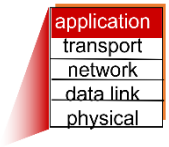
תקשורת מחשבים – סיכום

DELAY

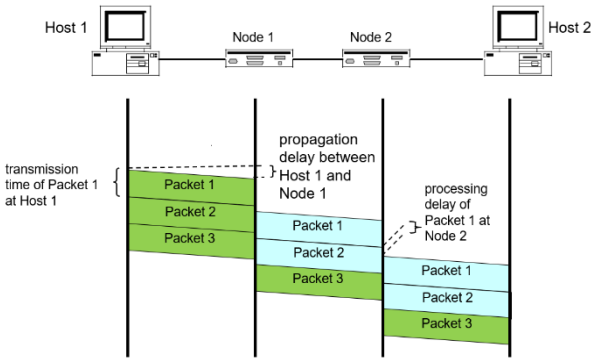


d_{trans}: transmission delay:
- L: packet length (bits)
- R: link *bandwidth (bps)*
- $d_{trans} = L/R$

d_{prop}: propagation delay:
- d: length of physical link
- s: propagation speed ($\sim 2 \times 10^8$ m/sec)
- $d_{prop} = d/s$

Packet Switching (מיתוג חבילות)- נתב (router) מקבל חבילה (packet) ועליו לקבוע איזה חבילה להעביר ולאן. נעשה בשיטת FIFO וכאשר חבילה תקועה בנתב זמן רב היא תעבור לשיטת LIFO. כמות גדולה של מידע ומשתמשים, לא תמיד החלטות הניתוב טובות מה שיכול לגרום לעיכוב בהעברת מידע על הקו, הקו פנוי וניתן לשימוש (מייעל את השימוש).

Circuit Switching(מיתוג מעגלים)- החיבור בין המוצא אל היעד נעשה בחיבור ישיר, כלומר אין נתבים שצריכים לקבל החלטות בדרך, הכול נעשה בזמן אמת, ידוע ומוקצב מראש. השיטה מוגבלת בכמות המשתמשים. המידע יגיע בוודאות בזמן ידוע. כאשר הקו מחובר הוא תפוס גם אם אין העברת מידע(ניצול גרוע של רוחב הפס).



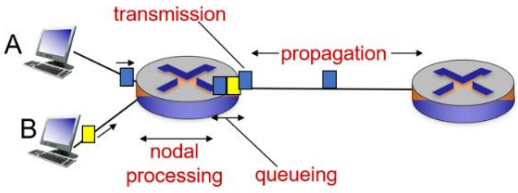
Packet Delay- עיכוב בשליחת נתונים מנתב לנתב תלוי ב-4 גורמים:

Propagation Delay- התפשטות מנתב לנתב (מהירות זרימת הנתונים).

Transmission Delay- שידור, רוחב פס. העיכוב תלוי באורך החבילה (L bits) לעומת רוחב הפס(R): dtrans=L/R.

Queueing Delay- עומס שנשאר לקו כורם לתור של העברת החבילות (קצב הכניסה לעומת קצב היציאה).

Nodal Processing- קבלת החלטות של הנתב, בדיקת החבילה והיעד שלה.



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

200 OK
- request succeeded, requested object later in this msg

301 Moved Permanently
- requested object moved, new location specified later in this msg (Location:)

304 Not Modified
- requested object was not modified since it was cached

400 Bad Request
- request msg not understood by server

404 Not Found
- requested document not found on this server

505 HTTP Version Not Supported

שכבת האפליקציה (Application Layer)

אפליקציה – תוכנה שיודעת לעבור "לדבר" בשפה מקובלת שתומכת ברשת– (HTTP,SMTP,FTP) היא מכילה את המידע שיועבר לידע.

ארכיטקטורת אפליקציית רשת - server-client חיבור בין יחידות קצה (clients) בעזרת שרתים. כלומר כל המידע עובר קודם דרך שרת, ורק אחר כך המידע מוצג ביחידת הקצה. זאת הארכיטקטורה הכי מקובלת באפליקציות רשת.

ארכיטקטורת P2P (Peer To Peer)- החלפת סרברים ביחידות קצה של הלקוחות. כלומר, ניצול של המשאבים של שאר הלקוחות לטובת יחידות קצה הנדרשים לכך. התקשורת מתבצעת בין 2 משתמשי האפליקציה .

Process communication- כאשר 2 אפליקציות רוצות לדבר אחת עם השנייה באותו המחשב, מערכת ההפעלה משחקת את תפקיד השרת. כלומר היא מעבירה מידע בין ה-client process ל-server process.

Addressing Processes- לכל יחידת קצה מאחרת יש כתובת IP בעל 32 ביט. כשמנסים לגשת לאפליקציה ביחידת קצה מסוימת, לא מספיק לציין רק את כתובת ה IP-של יחידת הקצה, אלא גם לציין לאיזו אפליקציה לגשת. בשביל זה יש מזהה בשם פורט Port – לכל אפליקציית רשת.

Socket-הממשק שאחראי על שליחת ההודעות בין אפליקציות (כמו קובץ), הצורה שבה מבקשים ממערכת ההפעלה לשלוח הודעה ליחידת הקצה האחרת. כשפותחים socket לשליחת נתונים צריך לציין איזה סוג תקשורת רוצים, TCP/UDP) כתובת ה IP-של הלקוח (side client), מספר פורט של האפליקציה שלי ניתן ע"י מערכת ההפעלה, כתובת ה IP-ביחידה איתה רוצים לתקשר, ומספר הפורט של אותה אפליקציה איתה רוצים לתקשר.

Socket ID-Protocol(UDP/TCP),My-IP-Address,My-Port-Number,Peer-IP-Address,Peer-Port-Number.

שירותים דרושים עבור שכבת האפליקציה משכבת התעבורה- תזמון(Timing),ביטחון(Security),אמינות מידע(Data Integrity),תפוקה(Throughput).

שכבת התעבורה (Transport Layer)

Transport-קישור בין אפליקציות ביחידות הקצה, אחראי על המידע שעובר(TCP, UDP) .

Internet Transport Protocol Servers

UDP-שירות לא אמין בכל הצרכים של התעבורה, לעומת זאת העלות שלו נמוכה, כלומר אין הבטחה עזה שהמידע יגיע ליעדו בזמן וכו'.

UDP segment format- Length- אורך הסגמנט בבתים כולל ה header-אורך של הודעה מקסימלי בפרוטוקול זה היא 64 קילובייט(2^16 bit) לשדה ה - length מוקצים 16 ביטים.

Checksum- בודק שההודעה שהתקבלה היא באמת ההודעה שנשלחה. שדה ה checksum-מכיל את אותו המספר בן 16 ביטים שמכיל את תוצאת החישוב על תוכן ההודעה (application data)כאשר ההודעה מתקבלת עושים חישוב בעזרת אותה פונקציה ידועה ומשווים את התוצאה, אם התוצאה זהה לערך ה checksum-אז כנראה שההודעה שהתקבלה תקינה.

FTP- פרוטוקול המשמש להעברת קבצים, מפעיל קישור נפרד לאיתות וקישור נפרד להעברת הקובץ.

TCP-אמינות גבוהה בין שליחה לקבלת המידע, סדר שליחת המידע והצורה שבו הוא נשלח .שני הפרוטוקולים הללו אינם מצפינים את המידע שמועבר בעזרתן. בפרוטוקול הזה כאמור ישנה אמינות checksum ,שבודקת תקינות ההודעה. ובנוסף אם ההודעה " הולכת לאיבוד" כלומר לא מגיעה ליעדה,

ההודעה נשלחת שוב (ישנה כמות מסוימת של ניסיונות חוזרים לשליחת ההודעה, ברגע שמגיעים לכמות זו ההודעה פשוט לא תישלח). בעבודה עם פרוטוקול זה , קיים סדר בהגעת הסגמנטים לפי סדר שליחתם .ב TCP-ניתן לקרוא, stream-byte כלומר את ההודעה שמתקבלת ניתן לקרוא עפ"י כמות בתים מוגדרת ללא שום קשר איך ההודעה נשלחה. (כלומר אם כל סגמנט מכיל 200 בתים, והצד שקורא, קורא את ההודעה כל 120בתים, אין שום בעיה עם זה). **Flow Controlled -** בקרת "הצפה", מניעת מצב שבו השולח שולח יותר מידע ממה שהצד המקבל מסוגל לקלוט. דבר אשר עלול לגרום לעומס. **Congestion Controlled-**תפקידו לזהות שיש עומס על הרשת, כלומר שההודעות מתעכבות בתורים יותר מידי, ובעצם במצב כזה מתבקש הצד השולח לשלוח פחות מידע ולכן אומנם ישלח פחות מידע בבת אחת ,אבל הוא יתקבל בעיכוב נמוך יותר.

TCP Header- Sequence Number-מספור לפי הסדר של הסגמנטים,Acknowledgment Number-הודעת אישור,Checksum-תוצאת חישוב של פונקציית ה-checksum,Receiver Window-מודיע כמה זיכרון פנוי יש לצד המקבל לקבלת ההודעה, כלומר מצב "הצפה "(הזיכרון הפנוי) של הצד המקבל,Options-מילה אופציונלית,Head Length-גודל ההדר(שדה בן 5 ביטים),6, Urg Data Pointer, Not used-ביטים-UAPRSF-FIN,Reset,SYN,ACK,PSH,URG.

Timer- קבוע את משך הזמן הסביר להמתין לאישור קבלת ההודעה. נקבע ע"י הזמן, RTT כלומר הזמן שלקח מאז ששלחנו בקשת פתיחת קו עד שקיבלנו אישור לפתיחת הקו(syn -> syn-ack)

SSL- שירות הצפנה שניתנת על ידי מערכת ההפעלה למידע המועבר, בעלות של זמן וגודל המידע. דרך הצפנה יחסית חלשה.

HTTP (Hyper Text Transfer Protocol)- פרוטוקול שמשמש אפליקציות שרוצות לפנות לשרת- פותח במטרה להעביר עמודי web ממקום למקום(Hyper Text,URL,HTML). הגישה לשרתי ה HTTP-מתבצעת על ידי תקשורת TCP לפורט 80 בשרת.(HTTP is stateless) כלומר הפרוטוקול הזה לא תוכנן לאחסן מידע לגבי בקשות קודמות.

חיבור Persistent HTTP- ניתן לשלוח מספר אובייקטים בחיבור TCP יחיד בתקשורת בין לקוח לשרת .לעומת **חיבור Non-Persistent HTTP** ששם לאחר כל חיבור ה TCP-נסגר כלומר להעברת מספר אובייקטים נדרשות מספר חיבורים.

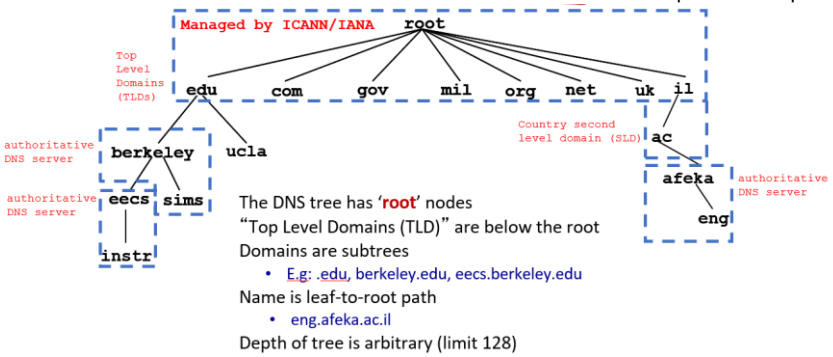
פרק הזמן RTT (Round Trip Time)- הזמן שלוקח להודעה לצאת מהלקוח עד השרת והדרך חזרה, לרוב ההודעות הן קטנות ("לחיצת יד", הבאת HTML ,אובייקטים קטנים)לכן כול תהליך ייקח זמן RTT מסוים מהתחלה עד הסוף .עבור אובייקטים גדולים, ייקח זמן RTT לביט הראשון של האובייקט להגיע לשרת מהלקוח, ולכן הביט הבא אחרי יגיע בזמן transmission אחרי הביט שלפניו, כלומר עבור אובייקט גדול ייקח לו:

RTT + transmission delay = number of bits

User-Server state: Cookies- כל דפדפן שתומך עבודה עם cookies שומר ב file-cookie-במחשב של הלקוח פרטים על האתר. ה cookie file -הוא מסד נתונים ששומר את שם האתר ומספר מזהה של הלקוח. כאשר האתר מזהה שלקוח ניגש לאתר וללקוח אין מספר מזהה הוא מעניק לו מספר זיהוי חדש, תחת מספר הזיהוי הזה ישמרו הפרטים על הלקוח .הסיכון בשמירת cookies הוא אוסף מידע ע"י אתרי מכירות.

Web Cashes (Proxy Server)-שרת proxy הוא בעצם שרת ביניים בין הלקוח לבין שרת הראשי אליו הלקוח פונה. במקרה כזה מידע (אתר ,תמונה וכו') נשמרים בשרת ה proxyעל מנת לייעל ולזרז את תהליך העברת וקבלת נתונים. כך שלמעשה כול המידע שהלקוח ביקש מאתר מסוים ייקח פחות זמן להביא, במידה ואותו מידע שמור על ה. proxyכמובן שמידע שמתיישן בשרת ה , proxy מבחינת פניות אליו, ימחק מהשרת ויתפנה הזיכרון. היות ויש אובייקטים שמתחדשים\משתנים שרת ה proxyצריך לדעת להתעדכן על מנת שתעבורה של המידע תהיה יעילה. רוב האובייקטים שידוע שהם עתידיים להתעדכן בשלב מסוים, נושאים איתם מידע שכולל את זמן העדכון וכך שרת ה proxy יכול לעדכן את המידע כך שיהיה זמין ללקוח. היות ששרת ה proxy חוסך פניות של לקוחות ישירות לשרת היעד זה לטובתם של שרתי היעד לשלוח מידע לגבי עדכון האובייקט. האסטרטגיה השנייה של עדכון המידע היא **conditional get**.במידה ולא ידוע מתי האובייקט יתעדכן, שרת ה-proxy ייגש לשרת וישאל אותו אם אותו עמוד מבוקש השתנה, אם כן ה-proxy יעדכן את המידע הקיים וישלח אותו ללקוח, במידה ולא ה-proxy ייתן ללקוח את המידע הקיים אצלו.

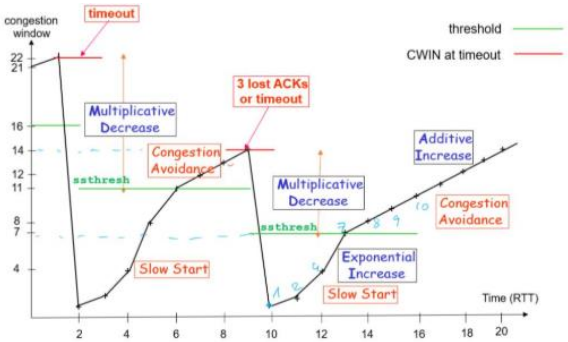
DNS-Domain Name System- תפקיד מערכת ה DNS-הוא לתרגם את שמות הכתובות של אתרי האינטרנט URL) עד ה-'/' הראשון בכתובת ,(לרצף ה IP-של אותו אתר (שרת). הרעיון של שמירת השמות של האתרים לעומת כתובות ה IP-שלהם מביא אתו ישר מחשבה על טבלה כלשהי (table-hash) שמקשר ישירות בין מספר לבין שם אתר, אך היות ויש המון שמות של אתרים קשה לשמור את כל השמות בשרת כלשהו שידע לתת את הסנכרון הזה, היות והגישה לטבלה כזו תתבצע אלפי פעמים בשנייה (בגלל שאנשים מסביב לכל העולם מנסים לגשת לאותו אתר באותו הרגע), וגם אם ניתן (וניתן) ליצור טבלה כזאת הגישה אליה של לקוחות מכל העולם תיצור עומס בשרת, ועיכובים בקבלת המידע גם בגלל המרחק הפיזי של אותו שרת לבין הלקוחות מסביב לעולם (רוחב פס של הלקוחות ועוד..). חיסרון נוסף הוא תחזוקת השרת הזה.

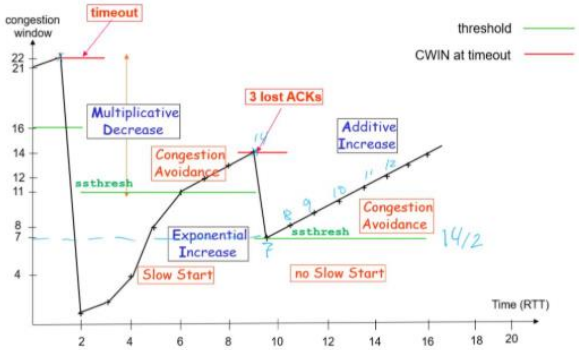


כך בעצם כאשר מנסים לגשת לאתר מסוים, ניגשים קודם ל DNS ROOT-והוא יודע להפנות את הבקשה לשרת ה DNS-המתאים. ברגע שגילינו שרת שמכיר את כל האתרים שמסתיימים ב. com-נגיד , אנחנו נפנה מעכשיו רק אליו (במקרה שנדרשת גישה לאתר שנגמר ב.) com-כדי למנוע פניות מיותרות ל, DNS ROOT-וכך גם לגבי שאר הסיומות. לרוב כל שרתי ה, DOMAIN LEVEL TOP-גם לא יודעים להכיל את כל האתרים הקיימים באותה הסיומת, לכן יוצרים מין מערכת היררכית דומה לזו הראשית, אשר מפצלת בין אפשרויות הסיומות השונות (לדוגמא: שרת ל, il.ac-שרת ל-il.co,שרת ל il.gov-וכו').

$$DevRTT = (1-\beta) * DevRTT + \beta * |SampleRTT - EstimatedRTT|$$
$$EstimatedRTT = (1-\alpha) * EstimatedRTT + \alpha * SampleRTT$$
$$TimeoutInterval = EstimatedRTT + 4 * DevRTT$$

ה- EstimatedRTT שמוכנס לתוך המשוואות, הוא ה-EstimatedRTT האחרון שנמדד





DNS: caching, updating records- כאשר נוספים domains, שרתים חדשים, נדרש לעדכן זאת בשרת ה DNS-הרלוונטי. כלומר צריך לגשת לשרת ה DNS-ולעדכן את המסד נתונים שלו (אחרי שהתקבל אישור לפתיחת הדומיין).כך גם כאשר אתרים נסגרים .לכל רשומה כזאת של כתובת ישנו תוקף, עד מתי הוא יהיה תקף וניתן יהיה לאחזר כתובת IP לשם האתר המסוים. הבעיה נוצרת כאשר משנים כתובת IP של אתר מסוים לפני סוף התוקף שלו, כך שהכתובת הזו שמורה בשרתי ה DNSהרלוונטיים, וכשיהיה ניסיון גישה לאתר, לא ניתן יהיה לאתר אותו. כדי לפתור את הבעיה הזו ישנו מנגנון אשר דואג לעדכן את כל מי שפנה לאתר הספציפי, בפנייה הבאה שלו, שהכתובת שונתה, בנוסף מתן הכתובת החדשה. על מנת ששרת אחד לא יצטרך לפנות לאלפי לקוחות שונים שניגשו אליו וקיבלו התחייבות על התוקף, השרת יפנה לשרתי DNS והם ידאגו לשנות איפה שצריך, לפי ההיררכיה של שרתי ה-DNS.

DNS records

TTL (Time To Live)- עד מתי התשובה שמתקבלת בתוקף. לדוגמא שרתים אשר צפויים להיות עמוסים ,מאזנים את העומסים בכך שהם מפנים את הלקוחות לשרתים אחרים של אותה הכתובת, וה TTLמגדיר לכמה זמן תקפה ההפניה הזו.

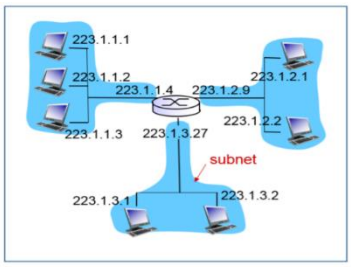
TYPE- סוג השאילתה/ תשובה.

TCP slow start- בתחילת התקשורת מול השרת נתחיל את שליחת המידע בקצב איטי (נגיד סגמנט אחד ל). RTT-ונגביר את קצב השליחה בשליחה הבאה (נגיד ל-2 סגמנטים) וכך הלאה מכפילים את כמות הסגמנטים בכל שלב. עם כל הכפלת כמות הסגמנטים עולה ניצולת הקו ,ברגע שנגיע לניצול של כ80-60 , % ה delay queue -יגדל יותר מידי. ולכן ייווצר עומס.

אסטרטגיית TAHOE- השליחה מתחילה לאט, סגמנט אחד ל-RTT,לאחר מכן 2 סגמנטים,וה-cwnd גדל אקספוננציאלית, ברגע שקצב השליחה יגיע לסף מסוים-threshold ,הוא יגדל לינארית. ברגע שנזהה עומס נחל שוב מסגמנט אחד וכן הלאה. קצב ה, threshold-נקבע עפ"י מחצית מהקצב שבו זוהה עומס בפעם הקודמת (כלומר אם בפעם הקודמת הגענו לקצב של 12 סגמנטים ל, RTT-אז בפעם הבאה, ה cwnd-יגדל אקספוננציאלית עד שנגיע ל-6 סגמנטים ל RTT-שזהו בעצם ה threshold -הנוכחי ומשם הוא יגדל ליניארית).

אסטרטגיית RENO- אסטרטגיה זו דומה ל, TAHOE-אך לאחר זיהוי העומס קצב השליחה לא ירד לסגמנט יחיד ל RTT-אלא ירד ל - thresholdהחדש (מחצית מקצב השליחה שבו זוהה עומס), ומשם קצב השליחה יעלה ליניארית. שיטה זו טובה במקרה שזוהה "עומס קל" ולכן אין צורך להוריד את קצב השליחה להתחלה. עומס "קל" הכוונה שקיבלנו 3 הודעות אישור על סגמנט מסויים (כלומר, אבד סגמנט בדרך או שהוא מתעכב). ולכן כדאי להוריד את קצב השליחה קצת. אבל אם זוהה עומס "כבד" ברשת אז ההתנהלות היא כמו של שיטת. TAHOE עומס "כבד" יזוהה ע"י כך שלא נשלחו הודעות אישור ACK– ברגע שהגיע הזמן -timeout.

TCP Throughput-הצד השולח מחזיק משתנה שנקרא – CWIN גודל המידע שהוא מוכן לשלוח בהתאם לעומס ברשת. כאשר נתקלים בעומס ברשת יורד גודל ה CWIN-בחצי ומשם עולה חזרה לגודל CWIN עד העומס בפעם הבאה (באופן גס). כלומר רוב קצב השליחה של מידע נע בין CWIN ל- (cwin/2) לכן בממוצע קצב השליחה הוא 3/4cwin ל-RTT.

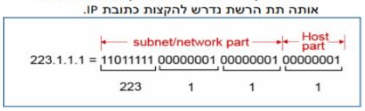


IP Addressing

Subnet part - הביטים הראשונים בכתובת ה-IP (MSB),מאפיינים את תת הרשת שבה נמצאות יחידות הקצה. (באיור 223.1.2.)

Host part – הביטים הנמוכים בכתובת ה-IP אשר מציינים את יחידות הקצה.

לא נהוג לתת סיומות של 111... או 000... ליחידות הקצה. הסיומת 000... בד"כ שמור למספר תת הרשת. כלומר ישנם 2^(32-k) כתובות שאפשר להקצות ליחידות הקצה בתת הרשת. כמובן שגם לראוטר של אותה תת הרשת נדרש להקצות כתובת IP.



IP Address structure

The network prefix: בכל תת רשת ישנם k ביטים קבועים שהם אופייניים לאותה תת רשת (כמו שניתן לראות באיור למעלה 223.1.3 – כלומר k=24 ביטים קבועים). ישנם שני דרכים לציין את מספר הביטים הקבועים לכל תת רשת:

- לכתוב את כתובת ה-IP סלאש מספר הביטים הקבועים : 223.1.3.0/26
- הדרך השנייה היא ע"י מסיכה : 255.255.255.192 = 1111 1111 1111 ... 11 1100 0000

שכבת הרשת (Network Layer)

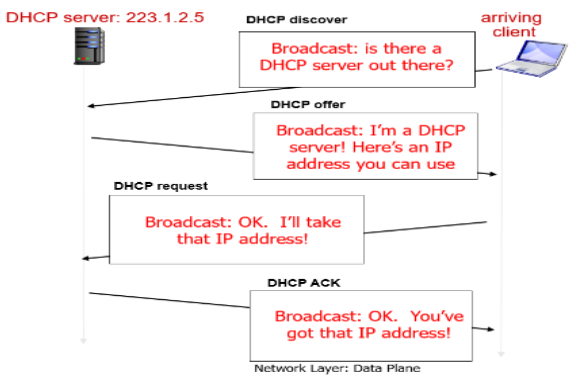
תפקידה להעביר הודעות מיחידה אחת לשנייה באמצעות שכבת הערוץ. שכבת הרשת, לעומת השכבות הקודמות (שכבת האפליקציה, שכבת התעבורה) – נמצאת גם על נתבים ברשת. בעצם שכבה זו אחראית על קיבעת המסלול מהמוצא אל היעד וגם להעביר פאקטים מכניסת הראוטר אל היציאה המתאימה לראוטר המתאים. קביעת הראוטר המתאים נעשית בד"כ בטבלאות ניתוב אשר מובנות בנתבים .עוד דרך לקביעת הנתב המתאים היא לבחור במסלול הזול/ הקצר ביותר. אבל הדרך הנכונה היא לדעת את המיפוי של הרשת הקרובה ובכך ישנה היכולת לחשב את המסלול האופטימלי בלי לבזבז זמן על "לשאול "ולקבל "תשובה" נתבים סמוכים.

כאשר מבקשים לייצג את הsubnet ב- Format dotted decimal notation :

הכוונה היא להשאיר את הביטים של הרשת דולקים, ואת כול שאר הביטים "החופשיים"

של התת רשת לאפס. **כתובת הפצה של תת רשת:**

הביטים של הרשת נשארים אותו דבר, והביטים החופשיים דולקים.



ה DHCP מחזיר את כתובת IP של המארח, כתובת IP של ה gateway, subnet בו נמצא המארח וכתובת ה LNS.
לצורך ה ARP , המארח בודק בעזרת ה subnet mask אם כתובת ה IP עליה הוא דורש MAC נמצאת באותו subnet, ז"א על אותו LAN. אם הוא לא על ה LAN, המארח יבקש את ה MAC של ה gateway.
מקבל את שם שרת הns, כתובת IP של הנתב, IP של המארח וכתובת הרשת המקומית,
בשימוש בDHCP, לא ניתן לקבל את כתובת הMAC של המארח

שכבת הערוץ (Link Layer)

Link Layer Services

בקרת זרימה(Flow Control)- ברמת הערוץ קצב העברת המידע מנוהל בין 2 יחידות סמוכות זו לזו ,כלומר יחידה אחת יכולה להודיעה לשכנה שהיא משדרת בקצב גבוהה מידי. ככל שמורידים את קצב השידור עולה הסיכוי לפיענוח נכון יותר של המסגרת עקב רעשים.
Error Detection- זיהוי שגיאות במסגרות דרך 2 יחידות סמוכות זו לזו.
Error Correction- תיקון שגיאות במסגרת ע"י מזהים של המסגרת, ללא צורך בשליחה חוזרת של אותה המסגרת.
Half Duplex-היכולת לשדר ולקבל אך לא בו זמנית, כלומר אם צד אחד משדר הוא לא יכול לקבל ולהיפך. (לדוגמא מ.ק 77 – מכשיר קשר צבאי). **Full Duplex-** היכולת לשדר ולקבל בו זמנית (לדוגמא הטלפונים).
Slotted ALOHA-אם 2 משתמשים או יותר שולחים בו זמנית הודעה ,תזוהה התנגשות. זמן השליחה מחולק לתאים slots – לפי זמנים בהם ניתן לשלוח פריים אחד. כל הודעה נשלחת למחשב ראשי. כאשר ישנה התנגשות המחשב הראשי מודיע על כך, ועל כל משתמש להגריל ערך שיקבע האם ב slot - המשתמש ישדר. יכול להיות מצב כך שאף משתמש לא ישדר והקו ישאר לא מנוצל. יכול גם להיות מצב ששנים ישדרו שוב ביחד ותהיה שוב התנגשות. וכך הלאה כל משתמש משדר עפ"י ערך "מוגרל" האם ישדר או לא. פרוטוקול זה עובד על הרעיון שאם יש לך משהו לשדר אז תשדר, אם תהיה התנגשות תקבל הודעה ואז תגריל ערך שיקבע האם ב slot -הבא תשדר או לא. אם לא הייתה התנגשות אז ממשיכים לשדר כרגיל עד להתנגשות. אחת ההנחות בפרוטוקול זה היא שהפריימים בגודל זהה. ניצול הערוץ המקסימלי בפרוטוקול זה הוא 37% .כלומר ב-63% מהזמן הערוץ יהיה מבוזבז. בשיטה זו לא נדרש מישהו שינהל את הערוץ.
Pure (unslotted) ALOHA-עפ"י פרוטוקול זה אין צורך במחשב מרכזי שינהל את ההודעות ויודיע על התנגשויות קיימות. החיסרון בזה הוא שהסיכוי להתנגשות (חוסר הסנכרון) גדל. כלומר כל משתמש ישדר ברגע שיש לו מה לשדר, מה שיגרור ירידה בניצול הערוץ פי 2 לכ- 18% .אם המשתמש לא קיבל אישור על ההודעה שלו, סימן שהייתה התנגשות והוא ישדר מחדש.
CSMA – carrier sense multiple access-פרוטוקול זה רעיונית דומה ל, ALOHA-אם יש מה לשדר אז תשדר, בשינוי אחד: לפני שאתה משדר תאזין לקו. אם מישהו כבר משדר אז תמתין עד שהוא יסיים כדי למנוע התנגשות ואז תשדר. אך עדיין יכולה להיות התנגשות ,במידה ו2 משתמשים ממתינים לקו פנוי, ברגע שהקו התפנה שניהם מתחילים לשדר ואז ישנה התנגשות .בגלל ה delay propagation-יש זמן שבו יחידה אחת התחילה לשדר והיחידה האחרת עדיין לא "שומעת" את השידור על הקו , ולכן היא בטוחה שהקו פנוי ומתחילה לשדר גם. לכן בזמן מסויים תהיה התנגשות בין הפריימים .בפרוטוקול זה היות וכל יחידה יכולה רק לשדר או להאזין ולא בו זמנית, שתי היחידות ימשיכו לשדר. כדי לוודא שלא הייתה התנגשות נמתין לאישור על הפריים, אם לא התקבל האישור סימן שהייתה התנגשות.
CSMA/CD – collision detection-בפרוטוקול זה ההנחה היא שכל יחידה יכולה גם להאזין וגם לשדר בו זמנית. במצב של התנגשות תוך כדי שידור היחידה תדע שיש עוד מישהו שמשדר במקביל. במצב כזה מקובל להוסיף, signal collision כך שכל מי שמאזין לקו ידע שהייתה התנגשות ולאחר מכן להפסיק לשדר. על מנת לזהות התנגשות תוך כדי שידור יש צורך לוודא שזמן השידור לא יהיה קטן מפעמיים ה . delay propagation-זאת מכיוון שאם 2 יחידות רחוקות אחת מהשנייה על אותו ערוץ במרחק מקסימלי, יחידה א' מתחילה שידור בזמן זאת-(propagation delay = (distance/velocity . ו t0=0, אומרת שבזמן tprop יחידה ב' תשמע שישנו שידור על הערוץ. לכן הזמן המקסימלי בו יחידה ב' יכולה להתחיל לשדר הוא שואף ל. tprop-ועד שיחידה א' תשמע שיחידה ב' שידרה יעבור זמן השווה ל- 2 . tprog+t0לכן נדרוש: 2t_propagation<t_transmission וכך נבטל את הצורך באישור על הודעות. אם זמן השידור(transmission)קטן יותר מפעמיים tprop לא ניתן לדעת אם הייתה התנגשות בשידור, היות ויחידה א 'הפסיקה לשדר כבר ולכן מבחינתה זה בסדר שיש עוד שידור על הקו. את המרחק וזמן ההתפשטות של האות ניתן לקבוע ולחשב עפ"י תקנים שונים (לרוב תקני), ETHERNET וכך ניתן לדעת את. tprop כמובן שעל זמן השידור ניתן לשלוט ע"י כך שמגדילים את גודל הפריים הנשלח בכל פעם. לאחר שזוהתה התנגשות נדרש לשדר את 2 הפריימים (או יותר), השידור מחדש נעשה רנדומלית כמו בפרוטוקול ALOHA ע"י הגרלת ערך שיגדיר רנדומלי כמה זמן על היחידה הזו לחכות לשידור חוזר.
MAC addresses and ARP-בשכבת הערוץ, לכל כרטיס רשת יש כתובת קבועה מהיצרן, כתובת זו היא כתובת (MAC אוEthernet ,physical , LAN) מספר זה הוא בן 48 ביט. לדוגמא. AD-09-76-BB-2F-IA :בניגוד לשכבת הרשת, בשכבת הערוץ ההודעה נשלחת ומתקבלת עפ"י כתובות. MAC כתובות ה MAC- רלוונטיות רק לערוץ בין 2 נתבים, כלומר הכתובת של השולחת והכתובת של הנתב המקבל . מערכת הARP (application resolution protocol) יודעת לתרגם מכתובות IP לכתובות. MAC המערכת הזאת צריכה להיות מאוד פשוטה היות והיא נמצאת על כרטיסי רשת שיכולים להיות מאוד פשוטים כמו בבובות ,אוזניות וכו'. וגם היא צריכה לדעת רק את כתובות ה MAC-של אותם המכשירים המחוברים ישירות באותה תת הרשת (טלפון – אוזניה וכו', מחשב – נתב וכו'). **הערה חשובה - MAC BROADCAST לא עובד מחוץ ל-LAN.**
Nat-זה אומר שכול הודעה שנשלחת "החוצה" מהתת רשת, נ שלחת כאילו המקור שלה הוא ה- IP של הראוטר, ולא המקורי.

ARP Messages

1. A (a, α) knows B's IP addr. (b) & wants to know B's MAC addr (β)
2. A sends ARP Query Message for B's MAC address:
   - message sent as broadcast frame on Ethernet
   - **Everyone on LAN learns A's MAC**

Src MAC	Dest MAC	Type	Source IP	Src MAC	Dest IP	Dest MAC
α	FF-…-FF	Query	a	α	b	?

Ethernet Header ← ARP Message →

3. B reads the message and sends ARP reply to A
   - reply sent as a unicast frame to A's MAC address
   - **Only A learns B's MAC**

Src MAC	Dest MAC	Type	Source IP	Src MAC	Dest IP	Dest MAC
β	α	Response	b	β	a	α

Ethernet Header ← ARP Message →

Collision avoidance

כשיחידה רוצה לשדר, היא תשלח הודעת RTS כדי לבדוק האם הקו נקי ואפשר לשדר, כל עוד לא התקבלה הודעת CTS היחידה לא רשאית לשדר. ברגע שהתקבלה הודעת CTS היחידה רשאית לשדר את ההודעה, בתקווה שבמקביל כל שאר היחידות שמעו את הודעת הCTS ולא ישדרו לאותו AP. כמובן שיכולה להיות התנגשות גם בשליחת הודעת RTS, אבל היות וזאת הודעה קטנה אין בעיה לשדר אותה שוב (גם כאן מגרילים ערך אשר יקבע האם היחידה תשלח הודעת RTS ומתי). כאשר המסגרת התקבלה בשלמותה ב-AP, ה-AP יוציא הודעת ACK כך שכל שאר היחידות ידעו שה-AP פנוי לקבל עוד מסגרת וגם זאת הודעת אישור שההודעה הגיעה תקינה.

Sending a Packet

- On same subnet:
  - Need MAC address of destination: **ARP**
- On some other subnet:
  - Need MAC address of first-hop router: **ARP**
- Need to tell whether destination is on same or other subnet?
  - Use the network mask: **DHCP**
- MAC addresses?
  - my own: hardcoded
  - others: ARP (given IP address)
  - ARP announcement: Hosts may send an ARP query on themselves, to announce a change in IP or MAC (no reply expected)
- IP addresses?
  - my own: DHCP
  - others: DNS (given domain name)
    - how do I bootstrap DNS communication? (DHCP)