלבל את הלקוח.

מקורות

1. מצגות של ד״ר דביר עמית זאב
2. Wikipedia
3. http://searchnetworking.techtarget.com/definition/OSI