

סיכון פרוטוקולי תקשורת

מבוא

מה זה רשת?

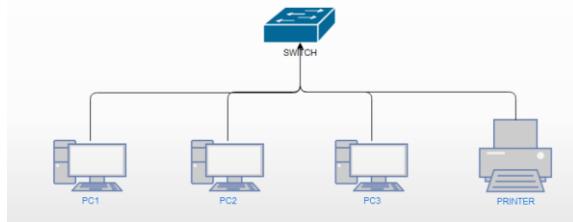
רשת היא דרך לחבר מכשירי קצה בלבד כדי שיכלו לתקשר ביניהם.

Message source -> Encoder+Transmitter -> Transmission Medium -> Reciever+Decoder -> Message Destination

LANs

.LAN = Local Area Network

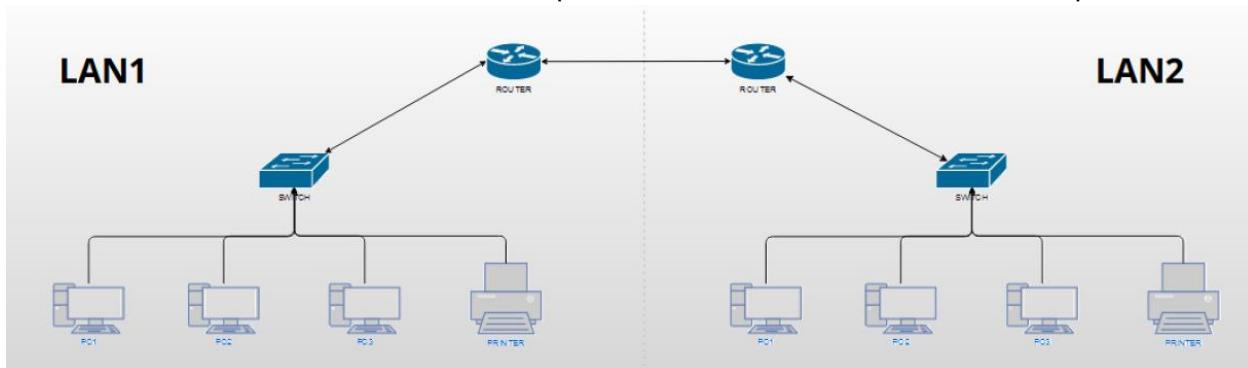
קיבוץ של מכשירי קצה לתוכה לוקלית אחת. LAN משתמש ב-switch כדי להעביר פקודות מממיכיר קצה אחד לאחר.



WANs

.WAN = Wide Area Networks

משמש לחיבור בין LANs. WAN משתמש בראוטרים על מנת לחבר בין ה-LANs.



מהו האינטרנט?

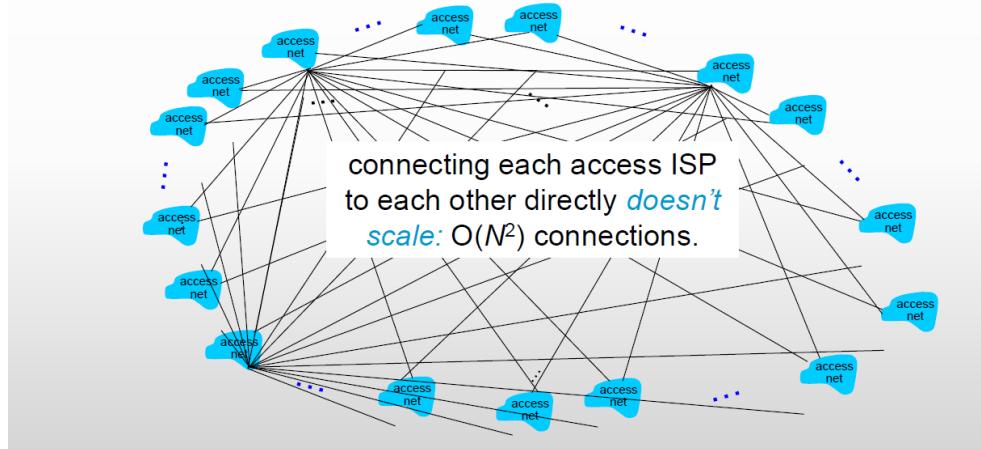
The internet = internetwork
חיבור של מיליארדי התקני מחשב מסביב לעולם.

האלמנטים לתקשורת:

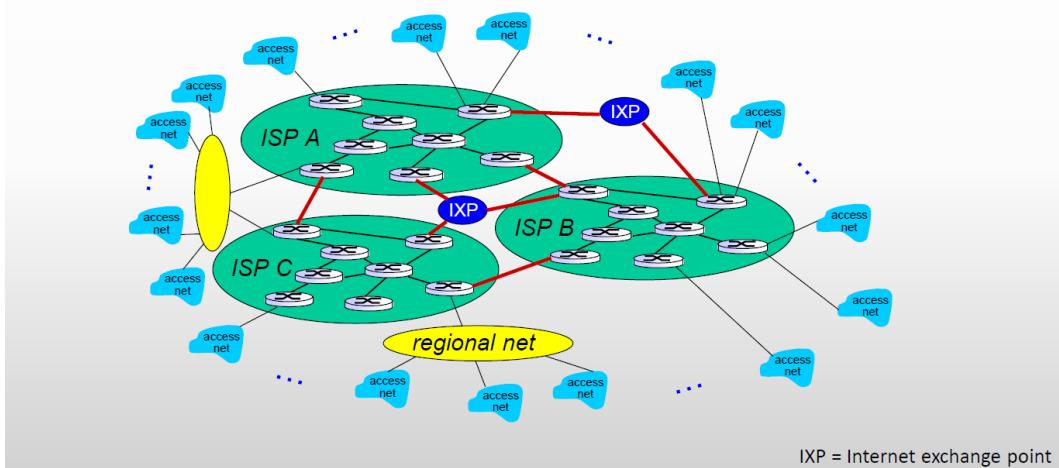
- Hosts (מערכות קצה).
- מכשירי תקשורת (סיבים, נוחשת, רדיו, לוויינים, מיקודול וכו').
- Packets – היחידה של המידע המועבר.
- פרוטוקול – הזרה בו המידע מאורגן ונשלח.

האינטרנט הוא סובלני לתקילות (fault-tolerant) – משתמש בחיבורים עודפים.
 Scalable – אפשר להרחיב ולהגדיל בקלות.
 האינטרנט הוא מבוזר – אין נקודת אחת של כשלון.
 Packet-switched network
 ניגשים באמצעות ISP – internet service provider

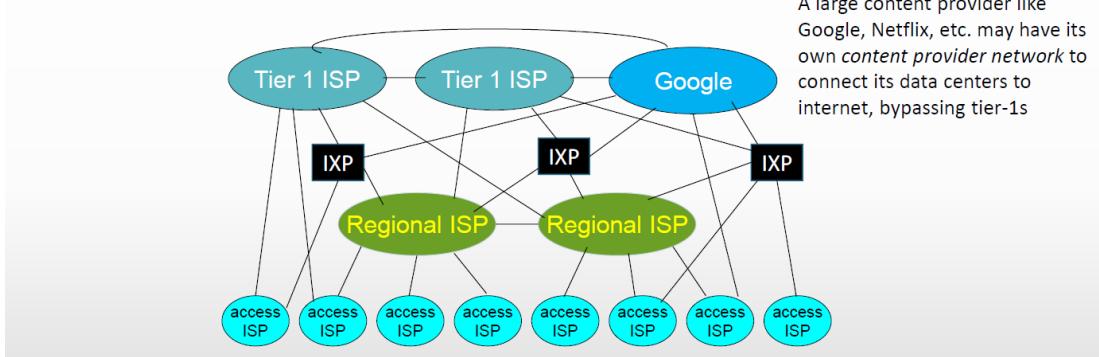
Accessing the Internet



Accessing the Internet



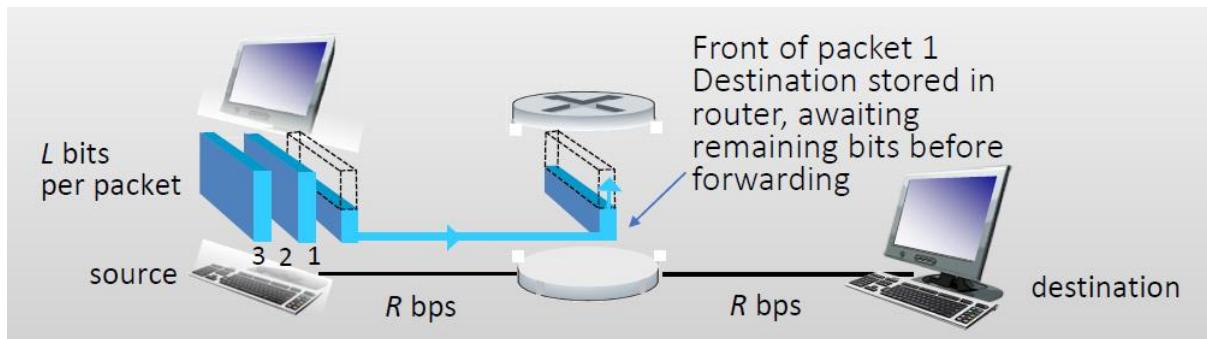
Accessing the Internet



Packet Switched Network

דרך יעה יותר לשימוש של אמצעי תקשורת אל מול מיתוג מעגלי (circuit switched). הودעה אחת מוחלקת לבlokים קטנים של>Data, כל בлок צזה נקרא פקטה. פקטות נעוט בעוצמות מהפקטות האחרות ואולי אפילו בדרכים שונות – עודפות (redundancy).

- רציפות של פקטות לא מובטחת, כי זה דורש buffering וניהול תורים.
- פקטות יכולות להיאבד בדרך (Packet loss).
- צריך לזכור מי השולח וממי מקבל (Addressing).
- משתמש בשיטת store-and-forward, כלומר כל פקטה צריכה להגיע לרואטר לפני שהיא יכולה להישלח לחיבור הבא.



Packet Switched vs. Circuit Switched

: Circuit Switched

- "קוויים" שמורות לשימוש של כל לקוח, אפילו אם הם לא בשימוש.
- מבזבץ משאבים ויקר.

: Packet Switched

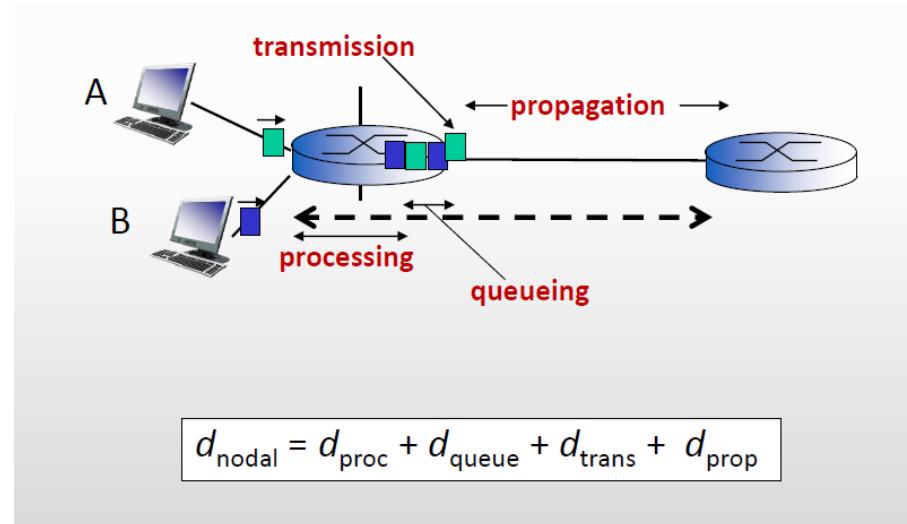
- "קוויים" משומשים במקביל (multiplexed), מאפשר שימוש יעיל יותר במשאבים.
- יותר משתמשים יכולים להתחבר.

נניח שיש 10 קוויים זמינים אבל כל משתמש פעיל רק 10% מהזמן. כמה משתמשים יכולים להתחבר אם החיבור הוא circuit switched?
תשובה: 10. הקווים שמורות מראש לכל לקוח, לא משנה כמה זמן הוא משתמש בהם.

נניח שיש רשתpacket switched עם 35 משתמשים, מה הסיכוי שייתר מ-10 משתמשים פעילים בו בזמן?

- n out of 35 users active = $\binom{35}{n} p^n (1-p)^{35-n}$ (binomial distribution)
- More than ten simultaneous users = $1 - \sum_0^{10} \binom{35}{n} p^n (1-p)^{35-n}$
- 0.00042

המקור ל-Packet Delay ברשת



דילוי כתוצאה מעיבוד (processing delay) :

- בדיקת בית שגיאות.
- קביעת החיבור הבא, ככלומר output link .(forwarding)
- בד"כ לוחות מ封חות מ-1ms.

דילוי כתוצאה מתורים (queueing delay) :

- פקודות כניסה לתורים ב-buffers של ראיות.
- מחכים זמן מסוים ב-link output לשילוח.
- תלוי בגודש התנועה.
- מתרחש כאשר הרבה משתמשים משדרים במקביל ביותר מרחב הפס שנitin לשדר בו.

Transmission Delay

$$d_{\text{trans}} = \frac{L}{R}$$

L := packet length (bits),
R := link bandwidth (bps)

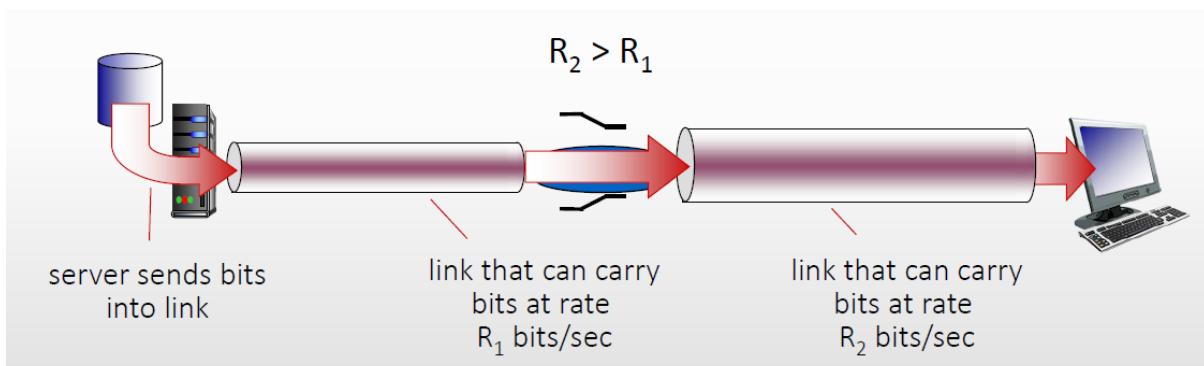
Propagation Delay

$$d_{\text{prop}} = \frac{d}{s}$$

d := length of physical link
s := propagation speed in medium
($\sim 2 \times 10^8$ m/sec for cable)

תפוקה (Throughput)

התפוקה של הקצב (bits/sec) בו ביטים מועברים בין שולח למקבל.
אם יש שני חיבורים שבהם הקצב הוא R1 ו-R2, כאשר R2>R1, התפוקה תמיד תהיה כגדל החיבור הקטן יותר, ככלומר R1



מה זה פרוטוקול?

פרוטוקול – דרך מסוימת לתקשורת. פרוטוקולים מגדירים את הפורמט, סדר ההודעות הנשלחות והמתקבלות, והפעולות הנדרשות להודעה נשלחת ומתקבלת. כל דרכי התקשורת באינטרנט משתמשים בפרוטוקולים.

דוגמה לפרוטוקול קניית ספר מארה"ב:

- Let us consider the example of buying a book from the USA



- Each level of distribution has its own container, protocol and medium

Level	Container	Protocol	Medium
Individual -> Local Post Office	Package	Give to local post office	By hand
Local Post Office -> Regional Distribution Center	Bag	Pack/unpack bag	By truck
Regional Distribution Center -> Regional Distribution Center	Air Freight Box	Pack/unpack container	By airplane

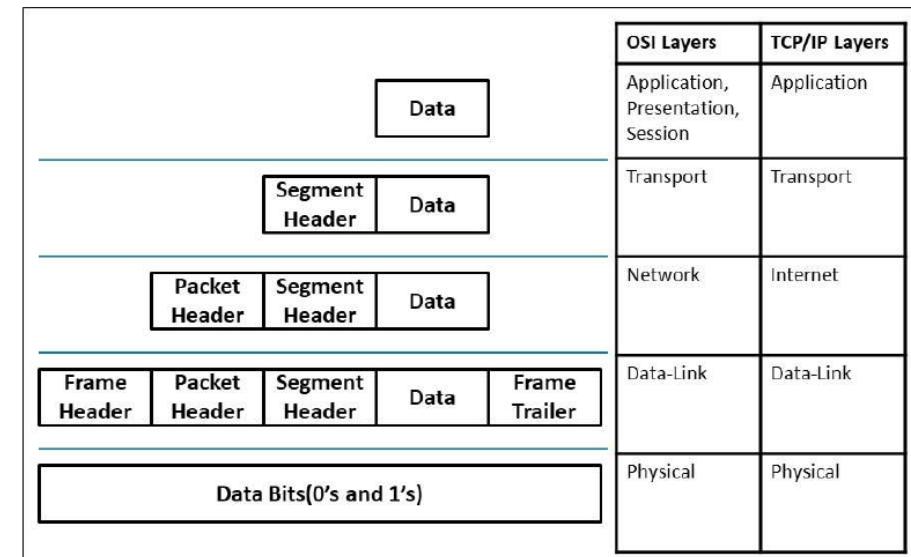
- The same is true of digital messages on the internet

Encapsulation & Decapsulation

בדוגמה הנ"ל נשים לב שבכל שכבה גבוהה יותר, "עוטפים" את ה-**container** ב-**container** גדול יותר לכל שלב.
 Package -> Bag -> Air Freight Box
 כשרוצים לחזור לשכבות נמוכות יותר, צריך לפתוח (.unwrap) בכל פעם Air Freight Box -> Bag -> Package

Wrapping == Encapsulation
 Unwrapping == Decapsulation

כל רואטר משתמש רק בכתובת היעד כדי לקבוע מה יהיה הצעד הבא (next hop), לא מערין אותו מה יש בתוך הפקטה. Encapsulation מומש ע"י הוספת header (או צירוף מידע נוסף לסוף הודעה) עם פקטה.



מה אנחנו רוצים מרשת תקשורת?

- אפשרות לתקשורת
- סדר נכון של הודעות
- ניתוב (Routing)
- ניהול שניים
- אימות קבלות (Verification of Receipt)
- aicot שירות
- בטחה
- וכו'

לכל אחד מהן"ל יש שלב לוגי שונה של אחריות. למשל אפשרות לתקשורת הקשור לחיבור הфизי בין שני מארחים, ניתוב הקשור לדרכים בין מארחים שונים בראשת, לעומת ניהול סשנים והצפנה שהן פעולות higher-level.

ישנם שני מודלים עיקריים לשכבות: IP/TCP (מודול 4 שכבות, יש יותר), OSI (מודול 7 שכבות, חדש יותר). כל שכבה יש פרוטוקולים מסוימים. שכבות נמוכות יותר "עוטפות" (Encapsulate) את המידע לשכבות הגבוהות יותר. כל שכבה יש תחומי אחריות מוגדרים. כל שכבה "חשובה" שהיא מחוברת לשכבה בקצתה השני. שכבות נמוכות אחראיות על המורכבות הגדלה (increasing complexity) של העברת המידע לקצתה الآخر.

- שכבת האפליקציה (Application Layer):
אחריות: מידע מהאפליקציה .(Application Data).
- **רֵק-ב-OSI** - שכבת הפרזנטציה (Presentation Layer):
אחריות: ייצוג מידע והצפנה .(Data representation and encryption)
- **רֵק-ב-OSI** - שכבת הסשן (Session Layer):
אחריות: תקשורת מארח פנימי .(Interhost communication)
- שכבת התעבורה (Transport Layer):
אחריות: חיבור קרצה לקצתה, אמינות .(End-to-end connections, reliability)
- **ב-OSI** – שכבת הרשת (Network Layer), **ב-TCP/IP** – שכבת האינטרנט (IP):
אחריות: קביעת מסלול ומיון כתובת לוגית (IP) .(Path determination and logical addressing)
- **ב-OSI** – שכבת קישור הדטה (Data Link Layer), **ב-TCP/IP** – שכבת הקישור (MAC):
אחריות: מיון כתובת פיזית (MAC) .(Physical addressing (MAC))
- **רֵק-ב-OSI** – שכבה פיזית (Physical Layer):
אחריות: מדיה, סיג널ים ושידור ביטים .(Media, signal and binary transmission)

מילוט מפתח (יחידת מידע=Protocol Data Unit, מכשיר=Device):

- שכבה האפליקציה:

- יחידת מידע:>Data).

- מכשיר: קוד האפליקציה (Application Code).

- שכבה התעבורה:

- יחידת מידע: מקטע/יחידת מידע (Segment / Datagram).

- מכשיר: דריבר הרשות של מערכת הפעלה (OS Network Driver).

- שכבה הרשות:

- יחידת מידע: חבילה (Packet).

- מכשיר: נתב (Router).

- שכבה הדאטה לינק:

- יחידת מידע: מסגרת (Frame).

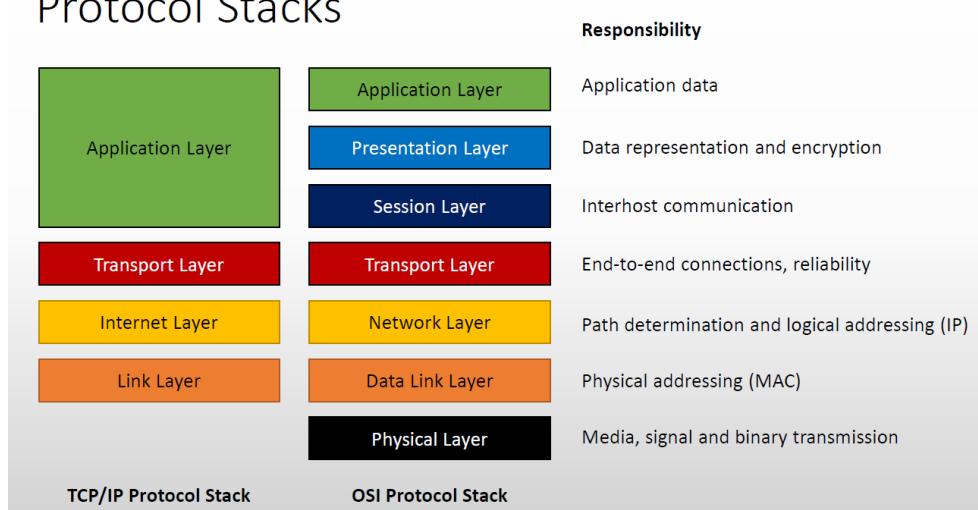
- מכשיר: סוויץ' הاب (Switch / Hub).

- השכבה הפיזית:

- יחידת מידע: ביטים (Bits).

- מכשיר: אין.

Protocol Stacks



Key Terms

	Protocol Data Unit	Device
Application Layer	Data	Application Code
Transport Layer	Segment / Datagram	OS Network Driver
Network Layer	Packet	Router
Data Link Layer	Frame	Switch / Hub
Physical Layer	Bits	

מיעון (Addressing)

למרות שאנו חושבים על שני מחשבים מתחברים ביניהם, במציאות התקשורת נעשית בצורה חשמלית דרך כרטיסי רשת. כל כרטיס רשת מתנהג כישות נפרדת בראשת ויש לו כתובת משלה. אם כל המחשבים בעולם היו נמצאים בתוך אוטו LAN, היינו יכולים להשתמש בכתובת אחת שצורובה בחומרה של כל כרטיס רשת – הכתובת הפיזית (Physical MAC Address). עם זאת, האינטרנט הוא רשת (web) מורכבת של LANs, כך שכרטיס רשת יכול להיות בכל מקום בעולם. לכן צריך איזורי דרך לתיאור הרשת או הכתובת הלוגית, כדי שייהי ניתן לנטרם תקשורת ביעילות מקורה לעד – Logical (IP) addressing.

כתובת פיזית מול לוגית (Logical vs. Physical Addresses)

כתובת פיזית משומשת בעיקר לתקשורת בתוך LAN (בגלל שבתוך LAN כל המחשבים יכולים "לראות" אחד את השני). כתובות לוגיות משומשות בעיקר לניטוב פקודות למחשבים שנמצאים ב-LANs שונים. באינטרנט (וברוב הרשומות) משתמשים בכתובת IP. הזרה של כתובת IP היא: XXX.XXX.XXX.XXX.

מיעון בהקשר של ה-Network Stack

בכל שכבה, נדרש איזורי דרך לזיהוי המארחים השונים שמתחברים. בשכבה האפליקציה, לכתובות יש משמעות ברמת **האפליקציה**, Application Level (למשל כתובת של אתר אינטרנט). בשכבה התעבורה, לכתובות יש משמעות ברמת **החיבור**, Connection Level (למשל סוקט Socket). בשכבה הרשת, לכתובות יש משמעות ברמה **לוגית**, Logical Level (כלומר באיזה חלק של הרשת המכשיר מזמין). בשכבה הדטה לינק, לכתובות יש **משמעות פיזית**, Physical meaning (כלומר איזה כרטיס רשת שולח ואיזה כרטיס רשת מקבל את המידע).

הסנפה (Sniffing)

כדי לדבג את המידע והפעולות ברשת ניתן להשתמש במסניף (Sniffer). מסניף רשת (Network sniffer) או תוכנה אשר מאזינה ומתעדת (records) את כל התקשורת האינטרנטית בכל השכבות.

שכבה 4 (TCP/IP / ISO) - שכבת האפליקציה

בשכבה זו מתרחש כל "העסק הלוגי". עיקרונו, המטרה של שכבה זו היא לאפשר קיום של אפליקציות. הלוגיקה של השכבה הלוגית ממומשת בקוד באפליקציה שרצ על המחשב או מכשור הקצה. יש הרבה פרוטוקולים בשכבה זו, כל אחד מהם "ארוז" (Encapsulated) לאחר מכן ניתן לקרוא (human-readable) את הפרוטוקולים בשכבה האפליקציה כדי לאפשר דיבוג (debug) קל.

מה לא נמצא בשכבה האפליקציה?

- מיעון פיזי/לוגי (Physical/logical addressing)
- ניתוב (Routing)
- בקרת חיבור/זרימה (Connection/flow control)
- מסירה מקצה לקצה (End-to-end delivery)

בזכות מודל השכבות, אין צורך להתחשב בכך כדי מאשר בונים אפליקציה.

שכבה 6 - שכבת הפרזנטציה ושכבה 5 - שכבת הסשן (Presentation and Session Layers)

במודול 7 השכבות (OSI), שכבות 6 ו-5 הן שכבות הפרזנטציה והסשן. השכבות ממומנשות בתוכנה של האפליקציה, לעיתים אפילו ע"י ספריות חיצונית.

פרזנטציה: מספקת מידע על תבניות נתונים (data format information) לאפליקציה, כולל הצפנה.
סשן: ניהול סשן בין המ משתמשים.

ארQUITקטורות בשכבה האפליקציה

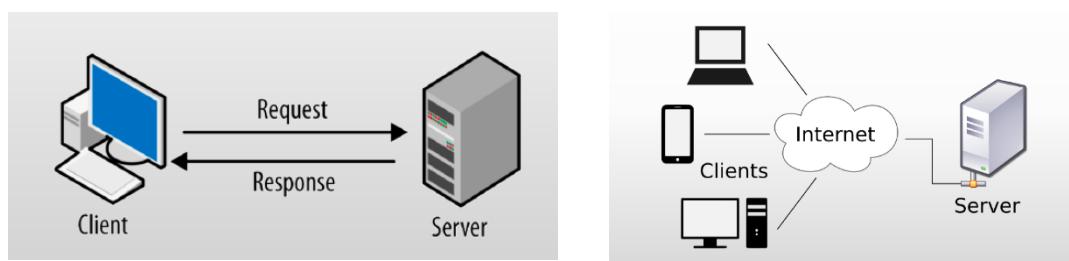
- Client / Server
- Peer to Peer (P2P)

ארQUITקטורת לקוח-שרת

קיימות שתי יישויות בארכיטקטורה זו:

- לקוח – מאמתים את הבקשה.
- שרת – עונה על הבקשה עם תשובה.

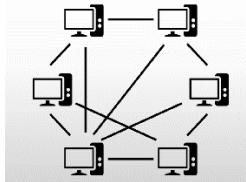
השרת תמיד פעיל וזמין, בד"כ במקומות קבועים. הלוקו יכול להשתנות, לשרת לא אכפת איפה הלוקו נמצא כי הלוקו הוא זה שמאתחל את התקשרות.
локוחות לא מתקשרים ישירות עם לקוחות אחרים.



ארכיטקטורת עמית לעמית (Peer-to-Peer)

אין הבדלה בין צמתים ברשת. כלם יכולים לתקשר גם בתור לקוח וגם בתור שרת. כל Peer מבקש שירות מ-peers אחרים והם מספקים שירותים כתגובה. לא חייבת להיות כטובת קבוצה.

דוגמאות לשימוש ב-P2P: .utorrent, file sharing, bitcoin



Sockets

סוקטים משומשים לחיבור בין תוכנות אפליקציה למכשורי הקצה ברשת. הם מנוהלים ע"י מערכת הפעלה, וסוקטים שייכים לתהליכים. הם הוגשר בין שכבות האפליקציה והתעבורה. אפליקציות "מרגישות" כאלו הן מתקשרות ישירות עם אפליקציות אחרות, אך בפועל סוקטים אחרים על התקשרות. סוקטים אחרים על כל תקשורת הרשות בשכבות הנמוכות.

סוקטים מזוהים ע"י מספר מסויים (הנמוכים יותר) בד"כ שמורות לשירותים מיוחדים, אך ניתן לשנות אותם אם רצים. סוקטים ידועים לדוגמה:

- HTTP: 80
- HTTPS: 443
- FTP: 21
- SSH: 22

כתובות של סוקטים באינטרנט: XXXXXX:XXX.XXX.XXX.XXX XXXX זה כתובות ה-IP ו-YY זה מספר הסוקט.

(Choosing the Transport Layer)

בהגדלת סוקט, אנו בוחרים איזה סוג של תעבורה אנחנו רוצים, כתלות בשירות המתבקש.

- **היגיון המידע (Data integrity):** אפליקציות מסוימות (למשל העברת קבצים, טרנזקציות אינטרנט וכו') דורשות 100% אמינות בהעברת המידע. אפליקציות אחרות (למשל אודיו) יכולות לסבול איבוד מידע מסוים.
- **תפוקה (Throughput):** אפליקציות מסוימות (למשל מולטימדיה) דורשות כמות מינימלית של תפוקה כדי להיות אפקטיביות. אפליקציות אחרות ("אפליקציות גמיישות") עושים שימוש בכל תפוקה שהן מקבלות.
- **זמן (Timing):** אפליקציות מסוימות (למשל טלפון נייד אינטרנט, משחקים אינטראקטיביים) דורשות דילוי נמור כדי להיות אפקטיביות.

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps	yes, 10's msec
text messaging	no loss	elastic	yes and no

(Transmission Control Protocol) TCP

- מונחה חיבורים (Connection-oriented): הגדרת חיבור ראשוני מחויבת להתקיים בין תהליכי הלקוח והשרת.
- אמינות (Reliable): אמין דו-כיווני (full-duplex). אמינות התüberה בין שליחת וקבלת התהילר.
- בקרת זרימה (Flow control): השולח לא "ייפתיע" (overwhelm) את מקבל.
- בקרת גודש (Congestion control): "חניקות" השולח כאשר הרשות עמוסה.
- לא מספק זמן (timing), תפוקה מינימלית מובטחת.
- מנוהל ע"י מכונת מצבים סופית (Finite State Machine).

(User Datagram Protocol) UDP

- העברת מידע לא אמינה בין תהליכי השולח וה מקבל.
- לא מספק הגדרת חיבור, אמינות, בקרת זרימה, בקרת גודשים, זמן או תפוקה מינימלית.
- מכיוון שאין UDP את התכונות שיש לTCP, הוא מספק פחת עיכובים (lower latency) והוא קל יותר למימוש.
- השימוש בפרוטוקול הוא בעיקר באפליקציות שאיבוד פקודות קטן (minor packet loss) הוא לא בעייתי.

פרוטוקולי אפליקציות ידועים והעברת המידע שלהם

application application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 5321]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 7230]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

(HyperText Transfer Protocol) HTTP

משומש להעברת מידע על דפי אינטרנט בין השרת לדפדפן. שיר למודל (ארQUITטורת) ללקוח-שרת.
.HTTPS = HTTP over TLS

- לכל בקשה (request) HTTP יש כמה מתודות אפשריות:
- GET – מתודה זו מבקשת ייצוג (representation) של המשאב המצוין (specified resource). בקשות GET יכולות רק להוציא (retrieve) מידע.
 - POST – מתודה זו מעבירה ישות (entity) למשאב המצוין, לרוב גורמת לשינוי במצב או לתופעות לוואי בשרת.
 - PUT – מתודה זו מחליפה את כל הייצוגים של משאב המטרה (target resource) עם המטען המבוקש (request payload).
 - יש עוד מתודות אפשריות שכמעט ולא משתמשים בהן.

לתשובות PPP יש קוד שגדיר את התשובה, ולפעמים מצורף גם תוכן לתשובה.

- | | | |
|-----------------------------|-----|---|
| (Informational – מידע) | 1XX | • |
| (Successful – הצלחה) | 2XX | • |
| (Redirection – הכוונה מחדש) | 3XX | • |
| (Client Error – שגיאת לקוח) | 4XX | • |
| (Server Error – שגיאת שרת) | 5XX | • |

קודים מוכרים:

- | | |
|----------------------------|---|
| OK :200 | • |
| Moved permanently :301 | • |
| Bad request :400 | • |
| Unauthorized :401 | • |
| Forbidden :403 | • |
| Page not found :404 | • |
| Internal server error :500 | • |
| Service unavailable :503 | • |

מיעון (Addressing) ב-HTTP: עמוד אינטרנט או מדיה הם רק אובייקטים בשרת, אך דרוש URL locator כדי לגשת לאובייקט.

.Host name, Path name: URL יש שני חלקים:

[למשל: !\[\]\(341b5bdc31177a6c7da7dc713da0d169_img.jpg\)](http://www.ruppin.ac.il/students/pic.jpg)

www.ruppin.ac.il :Host name

students/pic.jpg :Path name

Students, project managers

3.9. Example Message Exchange

The following example illustrates a typical HTTP/1.1 message exchange for a GET request ([Section 9.3.1](#)) on the URI "http://www.example.com/hello.txt":

Client request:

```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.64.1
Host: www.example.com
Accept-Language: en, mi
```

Request headers

Server response:

HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: text/plain

Response headers

Hello World! My content includes a trailing CRLF.

Source: RFC 9110 (the standard for HTTP)

HTTP הוא פרוטוקול שלא זכר מבנים (stateless), והוא לא יודע על בקשות קודמות. ניתן לשמר מבנים בעזרת טכנולוגיות אחרות כמו עוגיות (Cookies).

- לדוגמה לתקשרות HTTPS טיפוסית:

- השרת מקבל את החיבור.
- לקוחות שלוח הודעת GET HTTP לעמוד.
- השרת עונה תשובה עם קוד 200 (OK) ותוכן העמוד.
- החיבור נסגר.

HTTP עקי (Persistent): בחיבור HTTP לא עקי (HTTP/1.0) רק אובייקט אחד נשלח בכל חיבור TCP ואז החיבור נסגר. במקרה זה יותר מוקבץ אחד נדרש יותר מחיבור אחד. ב-HTTP/1.1, משתמשים בחיבור עקי על מנת לשЛОח כמה אובייקטים תחת חיבור TCP יחיד. התוצאה היא תוצאות מהירות יותר בזכות פחות "הוצאות" מיותרות (overhead).

No.	Time	Source	Destination	Protocol	Length	Info
+ 7214	11.513713	[REDACTED]	35.241.7.97	HTTP	343	GET / HTTP/1.1
+ 7218	11.598004	35.241.7.97	[REDACTED]	HTTP	493	HTTP/1.1 301 Moved Permanently (text/html)

```

> Frame 7214: 343 bytes on wire (2744 bits), 343 bytes captured (2744 bits) on interface \Device\NPF_
> Ethernet II, Src: [REDACTED], Dst: [REDACTED] (35.241.7.97)
> Internet Protocol Version 4, Src: [REDACTED], Dst: 35.241.7.97
> Transmission Control Protocol, Src Port: 65457, Dst Port: 80, Seq: 1, Ack: 1, Len: 289
  Hypertext Transfer Protocol
    > GET / HTTP/1.1\r\n
      Host: www.ruppin.ac.il\r\n
      User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0\r\n
      Accept: */*\r\n
      Accept-Language: en-US,en;q=0.5\r\n
      Accept-Encoding: gzip, deflate\r\n
      DNT: 1\r\n
      Connection: keep-alive\r\n
      Pragma: no-cache\r\n
      Cache-Control: no-cache\r\n
      \r\n
      [Full request URI: http://www.ruppin.ac.il/]
      [HTTP request 1/1]
      [Response in frame: 7218]
  
```

No.	Time	Source	Destination	Protocol	Length	Info
+ 7214	11.513713	[REDACTED]	35.241.7.97	HTTP	343	GET / HTTP/1.1
+ 7218	11.598004	35.241.7.97	[REDACTED]	HTTP	493	HTTP/1.1 301 Moved Permanently (text/html)

```

> Frame 7218: 493 bytes on wire (3944 bits), 493 bytes captured (3944 bits) on interface \Device\NPF_ id 0
> Ethernet II, Src: [REDACTED], Dst: [REDACTED]
> Internet Protocol Version 4, Src: 35.241.7.97, Dst: [REDACTED]
> Transmission Control Protocol, Src Port: 80, Dst Port: 65457, Seq: 1, Ack: 290, Len: 439
  Hypertext Transfer Protocol
    > HTTP/1.1 301 Moved Permanently\r\n
      Server: rhino-core-shield\r\n
      Date: Mon, 04 Jul 2022 12:40:48 GMT\r\n
      Content-Type: text/html\r\n
      Content-Length: 190\r\n
      Location: https://www.ruppin.ac.il/\r\n
      Via: 1.1 google\r\n
      Set-Cookie: [REDACTED] path=/; HttpOnly\r\n
      \r\n
      [HTTP response 1/1]
      [Time since request: 0.084291000 seconds]
      [Request in frame: 7214]
      [Request URI: http://www.ruppin.ac.il/]
      File Data: 190 bytes
  Line-based text data: text/html (7 lines)
    <html>\r\n
    <head><title>301 Moved Permanently</title></head>\r\n
    <body bgcolor="white">\r\n
    <center><h1>301 Moved Permanently</h1></center>\r\n
    <hr><center>rhino-core-shield</center>\r\n
    </body>\r\n
    </html>\r\n
  
```

למרות ש-HTTP בפורט (port) 80 עדין אפשרי, רוב השירותים כיום מבצעים הכוונה מחדש (redirect) לפורט 443 ומבצעים HTTPS over HTTP (HTTP over TLS) שהוא מוצפן.

בתוך ה"מערה" המוצפנת, מתודות HTTP בד"כ מוצגות כמו בדוגמה הבאה:

TCP	54 80 → 65457 [ACK] Seq=1 Ack=290 Win=66816 Len=0
HTTP	493 HTTP/1.1 301 Moved Permanently (text/html)
TLSv1.2	110 Application Data
TLSv1.2	213 Application Data
TLSv1.2	110 Application Data
TLSv1.2	213 Application Data
TCP	54 443 → 65361 [ACK] Seq=3449 Ack=1506 Win=178 Len=0
TLSv1.2	89 Application Data
TLSv1.2	89 Application Data

בגלל הצפנת ה-TLS, הרבה יותר קשה להסניף HTTPS. במקום זה, כל יותר להשתמש בדף ע"י לחיצה על F12 ולחפות מיידע תחת Network.

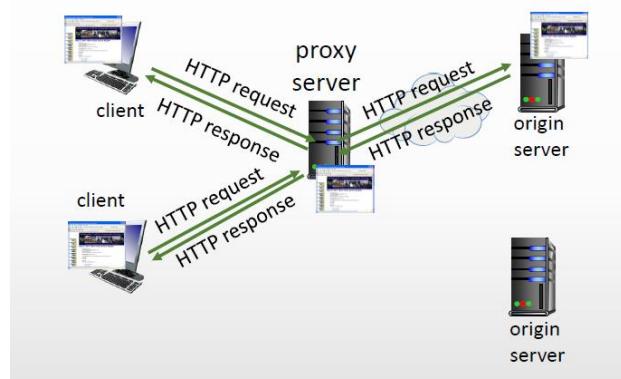
מטמון (Caching)

רוב האובייקטים לא משתנים לעיתים קרובות (למשל תמונות, טקסטים וכו'). בשימוש במתמונן, אין צורך לחתוך (fetch) אותו בכל פעם שxabרים לאתר, וזה מאייך את הגישה לאתרם. המידע הרלוונטי נשמר במתמון בדף.

Proxies

Proxy – שירות שמבצע את הבקשות בשבייל הלקוחות. תפקידו העיקרי לספק גישה למשתמשים חיצוניים באינטרנט. הסואאת כתובת ה-IP מטאפרה ע"י חיבור לשרת proxy דרך שדרכו עוברת התעבורה. לעיל בשבייל:

- מטמון, מאייך את זמן הגישה
- אבטחה
- אימוטים (Authentication) / הגבלות משתמשים (user management restriction) וכו'
- בקרת הורים (Parental controls)
- אNONIMITY



אימייל (Email)

בإيمיל קיימים שלושה פרוטוקולים עיקריים:

- Simple Mail Transfer Protocol – SMTP
- Post Office Protocol 3 – POP3
- Internet Message Access Protocol – IMAP

SMTP

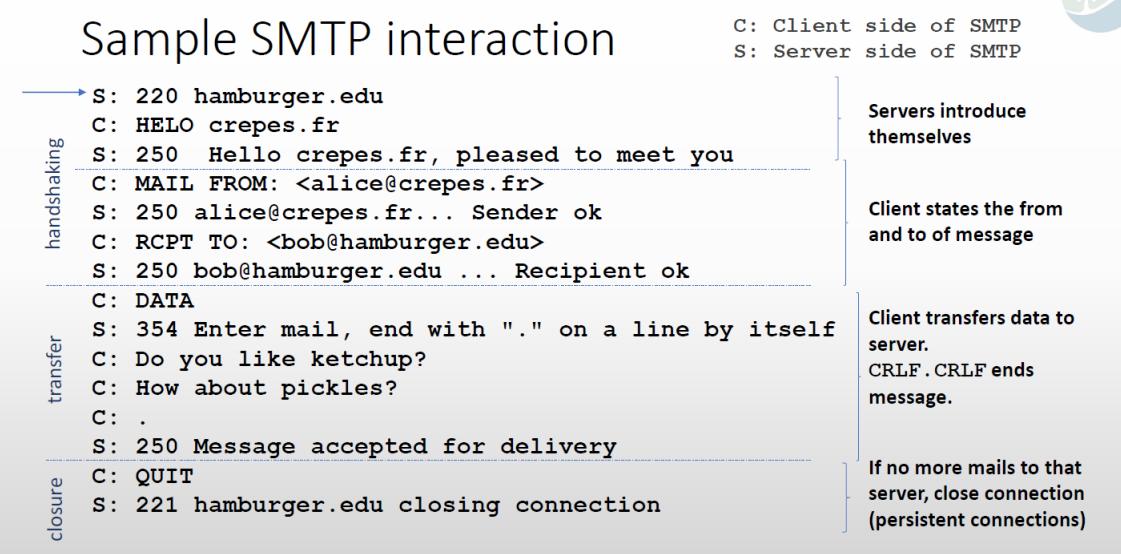
משומש לשיחת הדעות, בד"כ בפורט 25.

קיימים לו שלושה שלבים:

- Handshaking (greeting)
- העברת ההודעות
- סגירה (Closure)

הפקודות כתובות בטיקסט כך שנitin לקרוא אותן. התגובה מגיעה עם קוד סטטי ובטיקסט שניtin לקרוא, בדומה ל-HTTP.

Sample SMTP interaction



Mail message format

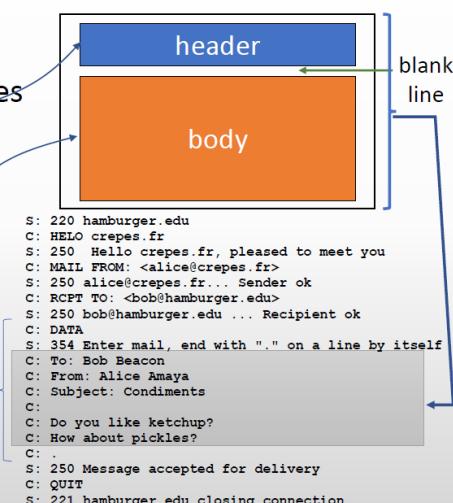
SMTP: protocol for exchanging email messages

RFC 822: standard for text message format:

- Header lines, e.g.,

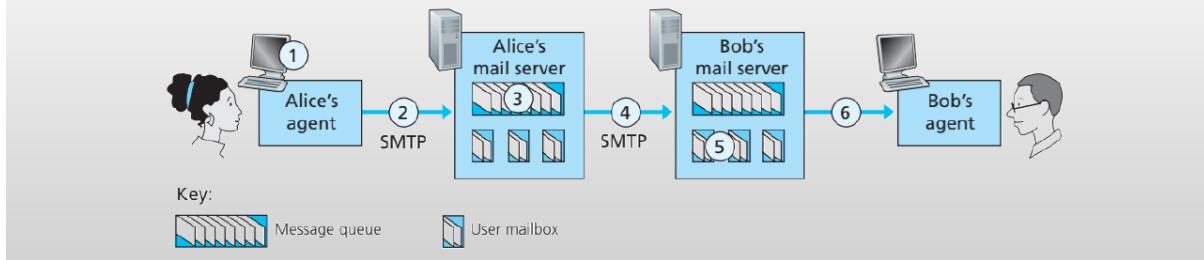
- To:
- From:
- Subject:

- Body: the “message”

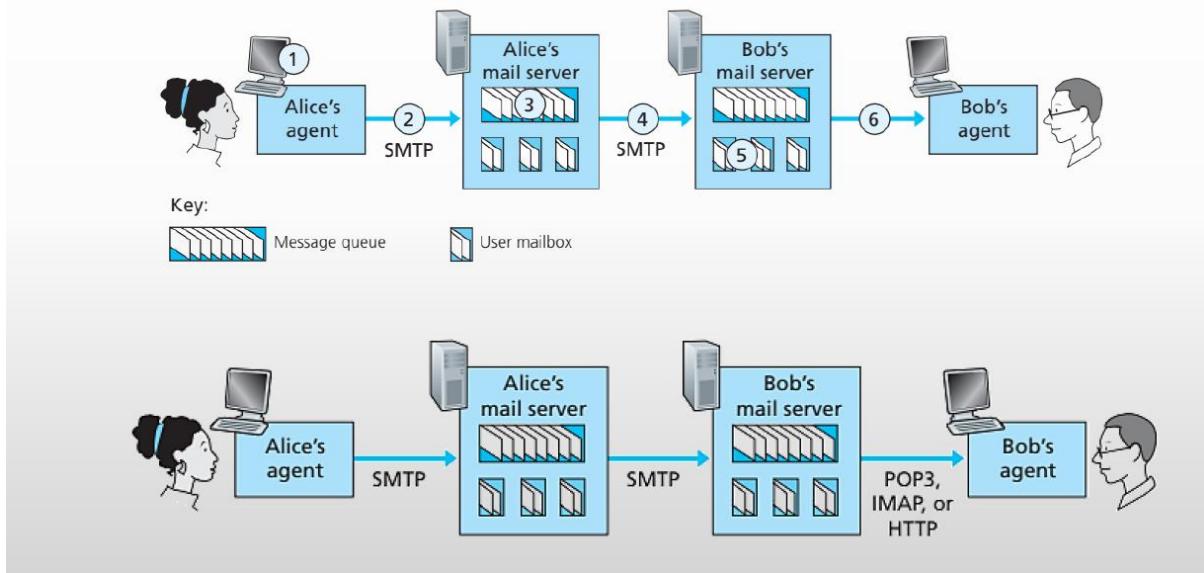


Alice Sends a Message to Bob

1. Alice uses a User Agent (UA) to compose message "to" bob@someschool.edu
2. Alice's UA opens TCP connection to her mail server and sends message to her mail server using SMTP; message placed in message queue
3. Client side of SMTP (mail server A) opens TCP connection with Bob's mail server B
4. SMTP client (mail server A) sends Alice's message over the TCP connection
5. Bob's mail server B places the message in Bob's mailbox
6. Bob invokes his UA to read message at his convenience (asynchronous)



Alice Sends a Message to Bob



קבלת מייל

SMTP משמש לשילוח מייל. כדי לקבל מייל, משתמשים ב-POP3 או IMAP. POP3 הוא פרוטוקול פשוט לקלות מייל: ברגע שלקוח מוריד את המייל, המייל נמחק מהשרת. כל ה"לוגיקה" מתרכחת אצל הלקוח. IMAP לעומת זאת מאפשר לשמור את ההודעות בשרת. ישנן תיקיות מסוימות בשרת לכך.

C: Client side of POP3
S: Server side of POP3

POP3 protocol

Simple, yet limited protocol

User Agent opens TCP connection to mail server
port 110.

Authorization phase

client commands:

user: declare username
pass: password

server responses

+OK
-ERR

Transaction phase, client:

list: list message numbers
retr: retrieve message by number
dele: delete
quit
After quit, server deletes messages marked for deletion

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498 List of messages and their sizes
S: 2 912 from older to newer
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 2 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

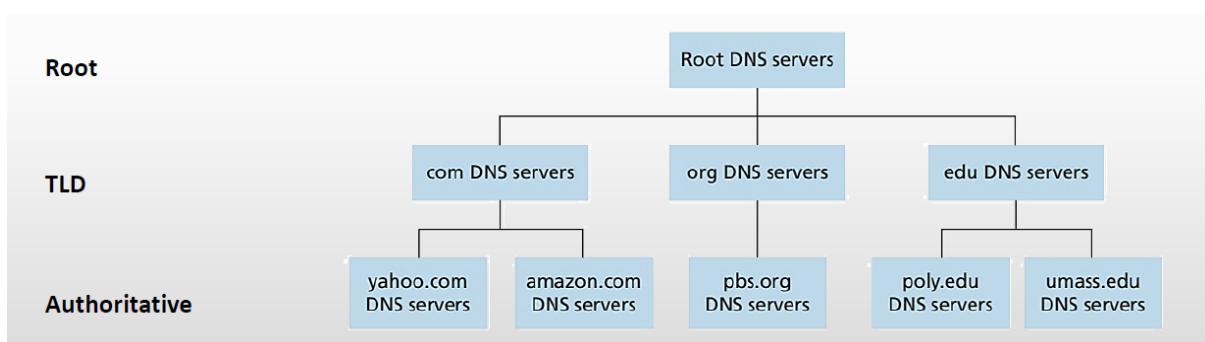
מה הוא Webmail

כדי לגשת למיילים שלנו, אנו משתמשים ב-webmail client. נכנסים ל-webmail בדף ה-הדף, כמו אתר אינטרנט רגיל. הוא משתמש בתוכנות צד שרת כדי לתקשר עם שרת המail, ובין הלקוח ל-webmail client אין שום שימוש בプロトコル מייל. ה-webmail מציע פיצרים רבים, כתלות בprotocokolים משתמשים בהם.

(Domain Name System) DNS

כדי לזרות לקוות ושרותים האינטרנט משתמש בכתובות לוגיות המבוססות על כתובות IP. כתובות IP הן לא נוחות לשימוש המשתמשים. לכן משתמשים בDNS לתרגום שמות לכתובות IP.

DNS היא מערכת היררכית שmaps שמות אתרים (domain) לכתובות IP. המערכת מבזורת כדי למנוע נזודה אחת של כישלון (single point of failure) ובכך לספק שירות מהיר.



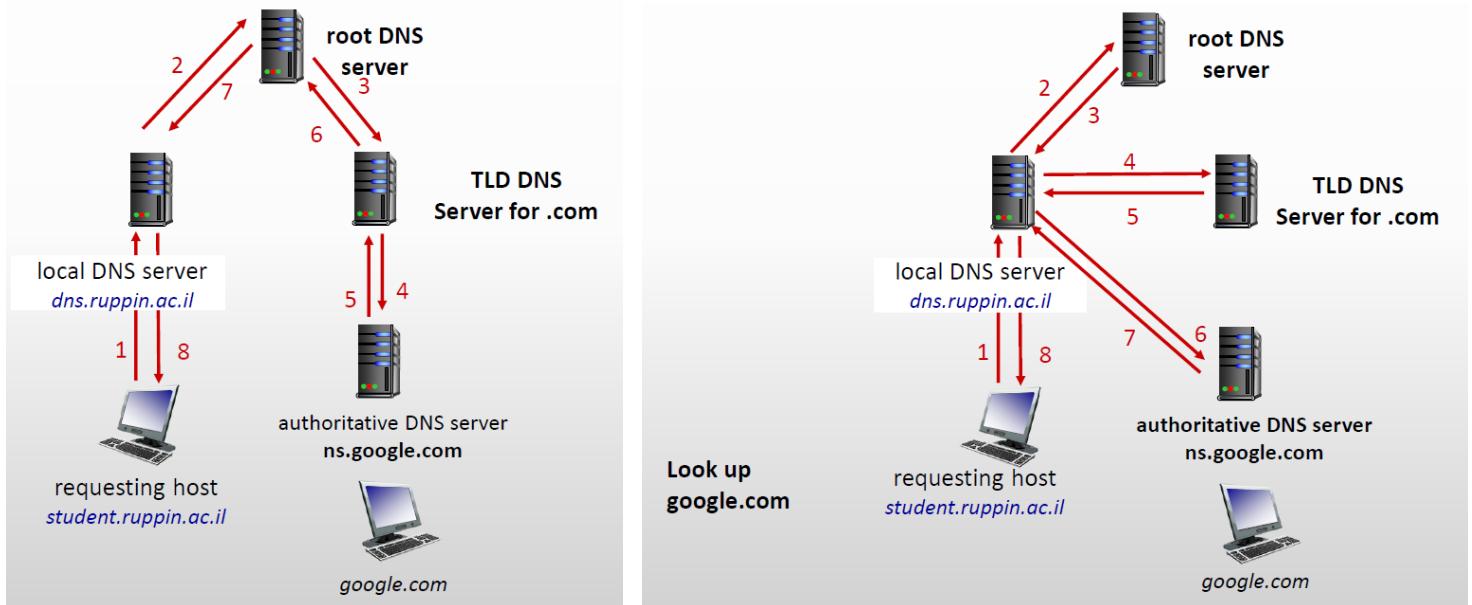
- DNS משתמש בשלוש רמות של שרתים שמות:
- אחראי על הרמה הגבוהה של שמות אתרים בכל האינטרנט. Root Name servers
 - אחראיים על סיווגם של תLD (TLD) – Top-level domain servers
 - אחראיים על המיפוי האמיטי והמהימן מ-IP, בד"כ בرمת הORGן כמו ruppin.ac.il.

Root Name Servers: קיימים בסה"כ 13 שרתים root DNS בכל העולם, עם השמות M-A. כל שרת הוא בעצם רשות של שרתים משוכפלים בשבייל אבטחה ואמינות.

Root Server	IPv4 address	Operator	Country
A	198.41.0.4	Verisign	USA
B	199.9.14.201	USC- ISI	USA
C	192.33.4.12	Cogent Communications	USA
D	199.7.91.13	University of Maryland	USA
E	192.203.230.10	NASA Ames Research Center	USA
F	192.5.5.241	Internet Systems Consortium	USA
G	192.112.36.4	Defense Information Systems Agency	USA
H	198.97.190.53	U.S. Army Research Lab	USA
I	192.36.148.17	Netnod	Sweden
J	192.58.128.30	Verisign	USA
K	193.0.14.129	RIPE NCC	Netherlands
L	199.7.83.42	ICANN	USA
M	202.12.27.33	WIDE Project	Japan

איך עובדת שאלות DNS:
יש שתי צורות לשאלות DNS:

- איטרטיבית – כל שרת מתשאול מספק את הכתובת לשרת הבא שיש לשאול אותו.
- רקורסיבית – ה"עומס" נופל על השרת הראשון שsspאל בשבייל שם.



DNS Records הם תיעודים מבוסיס הנתונים של שרת DNS, והם אלו שמופיעים כתובות URL ל-IP.

הם שמוראים בפורמט שנקרא RR אשר מכיל:

- שם האתר שמחפש.
- Name – שם האתר שמחפש.

- – הערך המתאים כתשובה.
- – סוג התיעוד.
- TTL – כל כמה זמן שרת DNS יעשה refresh לтиיעוד.

סוגי התיעודים:

- A – מחזק את כתובת ה-IP של הדומיין המבוקש.
- NS – מעביר דומיין אחד או תת-דומיין לדומיין אחר. לא מספק כתובת IP.
- CNAME – מחזק את שם שרת DNS שמוסמך (authoritative) לאותו דומיין. במילים אחרות, נותן את שרת DNS שמננו אפשרות להוציא את כתובת ה-IP של הדומיין המבוקש. לא מספק כתובת IP.
- MX – מכון מייל לשרת המייל. לא מספק כתובת IP.

DNS Records

RR format: (name, value, type, ttl)

type=A

- **name** is hostname
- **value** is IP address
- (relay1.bar.foo.com, 145.37.93.126, A,-)

type=CNAME

- **name** is alias name for some “canonical” (the real) name
- **value** is canonical name
- (foo.com, relay1.bar.foo.com, CNAME,-)

type=NS

- **name** is domain (e.g., foo.com)
- **value** is hostname of **authoritative** name server for this domain
- (foo.com, dns.foo.com, NS,-)

type=MX

- **value** is name of mail-server associated with **name**
- (foo.com, mail.bar.foo.com, MX,-)

- *Query and reply messages, both with same message format*

Message header

- identification: 16 bit # for query, reply to query uses same #
- flags:
 - query or reply
 - recursion desired
 - recursion available
 - reply is authoritative

identification	flags	
# questions	# answer RRs	
questions (variable # of questions)		
answers (variable # of RRs)		
authority (variable # of RRs)		
additional info (variable # of RRs)		

name, type fields
for a query

RRs in response
to query

records for
authoritative servers

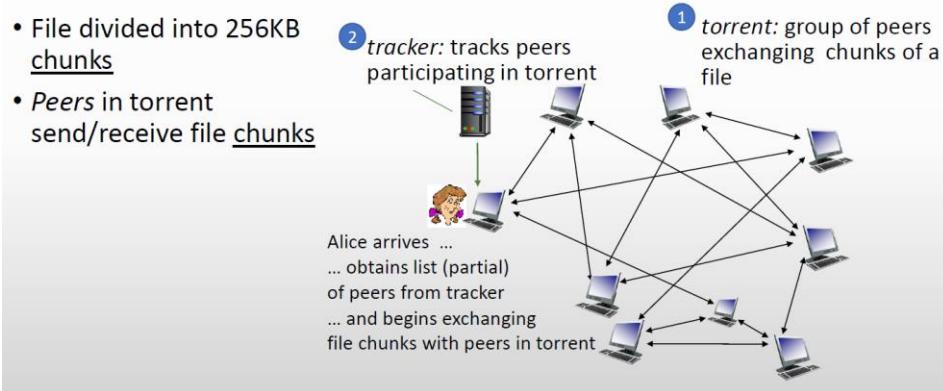
additional “helpful”
info that may be used

identification	flags	
# questions	# answer RRs	
# authority RRs	# additional RRs	
questions (variable # of questions)		
answers (variable # of RRs)		
authority (variable # of RRs)		
additional info (variable # of RRs)		

אין שרת פתוח כל הזמן. מערכות קבוצה שירוטיות מתקשרות ישירות ביניהן. Peers, שנשלטים ע"י משתמשים, לסייעון מתחברים ומשנים כתובות IP.

:BitTorrent – P2P

קובץ מחולק לחטיכות של 256KB. ה-peers בטורנט שולחים/מקבלים חתיכות של קבצים. קיימ tracker שודע למזהו peers משתפים בטורנט שיש להם את החטיכות שצורך. כאשר משתמש מתחבר לטורנט ללא חטיכות בהתחלה, אך הוא יאגור אותם עם הזמן מ-peers אחרים), הוא מתחבר ל-tracker שנותן לו רישיונה של peers להם יש את החטיכות, ומתחבר לתת קבוצה של peers ("שכנים"). בזמן ההורדה, ה-peer גם מעלה חטיכות לא-peers אחרים. Peer יכול להחליף את-peers שאויתם הוא מחליף חטיכות. Peers יכולים לבוא ולլכת ("נטישה"). ברגע של-peer יש את הקובץ השלם, הוא יכול לעזוב או להישאר בטורנט.



- File divided into 256KB chunks
 - Peers in torrent send/receive file chunks
- Peer (Alice) joining torrent:
 - has no chunks, but will accumulate them over time from other peers
 - registers with tracker to get list of peers, connects to subset of peers ("neighbors")
 - While downloading, peer uploads chunks to other peers
 - Peer may change the peers with whom it exchanges chunks
 - Churn (נטישה): peers may come and go
 - once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

BitTorrent: requesting, sending file chunks

Requesting chunks:

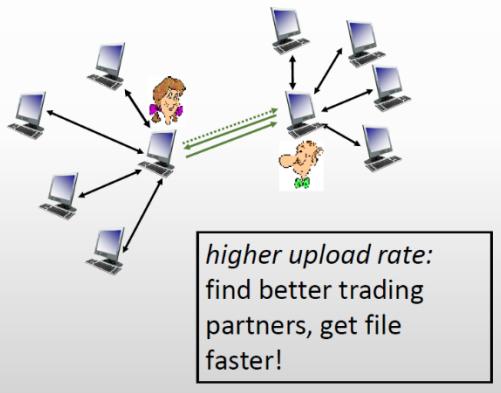
- At any given time, different peers have different subsets of file chunks
- Periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

Sending chunks: tit-for-tat

- Alice sends chunks to those four peers currently sending her chunks at highest rate
- Other peers are choked by Alice (do not receive chunks from her)
- Re-evaluate top 4 every 10 secs
- Every 30 secs: randomly select another peer, starts sending chunks
 - "Optimistically unchoke" this peer
- Newly chosen peer may join top 4

BitTorrent: tit-for-tat

1. Alice “optimistically unchoke” Bob (after 30 secs)
2. Alice becomes one of Bob’s top-four providers; Bob reciprocates (10sec)
3. Bob becomes one of Alice’s top-four providers



בקשת חתיכות: בכל זמן נתון, ל-peers שונים יש חלק סטים שונים של חתיכות קובץ. באופן מוחזק, ה-peer מבקש משאר ה-peers רשותה של החתיכות שברשותם. מבקשים בכל פעם את החתיכות החסורות מה-peers, כאשר קודם מבקשים את החתיכות הנדרות יותר.

שליחת חתיכות (tit-for-tat): ה-peer שולח חתיכות ל-4 peers אחרים שכרגע שלוחים לו חתיכות בקצב הגבהה ביתר שניתן. ה-peers האחרים ”ננקים“ מאותו peer (לא מקבלים חתיכות ממנו). בכל 10 שניות בוחרים מחדש 4 peers שייהיו בתופ. בכל 30 שניות בוחרים באופן רנדומלי peer אחר ומתחילה לשולח לו חתיכות (בתגובה לא ”לchnerוק“ אותו). ה-peer החדש שנבחר יכול להציג ל-4 שפטופ.

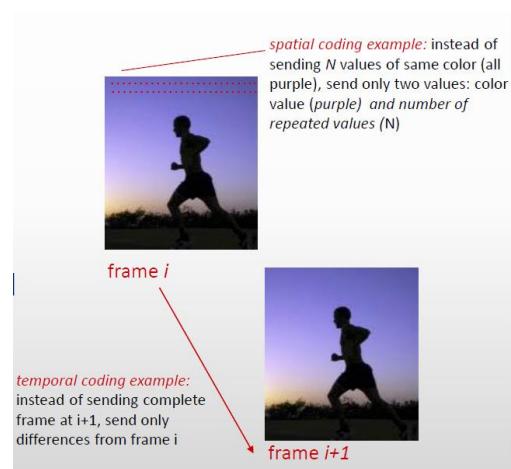
שידור וידאו ו-CDNs: context

רוב רוחב הפס באינטרנט נקבע בגלל שידורי וידאו. האתגרים:

- קנה מידת – איך ניתן להגיע ל-1B~ משתמשים?
 - רבגוניות (heterogeneity) – למשתמשים שונים יש יכולות (capabilities) שונות. למשל חיבור חוטי מול אלחוטי, רוחב פס צר מול רחוב וכו'.
- הפתרון: תשתיית מוצזרת ברמת האפליקציה.

מולטימדיה – וידאו:

- VIDEO – רצף תמונות שמצווגת בקצב קבוע.
 תמונה דיגיטלית – מערך של פיקסלים, כל פיקסל מיוצג ע”י ביטים.
 קוד – משתמש בשאריות בתונר ובין התמונות כדי להעלות את מספר הביטים המשמשים לקידוד תמונה.
- Spatial – ב透ctor תמונה.
 - Temporal – מתמונה אחת לאחרת, ככלומר ביןיה.



- קידוד וידאו בקצב קבוע. (constant bit rate) CBR
- קידוד וידאו בקצב משתנה כתלות בקידוד spatial, temporal VBR

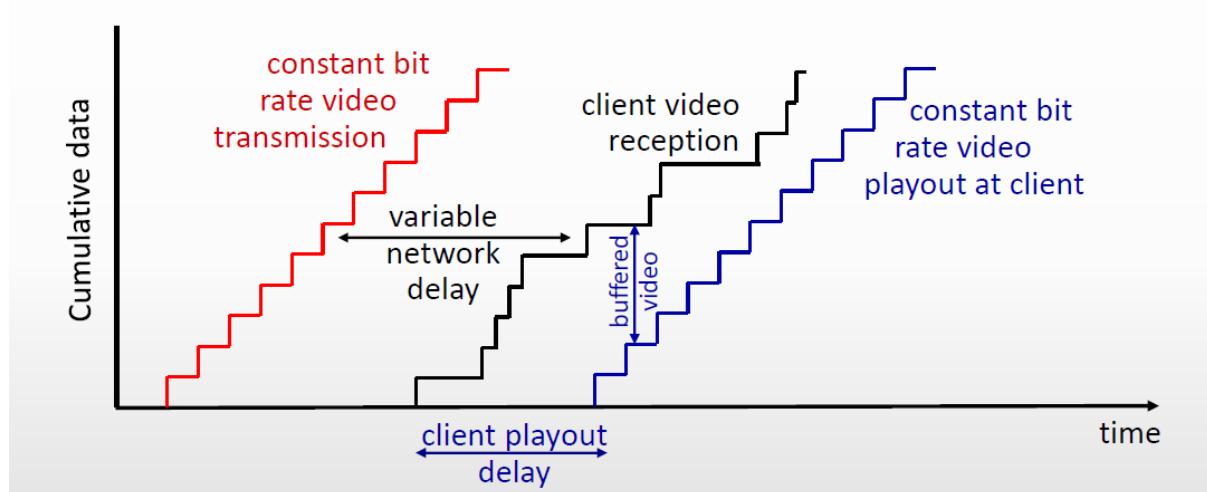
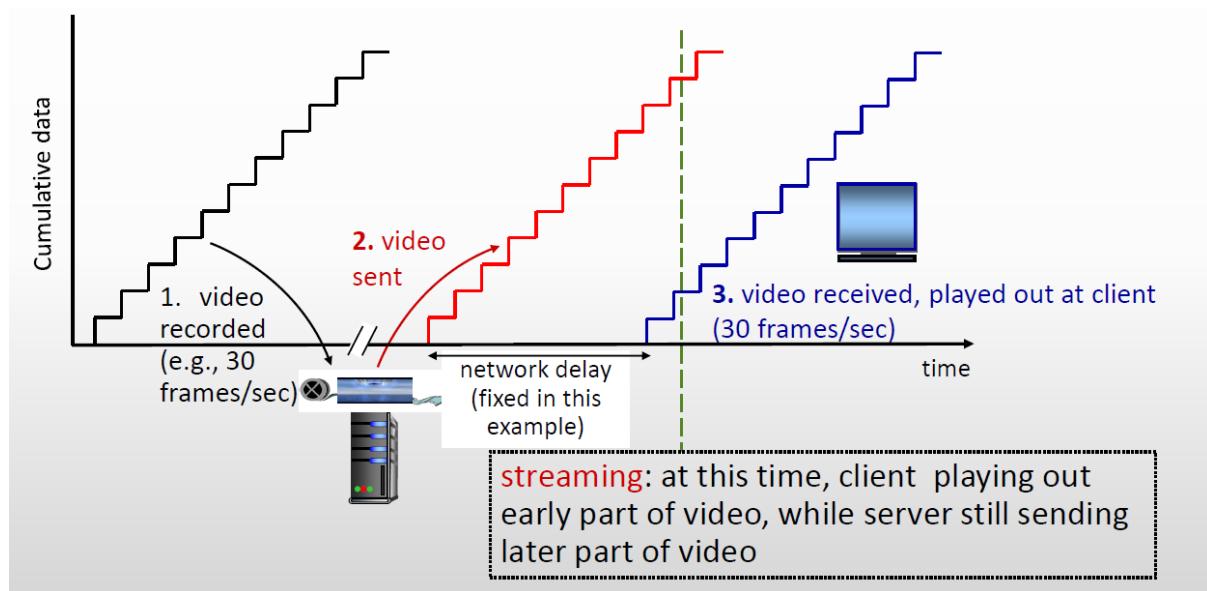
שידור וידאו שמור בשרת:

במצב הפשוט, הוידאו השמור בשרת עבר באינטרנט ללוקו.

האתגרים העיקריים:

- התפקיד של שרת-ללוקו יכול להשתנות עם הזמן, עם גודשי רשת משתנים (בבית, גישה לאינטרנט, מרכז אינטרנט, שרת וידאו).
- Packet loss (דילוי) בഗל עומסים וגודשים יגרמו לדילוי בשידור, או לאיכות וידאו ירודה.

Buffering בצד הלוקו ודיילי בהפעלת הוידאו מפסיק על דiley בשרת.



שכבה 3 (IP / 4 (ISO) - שכבה התעבורה

שכבה זו אחראית על העברת המידע מאפליקציה אחת לאחרת. כולמר אחראית על תקשורת לוגית בין תהליכיים. שליטה באמיניות של חיבור מסוים ע"י בקרת זרימה, חלוקה (segmentation) ובקרת שגיאות. שולחת מחדשSEGMENTים שלא הגיעו לעדמ.

מה לא נמצא בשכבה זו?

- מיעון לוגי/פיזי (Physical/logical addressing)
- נתות (Routing)

ישנם שני פרוטוקולים חשובים בשכבה התעבורה: TCP, UDP.

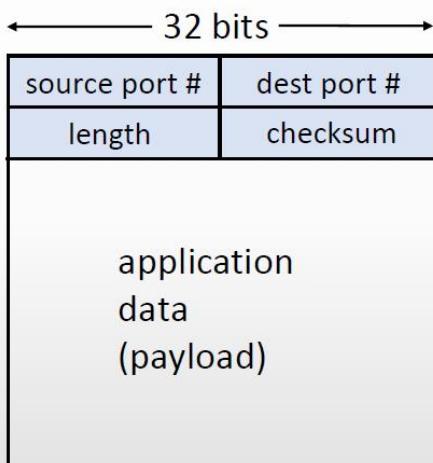
Multiplexing / Demultiplexing

בעזרת סוקטים, לכל מארח יכולם להיות הרבה חיבורים ברמת שכבה התעבורה.
סוקט של TCP מאופיין ע"ז:

- Source IP
- Source Port
- Destination IP
- Destination Port

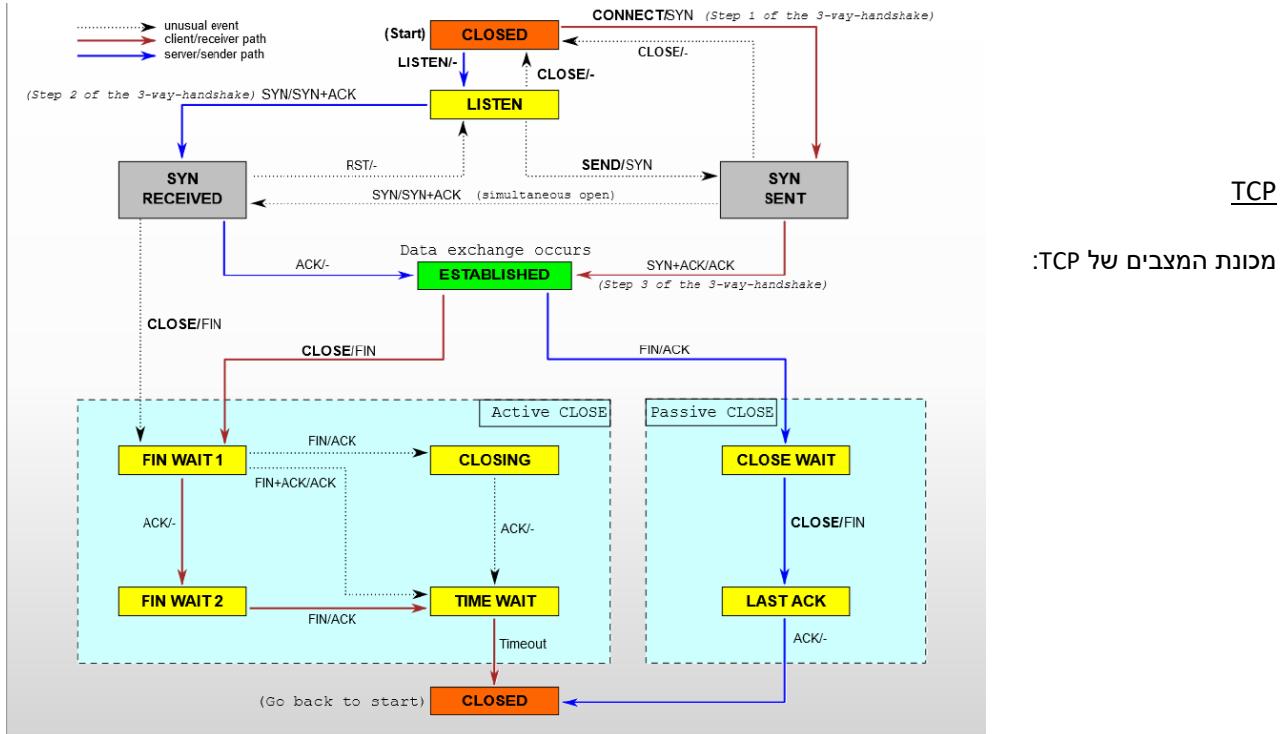
ב-demultiplexing, כל ארבעת הערך משומשים כדי להכוין את הסגמנט לסתוק הנכון.
כדי למצוא את כל הסוקטים שפתוחים במחשב, ניתן להשתמש בפקודה netstat.

UDP



Errors in the datagram detected using a checksum

UDP datagram format
Overhead = 8 bytes



מוכנת המצבים של TCP:

TCP

TCP Segment Header

TCP segment header																				
Octets	Octet	0				1				2				3						
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1				
0	0	Source port												Destination port						
4	32	Sequence number																		
8	64	Acknowledgment number (if ACK set)																		
12	96	Data offset	Reserved 0 0 0	N	S	C	W	E	R	U	R	A	C	P	S	H				
16	128	Checksum												Urgent pointer (if URG set)						
20	160	Options (if data offset > 5. Padded at the end with "0" bits if necessary.)																		
60	480																			

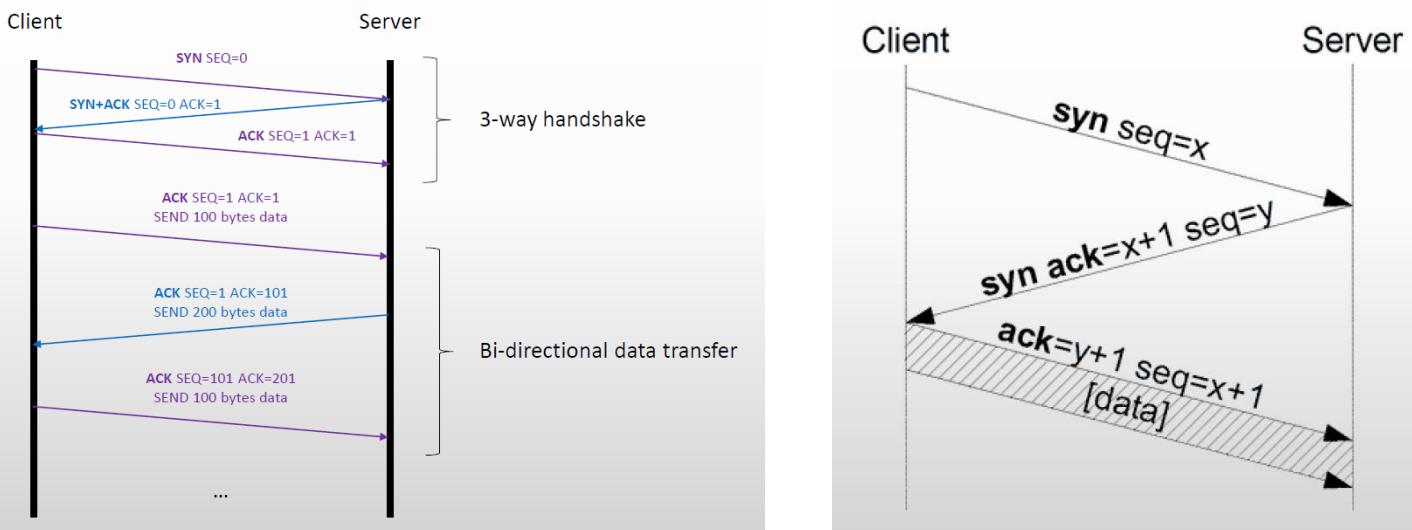
תחילת חיבור - 3-way handshake

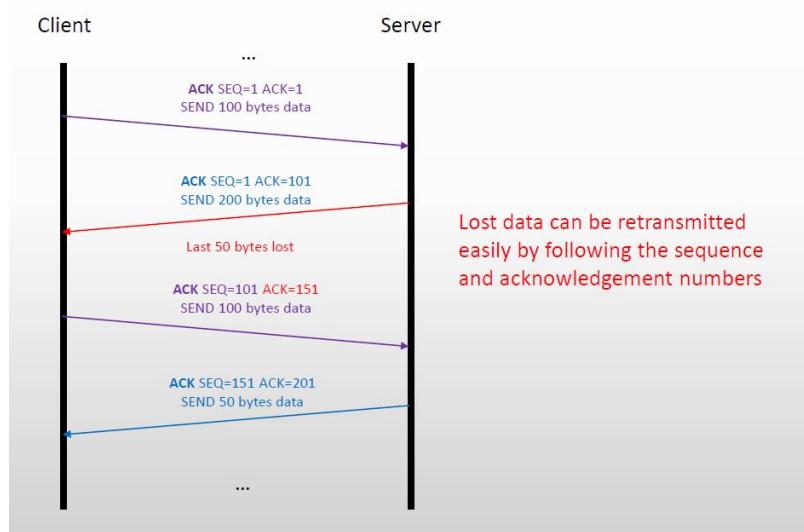
כל חיבור TCP מתחילה ב-3 way handshake. ללא ליחיצת היד, לא ניתן להתחיל להעביר מידע. מספרי ה-sequence וה-acknowledgment משמשים בכך לוודא קבלת נוכנה של נתונים בכל סגמנט.

- Sequence number – מציין את offset מהתחלת המידע שנשלח.

- Acknowledgement number – מציין את המספר של הבית הבא שהמקבל מצפה לקבל.

אם נתונים נאבדו, השולח ידוע על כך בגל ש-*seq* יהיה מספר נמוך יותר מאשר ה-*ack* המוצופה, ועוד ניתן לדעת שצריך לשלוח מחדש.



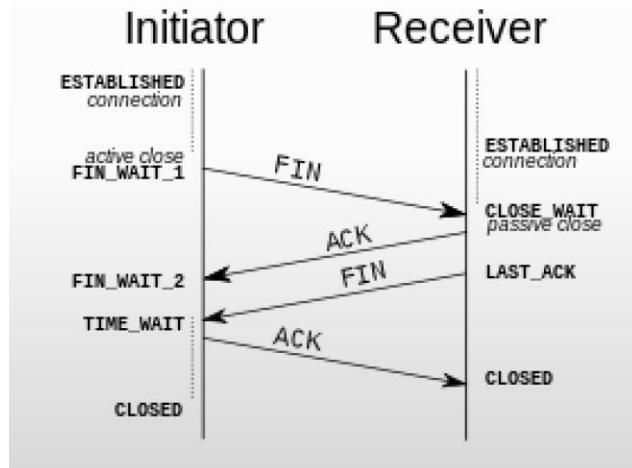


:Sliding Window Flow Control

אם כל סגמנט ב-TCP מצריך ACK, זה יכול להיות מאוד איטי בגלל ה-Round-Trip-Time (זמן שלוקח מרגע שליחת סגמנט עד קבלת תשובה).
לכן במקום לחכות בכל פעם ל-ACK לכל סגמנט, ניתן להגדיר גודל חלון (window size), שיגדר את מספר הסגמנטים שיכולים להשלוח ולהתקבל בכל פעם לפני שנדרש ACK.

TCP משתמש בחילון "מחליק" (sliding window) לביקורת זרימה. בכל פעם שמתקבל מספר ACK, החילון מתקדם הלאה. אם לא מתקבל מספר ACK, כל הסגמנטים בחילון נשלחים מחדש (n-back).

סגירת החיבור:



סגירת החיבור בגיל תקליה:

אם מתרחשת תקלה בחיבור TCP, נשלחת פקטה מיוחדת כדי לסמן שהשולח סגור את החיבור. הפקטה נקראת RST packet. לחיופין, זה יכול לקרות כאשרנוים להתחבר לפורט שאין לו אפליקציה שמזינה בוסוקט (למשל התחברות לפורט 80 בלי שירות אינטרנט כמו Amazon).

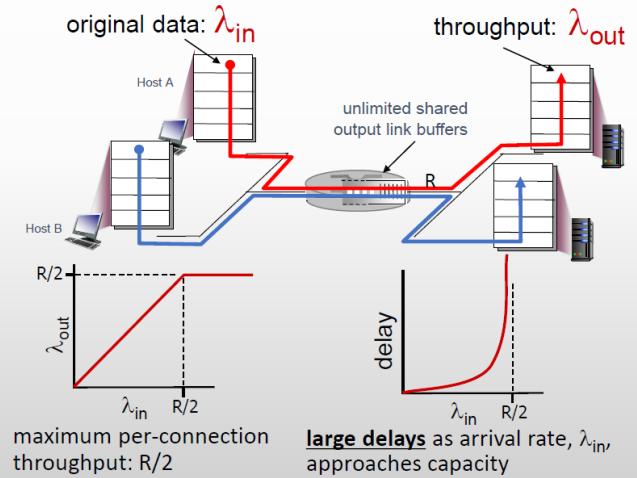
סיבות/עלויות של גודשים:

גודש בראוטר אחד יכול לגרום לרבה פקודות להישלח מחדש, וכך שלבוסף יגרם גודש בחלקים אחרים של הרשת.

מקרה 1: באפר אינסופי בראוטר (לא ריאלי). פקודות ממשיכות להיכנס כל הזמן, ורגע שרווח הפס יתמלא אין יתאפשר לצאת מהබאפר. אין שליחה חדשה של פקודות.

Causes/Costs of Congestion: Scenario 1

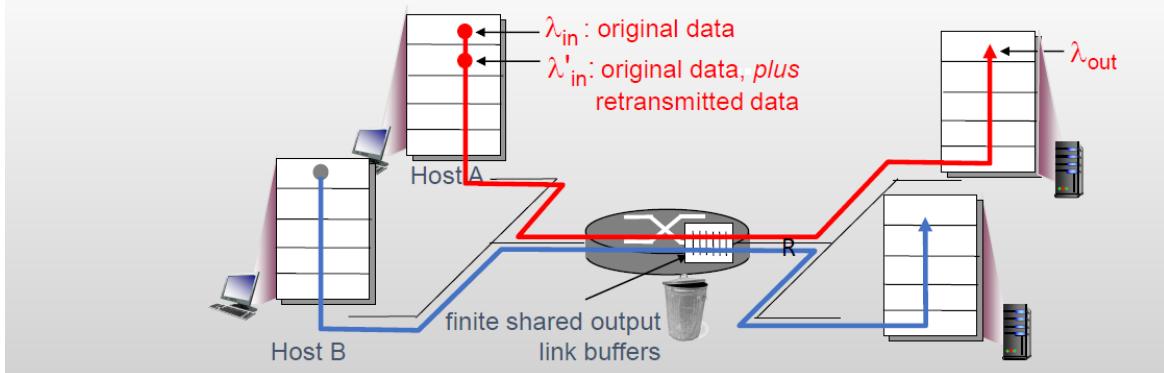
- Two senders, two receivers and fair behavior.
- One router, infinite buffers in router
- Each sender operates at λ_{in} bps (avg)
- Output link capacity: R
- No retransmission



מקרה 2: באפר סופי בראוטר. ברגע שהබאפר מתמלא פקודות מתחילה ליפול ואז באחריות השולח לשלוח אותן מחדש.

Causes/Costs of Congestion: Scenario 2

- One router, finite buffers: dropped packets
- Sender retransmission of timed-out packet
 - application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$
 - transport-layer input includes retransmissions: $\lambda'_{in} \geq \lambda_{in}$



מקרה 3: כמו ב-2 רק עם כמות גודלה יותר של רואוטרים ומארחים.

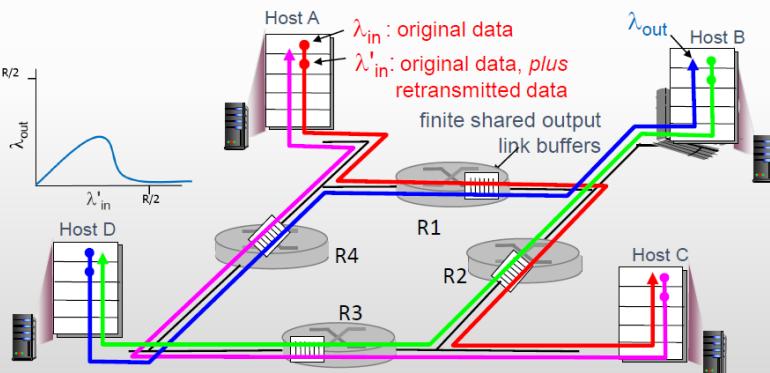


Causes/Costs of Congestion: Scenario 3

- Four senders
- Multi-hop paths
- Timeout/retransmit

As the red λ'_{in} increases, all arriving blue pkts at upper queue are dropped, blue throughput $\rightarrow 0$

When packet dropped, any “upstream” transmission capacity used for that packet was wasted!



גישות להתחממות עם גודשים:

:End-to-end (E2E) congestion control

- אין פידבק מפורש מהרשת.

годש נרמז מערכות קצה שצופות דילוי ואובדן.

- גישה קלאסית ב-TCP.

:Network-assisted congestion control

- ראיוטרים מספקים פידבק למערכות קצה.

ECN = Explicit Congestion Notification . (TCP/IP ENC, IBM SNA, DECnet, ATM)

- בית ייחד מסמן על גודש (TCP/IP ENC, IBM SNA, DECnet, ATM)

לכל שלוח ניתן קצב מפורש לשילוח בו (ATM).

:TCP end-to-end congestion control: AIMD

הגישה- השולח מעלה את קצב השידור (window size) ומהפוך רוב פס להשתמש בו, עד שמתרחש איבוד מידע.

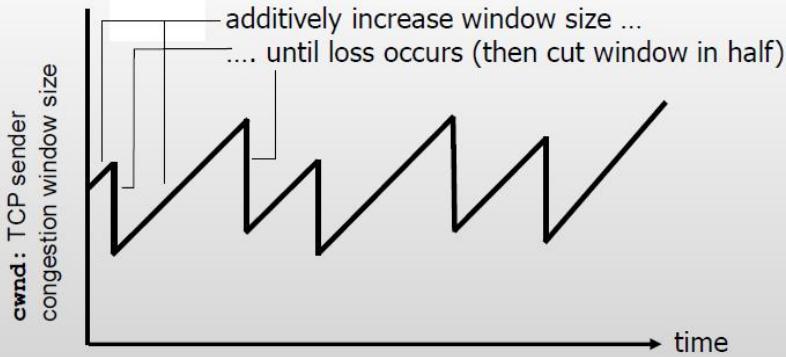
עליה אדטיבית: מעלה את חילון הגודש (cwnd – congestion window) – cwnd (congestion window) ב-1 MSS בכל RTT, עד שמתרחש איבוד מידע.

-

עליה בהכפלת: קיצוץ חילון הגודש בחצי אחריו כל איבוד מידע.

-

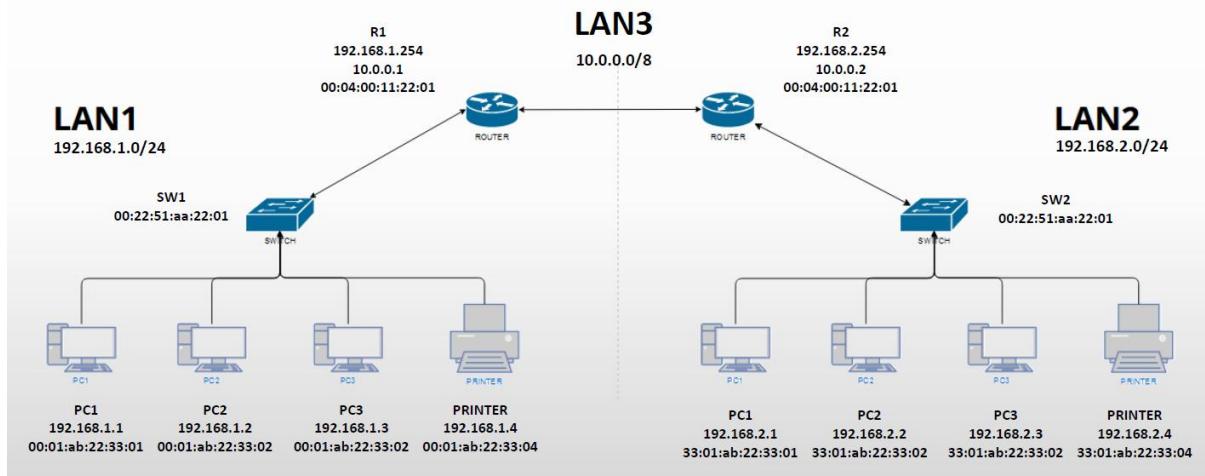
AIMD saw tooth behavior: probing for bandwidth



שכבה 2 (IP / 3 (TCP/IP) - שכבה הרשות

שכבה זו אחראית על ניתוב הפקודות ברשת הלוגית מהשלוח אל היעד. משתמשים בכתובות IP לניטוביים.

Fully Configured Two-LAN Network



מה היא כתובת IP?

כתובת IP מייצגת נקודות סיום לוגיות ברשת.
בכל כתובת IP יש 4 בתים, בד"כ מיוצגים במספרים דצימליים: (0-255).(0-255).(0-255).(0-255).

התנהלות כרטיס רשת ב-IP

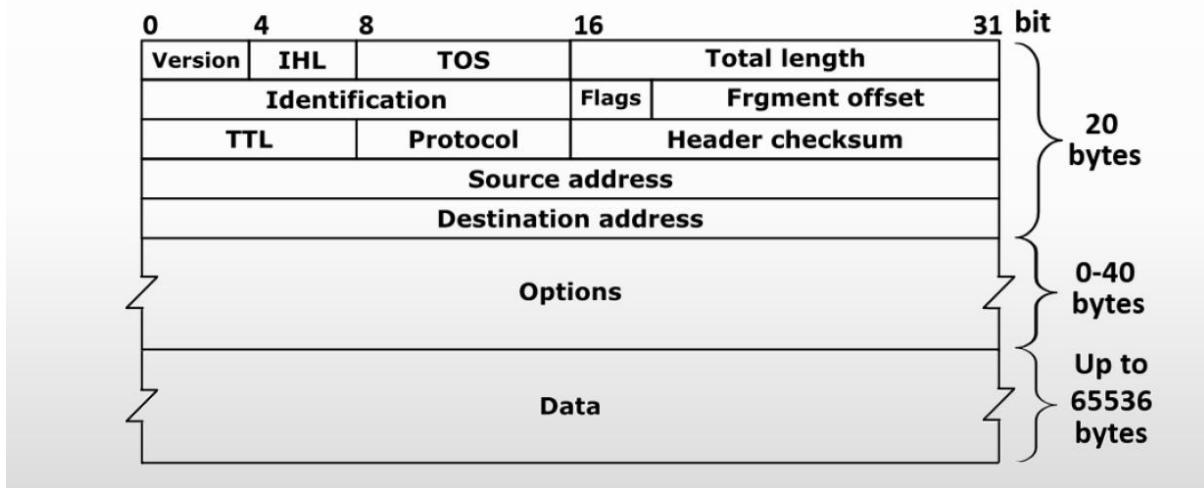
ה-IP המוקנית של כרטיס הרשות. אם כן, הוא "מקלף" (decapsulates) את הפקטה ומעביר אותה מעלה ב-network NIC (Network interface card) מקבל הרבעה פקודות IP. כאשר פקטה מגיעה, הוא בודק אם הפקטה מתאימה לכתובת IP המוקנית של כרטיס הרשות. אם כן, הוא "מקלף" (decapsulates) את הפקטה ומעביר אותה מעלה ב-network, אחרת הוא פשוט מועלם.

מבנה פקטת IP

- Version: גרסה ה-IP, 4 או 6 (IPv4, IPv6).
- Internet Header Length – IHL: גודל header. יכול להשתנות בגלל אופציות משתנות של שדות.
- Type Of Service – TOS: משמש עבור ניהול איכות השירות ועדיפות הפקטה.
- Total Length: סה"כ גודל הפקטה עם הדטה (20-65,535 בתים).
- Identification: משומש כדי להריכיב מחדש פקטים שחולקו לחליום.
- Flags, Fragment offset: משומשים בחלוקת הפקטה לחליום.
- Time To Live – TTL: מספר הקפיצות בין ראותרים שנשאר לפני שהפקטה נופלת.
- Protocol: אומר מה הפרטוקול upper-layer ש"ארוז" (encapsulated) בפקטה זו.
- Header Checksum: – כתובת המקור – Source address
- Destination address – כתובת היעד

- משומש לעיתים רחוקות, כדי לציין שירות ניתוב (routing services) •
- המידע עצמו, עד 65,515 בתים. •

Structure of an IP packet



מיעון לוגי עם IP

משתמשים במיעון לוגי כדי לאפשר ניתוב בין LANs שונים באינטרנט. מארחים מקובצים לוגית לתוך תתי-רשתות (subnets/subnetworks). בפועל הם בד"כ subnets/LANs. למשל בדוגמה הנ"ל יש שני תתי-רשתות: 192.168.1.* and 192.168.2.*. לכל מארח בתת הרשת יש את אותה התחלתה של כתובת.

כתובת IP מחולקת לשני חלקים:

- חלק הרשת (network portion) – מייצג את תת הרשת אליה המארח שייר.
- חלק המארח (host portion) – מייצג את המארח הספציפי.

כתובות מיוחדות:

כתובת ה-IP הראשונה בתת-רשת (למשל 192.168.1.0) תמיד שומרה לתת הרשת עצמה.
כתובת ה-IP האחרונה בתת-רשת (למשל 192.168.1.255) תמיד שומרה לכתובת broadcast. פקודות שנשלחות לכתובות זו יגיעו לכל המארחים בתת הרשת. כתובת broadcast היא הכתובת שבא כל הביטים בחלק המארח הם 1.

קלואים:

כש-IP ריק נוצר, כתובות רשת חולקו לקלואים כתלות בגודל שנייהן לחילוק הרשת:

- 1 בית לחילוק הרשת, 3 בתים לחילוק המארח. :Class A
- 2 בתים לכל חלק. :Class B
- 3 בתים לחילוק הרשת, בית 1 לחילוק המארח. :Class C

Class	Network Portion Size	Host Portion Size	Number of Networks	Number of addresses per network (including network address and broadcast)	Start Address	End Address
Class A	8	24	128	16,777,216 (2^{24})	0.0.0.0	127.255.255.255
Class B	16	16	16,384	65,563 (2^{16})	128.0.0.0	191.255.255.255
Class C	24	8	2,097,152	256 (2^8)	192.0.0.0	223.255.255.255
Class D (multicast)			N/A		224.0.0.0	239.255.255.255
Class E (reserved)			N/A		240.0.0.0	255.255.255.255

שימוש בקלאסים הוא מאד פשוט, אך ייצרו את CIDR. CIDR הוא פרטן נוסף שנדרה מסכת תחת הרשת (subnet mask), והוא מגדיר כמה ביטים יהיו בחלק הרשת וכמה בחלק המארח (לא רק חלוקה של 8/16/24). בצורה זו, רשותת יכולות להיות מוגדרות ברחוקה יותר טובה.
את subnet mask כתבים אחרי הכתובת עם '/'. למשל: 192.168.1.1/24, כלומר בחלק הרשת יש 24 ביטים. ניתן גם לרשום בתוור מספר: 192.168.1.1/24 == 192.168.1.1, netmask 255.255.255.0

כדי למצוא את כתובות הרשת, מבצעים פועלות **&** (AND) על כתובות IP וה-subnet mask == Network address
IP Address & Subnet mask == Network address

- Suppose we have a device with IP address 192.168.2.1 and subnet mask of 24 bits
- What is the network address?

192.168.2.1/24

C0.A8.02.01/24

1100 0000 . 1010 1000 . 0000 0010 . 0000 0001 (**IP Address**)

1111 1111 . 1111 1111 . 1111 1111 . 0000 0000 (**Subnet mask**)

- Now, let us consider a less obvious example

192.168.2.1/22

C0.A8.02.01/22

1100 0000 . 1010 1000 . 0000 0010 . 0000 0001 (**IP Address**)

1111 1111 . 1111 1111 . 1111 1111 . 0000 0000 (**Subnet mask**)

- What is the network address?

1100 0000 . 1010 1000 . 0000 0000 == 192.168.0.0

- What is the broadcast address?

1100 0000 . 1010 1000 . 0000 0011 . 1111 1111 == 192.168.3.255

- Now, let us consider another example

192.168.2.1/26

C0.A8.02.01/26

1100 0000 . 1010 1000 . 0000 0010 . 0000 0001 (**IP Address**)

1111 1111 . 1111 1111 . 1111 1111 . 1100 0000 (**Subnet mask**)

- What is the network address?

1100 0000 . 1010 1000 . 0000 0010 . 0000 0000 == 192.168.2.0

- What is the broadcast address?

1100 0000 . 1010 1000 . 0000 0010 . 0011 1111 == 192.168.2.63

כתובות IP מיוחדות:

ישן כתובות IP ששמרות לשימושים ספציפיים.

Address block	Scope
0.0.0.0/8	Current Network
10.0.0.0/8	Private network
100.64.0.0/10	Private network
127.0.0.0/8	Loopback
169.254.0.0/16	Link Local
172.16.0.0/12	Private network
192.0.0.0/24	Private network
192.0.2.0/24	Testing
192.88.99.0/24	Reserved
192.168.0.0/16	Private network
198.18.0.0/15	Testing
198.51.100.0/24	Testing
203.0.113.0/24	Testing
224.0.0.0/4	IP Multicast
233.252.0.0/24	Testing
240.0.0.0/4	Reserved
255.255.255.255/32	Broadcast

נתבים (Routers)

כאשר מארח שלוח פקטה למארח אחר באותה תת רשת, הפקטה נשלחת ישירות. היא כנראה עובר דרך Switch (מכ舍יר של שכבה 2 שמעביר מסגרות לפ' הכתובת הפיזית שלה). למורთ זאת, כאשר מארח רוצה לשלוח פקטה למארח שלא נמצא באותה תת רשת, הוא חייב להשתמש במכ舍יר שנקרא נתב (Router) כדי להעביר את הפקטה לתת רשת אחרת. הנתב זהה נקרא Default Gateway. ראים מחברים תתי רשת שונות ביחד. זה אפשרי שיתר מהת רשת אחת תהיה מחוברת לאותו ראייר, או שייהי ראייר אחד לכל תת רשת אחת.

פונקציות חשובות בשכבה הרשת

יתוב (Routing):

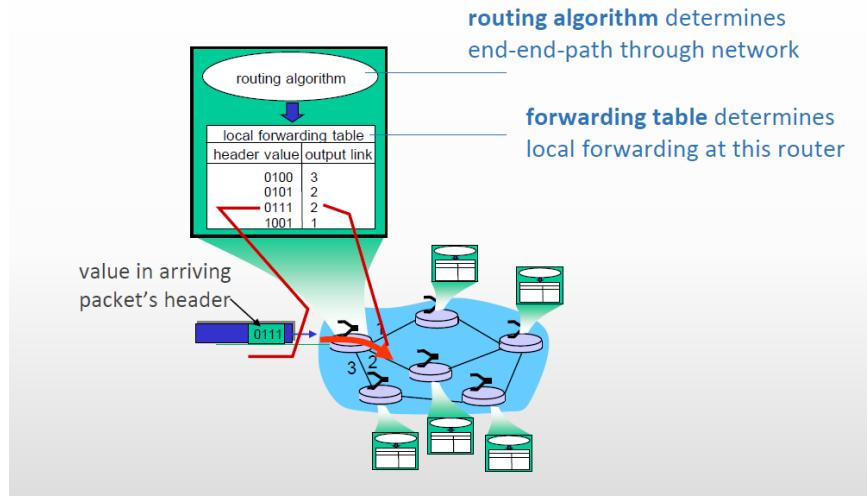
קביעת המסלול של הפקודות מהמקור אל היעד. אלגוריתמים שמחשבים את כל הדרך מהמקור אל היעד נקראים אלגוריתמי יתוב.

העברה (Forwarding):

העברה הפקטות מרגל אחת של הראייר לרגליים האחרות. פעולה לוקלית של הראייר.

ראייר מקבל הודעות פרוטוקול יתוב שימושיות ע"י אלגוריתם הניתוב. כתוצאה מאלגוריתם הניתוב נוצרת טבלת יתוב (Forwarding table).

אלגוריתם ניתוב: קובע את הדרך מcka להקצה ברשת.
טבלת ניתוב: קובעת את העברות בין הרגלים של אותו רוטר.



טבלת ניתוב (FIB)

מכילה מפה שמצוינה את המקשר لأن פקטה אמורה להגיע. בד"כ משתמשת ב-rule longest match rule, כלומר כל שיש יותר ביטים ל-network portion הניתוב יהיה יותר מדויק.

Network Prefix	Output Link	Next hop
192.168.1.0/24	eth0	
10.0.0.0/8	eth1	
0.0.0.0 (anything else)	eth1	10.0.0.2

איך לזרוק פקודות בתור?

ישנם אלגוריתמים שפותחו שזרקים פקודות ברגע שהטור מלא. ישםו אלגוריתם RED (Random Early Detection) (בצורה סלקטיבית) מונע רצף של נפילות בהרבה חיבור TCP, מה שיכל לגרום לתגובה מסונכרנת שבה יותר מדי חיבורים מורדים ובו זמינות מעלים את קצב שליחת ההודעות.

(NAT) Network Address Translation

כתובות IP אמורות להיות ייחודיות, וכשיצרו את ה-IP לא חשובו שייהי יותר מ-2 בחזקת 32 (4 בתים) מקשרים בכל העולם. لكن היו צריכים לחשב על דרך אחרת לייצוג מקשרים. כדי להתגבר על הבעיה, נוצרו כמה טכנולוגיות כמו IPv6 ו-NAT.

ה-NAT מרלב (multiplexes) את ה-IP בשכבות התעבורה. הוא מחזיק רשימה של כל ה포רטים שהוא מקשרים וה포רטים מאחוריו ה-NAT. External port => Internal IP + Port
מחוץ ל-NAT, כל התעבורה נראה כאילו היא מגעה מכטבת אחת. מאחוריו ה-NAT, משתמשים בכתובות רשות פרטיות.

אחד המאפיינים של NATs זה שהם גורמים להרבה איבוד של מידע (שני פרמטרים – IP ופורט – מצומצמים לאחד – פורט). זה אומר שמחוץ ל-NAT מארחים לא יכולים ליצור תקשורת עם מארחים מאחורי ה-NAT, אלא אם הם משתמשים ב-Port forwarding.

Port forwarding: מגדיר פорт קבוע ב-NAT שמננו יעברו פקודות לצירוף port+IP פרטיו מאחורי ה-NAT.

הגדרת כתובת IP

הגדרה ידנית:

ניתן להגדיר כתובת IP בצורה ידנית ולקבוע אותה בהגדרות הרשות. צריך לוודא שהיא תקינה (valid) אחרת יתכן ולא תעבור מוחץ לרשות הפרטית.

(Dynamic Host Configuration Protocol) DHCP

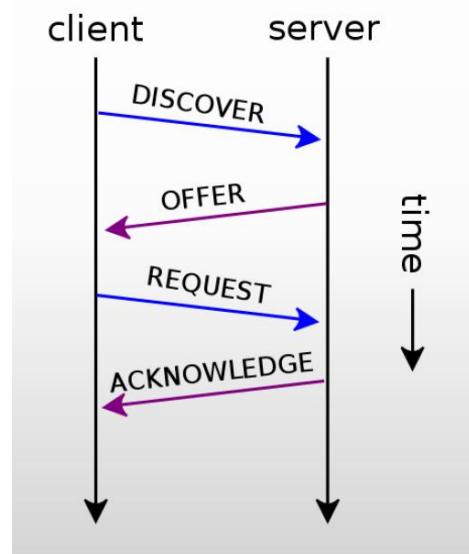
דרך אוטומטית לקבל כתובת IP. אם מחברים למחשב זה "פושט עובד" – סימן DHCP שומש.

DHCP

DHCP הוא פרוטוקול בשכבה האפליקציה אשר מנהל את קביעת כתובות ה-IP ברשות. הוא משתמש בארכיטקטורת ל Koh-Shart. מכשיר מתחבר לרשת ושולח broadcast כדי "לגלות" את שרת DHCP. שרת DHCP מציע למיכסир את ה-IP, המכיסיר מקבל את ההצעה. כתובת IP היא "מושאלת", זו שהיא עלול לפג ולהיא תשנה. בד"כ הלוקו מקבל גם את default gateway וה subnet mask.

:DORA

- Discover – הלוקו שולח פקחת broadcast discovery לכל המחשבים ברשת המקומיית על מנת לאתר שרת DHCP. בפקחה זו לא מוכנסת כתובת IP של היעד, ולכן מגיע לכל המחשבים ב- LAN.
- Offer – אם קיימים שרתים DHCP ברשת המקומיית בעלי כתובת פניהם לחילוקה, הלוקו מקבל חבילה offer עם כתובת IP מכל אחד מהם.
- Request – הלוקו שולח חבילה request עם הנתונים אותם בחר, גם כן ב广播 על מנת לידע את כל השירותים בכתובת שנבחרה.
- Acknowledge – השרת שולח פקחת acknowledge, אישור שהוא קיבל את הבקשה. לאחר בקשה זו המחשב מתחילה להשתמש בנתונים שקיבל. מעבר לכתובת IP, בחבילה זו נשלחים לרוב הפרטים המאפשרים להתנהל ברשת: default gateway, subnet mask ועוד.



התמודדות עם שגיאות/בקשות למידע בשכבה הרשת

.network-level ICMP (Internet Control Message Protocol) מושמש ע"י מארחים וראוטרים כדי להעביר מידע ב-level-network.).

דיווח שגיאות:

- Unreachable host
- Unreachable network
- Unreachable port
- Unreachable protocol
- Network/host unknown
- TTL expired
- Etc

בדיקה שマארח "ער": שליחת בקשת/תשובה ICMP Echo (פקודת Ping).

Traceroute and ICMP

ניתן להשתמש ב-ICMP כדי לדבג את החיבור שלנו לרשת. Traceroute: שולחת פקודות עם TTL שעולה בכל פעם, מתחילה ב-1 ועד שmaguiim למארח. בכל פעם שmaguiim לראים, הראורט שולח בהזאה הדריך חזרה פקודה ICMP עם ה-TTL שהגיע אליו וירד בכל פעם. ככל יותר אחרי הקפיצה הראשונה, ה-TTL יגיע לאפס ופקחת ה-ICMP תשליח חזרה ותראה את כתובת הראורט שבקפיצה הראשונה. אחר כך, פקודה עם TTL 2 תישלח שתגיע לאפס בראורט השני, ותחזור עם כתובות הראורט השני וכן הלאה. בסופו נקבל רשימה של כל הראורטים ביןינו לבין המארח הרצוי. ניתן להוציא עוד מידע מהפקודה כמו RTT, names resolved ועוד.

ניתוב

ניתן לדמיין את הרשת כמו גרפף מכון ($E, N = G$) שבו N הוא סט הצלמים ו- E סט הקשרות. קשרות מחוברות בין כל צומת לשכן ("מחיר" cost). המטריה של ניתוב היא למצוא את הדרך הזולה ביותר.

סוגים של אלגוריתמי ניתוב:

- גלובלי: לכל הראורטים יש טופולוגיה שלמה ומידע על מחירי קווים. לרוב מօcarsים אלגוריתמי "link state".
- מבוזר: ראורט ידוע מי השכנים שפיזית מחוברים אליו ואת עלויות הקווים לשכנים אלו. אף צומת אין את כל המידע. ישנן פעולות חוזרות של חישוב והחלפת מידע עם השכנים. מתודה אחת: אלגוריתמי "distance vector".

לינק-סטט – אלגוריתם ניתוב link-state

כל צומת עושים broadcast עם החיבור והעלויות שלו. לכל הצלמים יש את אותו המידע. מחשב את כל עלויות הדריכים מצומת אחד (source) לכל הצלמים האחרים. נותן את טבלת הניתוב (forwarding table) לצומת זו. איטרטיבי – אחרי K איטרציות,ណע את העליות ל-K יעדים.

סימונים:

- $C(x,y)$ – עלות קו (link cost) מצומת X ל-Y. אם הצלמים לא שכנים אז $\infty = C(x,y)$.
- $D(v)$ – הערך הנוכחי של עלות הדרך מהמקור לצומת V.
- $P(v)$ – צומת אב קודם (predecessor) בדרך מהמקור לצומת V.
- N – סט הצלמים שכבר ביצרנו בהם והדרך הזולה ביותר כבר ידועה בהם.

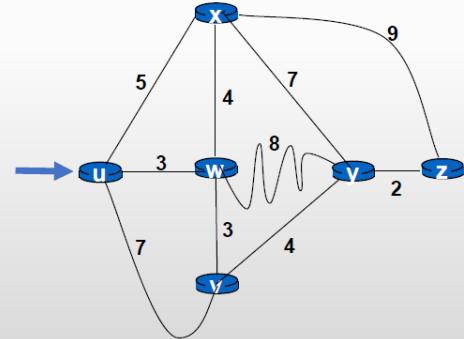
```

1  Initialization:      The node for which we are
2  N' = {u} ← trying to find its minimum
3  for all nodes v      cost to all other destinations
4  if v adjacent to u
5  then D(v) = c(u,v)
6  else D(v) = ∞
7

```

$p(v)$ – predecessor node to v

Step	N'	D(v)	D(w)	D(x)	D(y)	D(z)
0	u	7,u	3,u	5,u	∞	∞



```

1  Initialization:
2  N' = {u} ←
3  for all nodes v
4  if v adjacent to u
5  then D(v) = c(u,v)
6  else D(v) = ∞
7
8  Loop
9  find w not in N' such that D(w) is a minimum ←
10 add w to N'
11 update D(v) for all v adjacent to w and not in N' :
12   D(v) = min( D(v), D(w) + c(w,v) )
13 /* new cost to v is either old cost to v or known
14   shortest path cost to w plus cost from w to v */
15 until all N nodes in N'

```

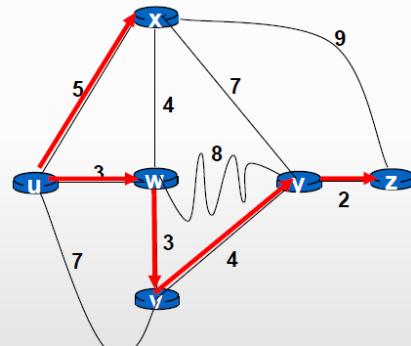
The node for which we are trying to find its minimum cost to all other destinations

D(v): current value of cost of path from source to destination v

The node whose distance from u is minimum

Step	N'	D(v)	D(w)	D(x)	D(y)	D(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w	5,u	11,w	∞	∞
2	uwx	6,w		11,w	14,x	
3	uwxv			10,y	14,x	
4	uwxvy				12,y	
5	uwxvyz					

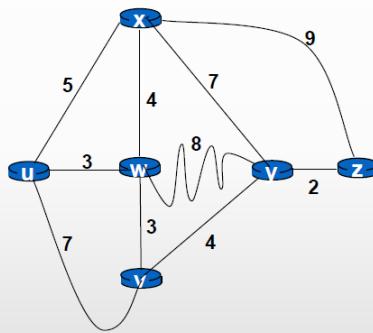
ע"ז של הדרך הקצרה נבנה ע"י מעקב אחר צמתי האב. תיקו יכול להתקיים, מקרים כאלה אוטם באופן שבירורו.



- Notes:
- Construct shortest path tree by tracing predecessor nodes
- Ties can exist (can be broken arbitrarily)

ברגע שהטבלה הנו'ל נמצאה, ניתן לקבוע את טבלת הניטוב של צומת U:

Step	N'	D(v)	D(w)	D(x)	D(y)	D(z)
		p(v)	p(w)	p(x)	p(y)	p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



Let

$d_x(y) := \text{cost of least-cost path from } x \text{ to } y$

then

$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

cost from neighbor v to destination y

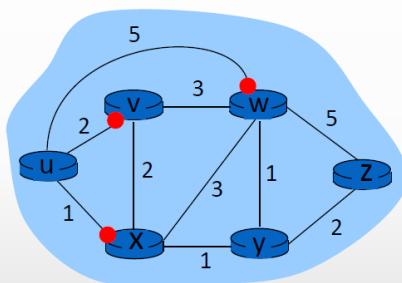
cost to neighbor v (of x)

\min taken over all neighbors v of x

בלמן פורד – אלגוריתם Distance Vector

אלגוריתם איטרטיבי, אסינכראוני.

Bellman-Ford example



u wants to find out its minimum cost to z.

u has 3 neighbors: v, x and w

Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

Bellman-Ford equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,x) + d_x(z), \\ &\quad c(u,w) + d_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

- Node achieving minimum is *next hop in shortest path, used in forwarding table*.
- Bellman-Ford equation suggests form of neighbor-to-neighbor communication.

Distance vector algorithm

- Each node x maintains the following routing information:
 - For each neighbor v , the cost $c(x,v)$ from x to directly attached neighbor, v
 - Node x 's **distance vector**, that is, $D_x = [D_x(y): y \in N]$ containing x 's estimate of its cost to all destinations, y, in N
 - The distance vectors of each of its neighbors, that is, $D_v = [D_v(y): y \in N]$ for each neighbor v of x
- **Key idea:**
 - From time-to-time, each node sends its own distance vector (DV) estimate to neighbors
 - When x receives new DV estimate from neighbor v , it updates its own DV **using B-F eq.:**

$$D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \} \text{ for each node } y \in N$$

- As long as all nodes continue to exchange distance vectors in, estimate $D_x(y)$ converges to actual least cost $d_x(y)$

Distance vector algorithm

- **Iterative, asynchronous:**

- Each local iteration at a node caused by:
 - Local link cost change
 - DV update message from neighbor node

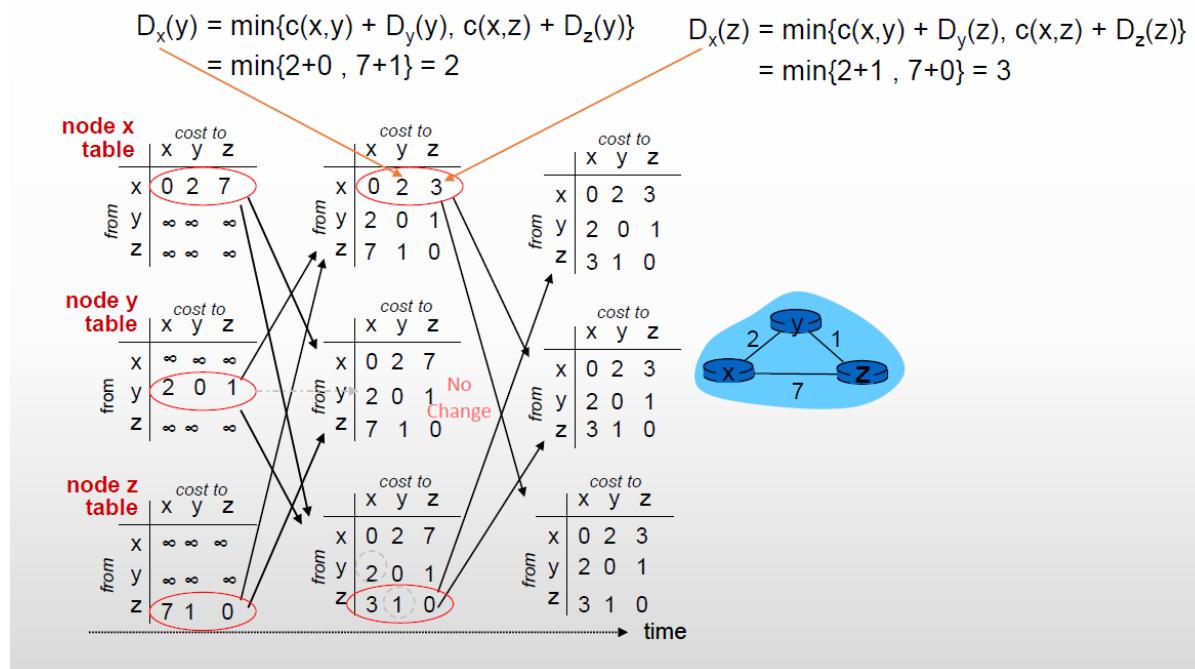
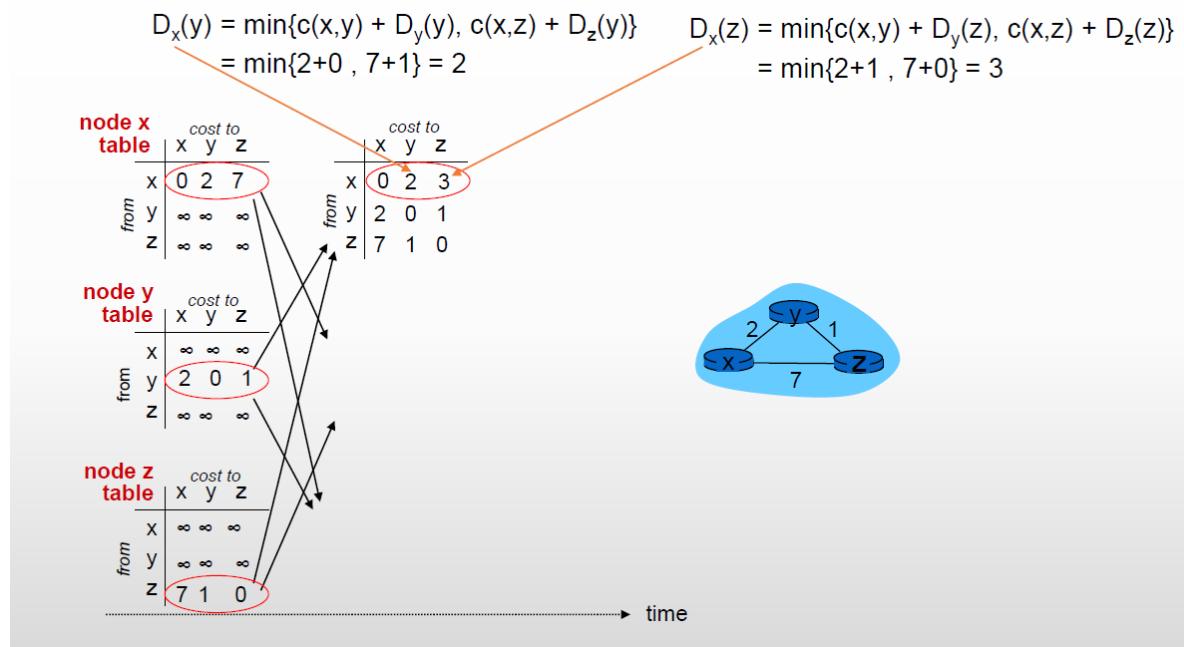
- **Distributed:**

- Each node notifies neighbors **only when its DV changes**
 - Neighbors then notify their neighbors if necessary

- **Each node:**

```

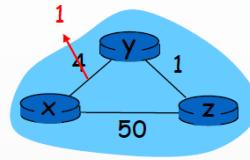
    wait for (change in local link
    cost or msg from neighbor)
    recompute estimates
    if D to any dest has changed,
    notify neighbors
  
```



Distance vector: effect of link cost changes

Link cost changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbors

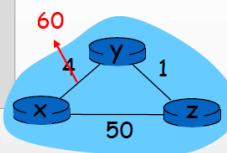


“good news travels fast”

- t_0 : **y** detects link-cost change, updates its DV, informs its neighbors.
- t_1 : **z** receives update from **y**, updates its table, computes new least cost to **x** (decreased from 5 to 2), sends its neighbors its DV.
- t_2 : **y** receives **z**'s update, updates its distance table. **y**'s least costs do *not* change, so **y** does *not* send a message to **z**. Stops.

Distance vector: link cost changes

Y knows Z can route to X at a cost of 5.
Y updates its table and sends update to Z (cost 6 to X).
Z recomputes and sees it can route to X through Y at a cost of 7.
Z updates and sends to Y.
Y recomputes...



Link cost changes:

- Node detects local link cost change
- **Bad news travels slow** – “count to infinity” problem!
- 44 iterations before algorithm stabilizes.

Poisoned reverse:

- ❖ **If Z routes through Y to get to X :**
 - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- ❖ Poisoned reverse solve loops between 2 nodes only. When multiple nodes are involved, it does not solve it.

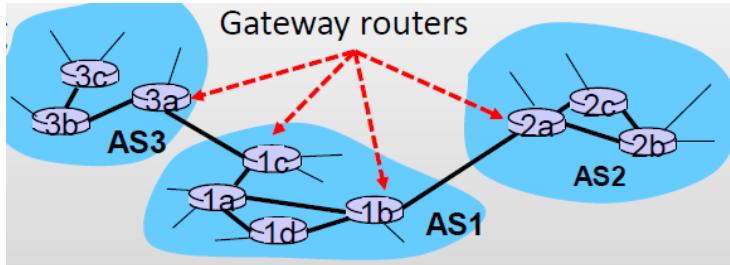
ניתוב היררכי

יש יותר מ-600 מיליון דרכים ברשות ולא ניתן לשמר את כלן בטבלאות ניתוב. בנוסח, טבלאות הניתוב משתנות כל הזמן. האינטרנט היא רשת של רשתות, ובכל רשת האחראי יכול לקבוע איך יהיה הניתוב.

הweeney בניתוב היררכי הוא לקבץ ראותרים לאזורים, למערכות אוטונומיות (Autonomous Systems – AS –). ראותרים באותו AS יש את אותו פרוטוקול ניתוב: פרוטוקול ניתוב "Intra-AS" (Within AS). ראותרים ב-AS שונים יכולים להריץ ניתובי intra-AS שונים. ב"קצה" של כל אחת AS, יש **Gateway router**. לראותר זה יש חיבור לראוטר שער בין-AS אחרה. ראותרי שער מרכיבים פרוטוקול ניתוב "inter-AS" (Between AS) ביניהם, כמו פרוטוקול שמחברים בין כל AS.

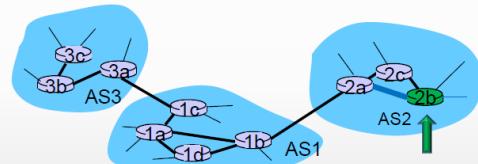
Interconnected ASes

- ישנה טבלת ניתוב שנקבעת גם ע"י אלגוריתם ניתוב intra-AS וגם ע"י אלגוריתם ניתוב inter-AS.
- פרוטוקול ניתוב AS Intra קובע כניסה עבור יעדים פנימיים ב-AS.
 - גם AS Inter-AS וגם Intra AS קובעים כניסה לעדדים חיצוניים.



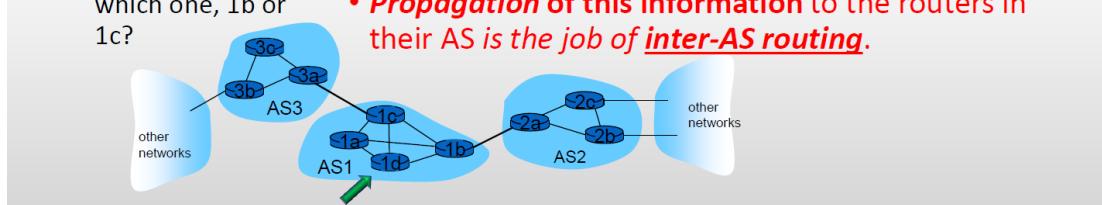
Inter-AS tasks – simple case

- Suppose router **2b** in AS2 receives a datagram destined **outside of AS2**:
 - Router should forward packet to **gateway router**.
 - Since there is only one exit from AS2 (2a) and this router is part of AS2, **intra-AS routing determines route to 2a**.



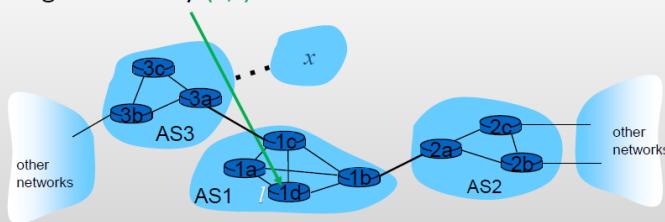
Inter-AS tasks – non-trivial case

- Suppose router 1d in AS1 receives datagram destined outside of AS1:
 - Router should forward packet to gateway router, but which one, 1b or 1c?
- **Gateway routers in AS1 (1c and 1b) must:**
 1. Learn which destinations are reachable through AS2, which through AS3
 2. **Propagate this reachability info to all routers in AS1** so that they can configure forwarding table.
- Gateway routers in all AS's must run same **inter-AS protocol** among them
- **Propagation of this information to the routers in their AS is the job of inter-AS routing.**



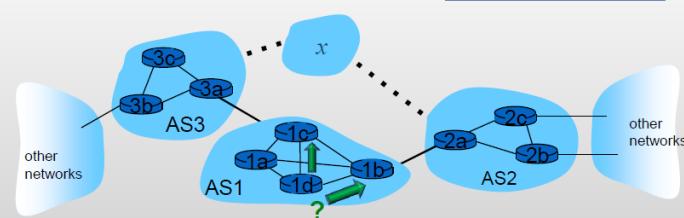
Example: setting forwarding table in router 1d

- Suppose **all routers in AS1** learn (**via information propagated from the gateway routers' inter-AS protocol**) that subnet x reachable via AS3 (gateway 1c), but not via AS2
- Router 1d needs to update its forwarding table how to reach subnet x .
 - Router 1d determines from **intra-AS routing** info that its interface **l** is on the least cost path to 1c
 - Installs forwarding table entry **(x,l)**



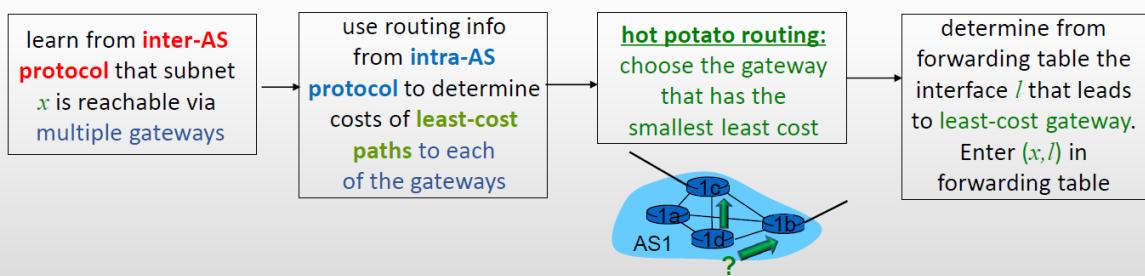
Example: choosing among multiple ASs

- Now suppose AS1 learns from inter-AS protocol that subnet x is **reachable from AS3 and from AS2**.
- The **inter-AS routing protocol** has propagated **two different possibilities** to all routers in AS1!
 - To configure forwarding table, router 1d must determine towards which of the two gateways it should forward packets for destination x
 - To determine which of them to chose, router 1d will seek the help of the **intra-AS protocol**



Example: choosing among multiple ASs

- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x
- Hot potato routing:** send packet towards closest of two routers.



פרוטוקול ניתוב Intra-AS

ידועים גם בתור IGP (Interior Gateway Protocols).

פרוטוקול Intra-AS הנפוצים ביותר:

- Routing Information Protocol – RIP

- Open Shortest Path First – OSPF

- (closely related to OSPF) Intermediate System to Intermediate System – IS-IS

- (Cisco proprietary) Interior Gateway Routing Protocol – IGRP

(Routing Information Protocol) RIP

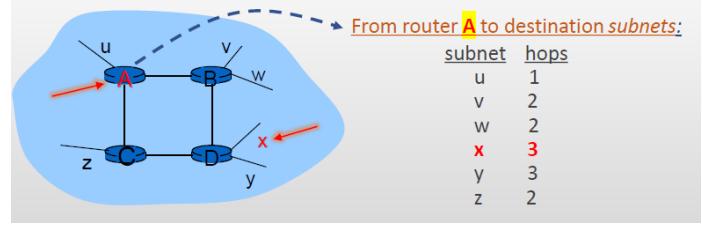
RIP משתמש בפרוטוקול Distance Vector, מבוסס על בלמן-פורד.

Distance metric: מספר הקפיצות (קוויים) כדי להגיע ל-Subnet IP (לא לмарח) מראותר. למשל, $3 = d(A \rightarrow x)$.

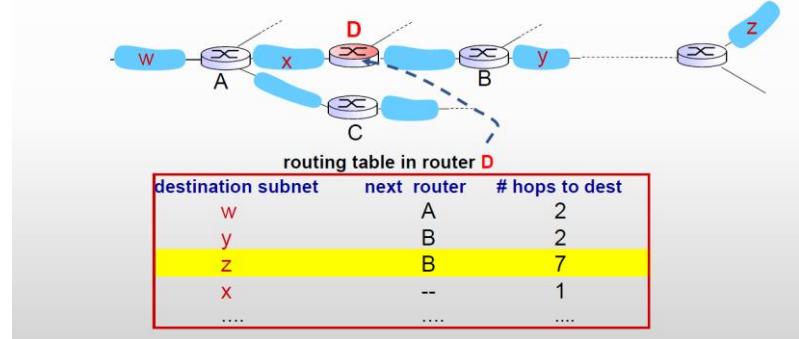
כל קפיצה (קו) יש מחיר שווה ל-1.

марחק מקסימלי = 15 קפיצות -> המגבלה של Autonomous System.

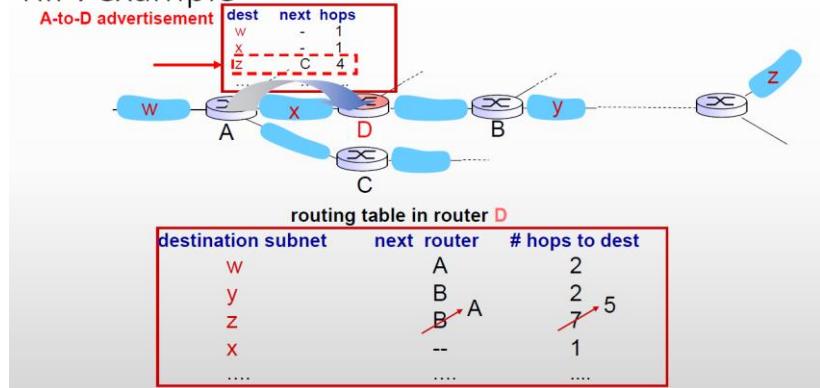
וקטורי מרחק מוחלפים עם שכנים בכל 30 שניות ב-RIP response message (ידעו גם כ-advertisement, פרסום).



RIP: example



RIP: example



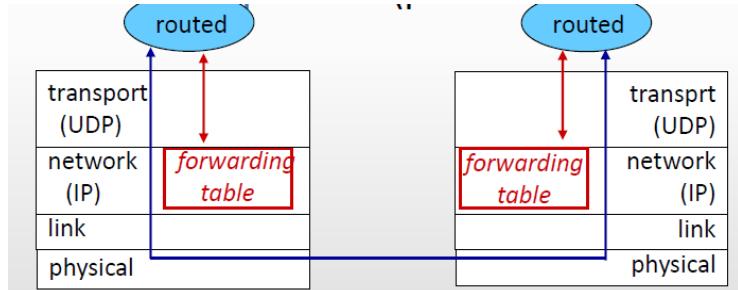
כישלון חיבורם והתאוששות:

וקטורי מרחק מוחלפים עם שכנים בכל 30 שניות. אם לא שומעים advertisement אחרי 180 שניות, השcn/הkn מוכרא "מת".

ראוטרים ליד אותו SCN מבוטלים (invalidated). Advertisements עם טבלה מעודכנת נשלחות לשכנים. השכנים בתורם שלוחים מחדש Advertisements (אם הטבלאות שלהם השתנו). מידע על כישלון חיבור מופץ מהר לכל הרשת. מעוד כדי למנוע "ping-pong loops".

טבלת עיבוד של RIP:

טבלאות ניתוב של RIP מנוהלות ע"י תהילר של רואוטר שנמצא ברמת האפליקציה, התהילר נקרא **routed** (route). Advertisements נשלחים ב-UDP (פורט 250) בזורה מחזיר.



(Open Shortest Path First) OSPF

פומץ בניתוב intra-AS. Publicly available = "Open" ב인터넷. משתמש באלגוריתם link state algorithm.

- הפעלת פקודות link state.
- הטופולוגיה של כל ה-AS נמצאת בכל צומת.
- חישוב דרך בעזרת אלגוריתם דיאקסטרה.
- מחירי החיבורם קבועים ע"י האדמין.

advertisements מוצפים לכל ה-AS. מועברים בהודעות OSPF ישרות ב-IP (לא ב-TCP או UDP). פорт 89, פרוטוקול שכבהعلונה (upper layer protocol).

יתרונות OSPF:

- אבטחה: כל הודעות OSPF מאומנות (authenticated) כדי למנוע וירוסים ותוכנות זדוניות.
- מותר להיות כמה דרכיים עם אותו מחיר. אם לכל הדרכים לאותו יעד יש אותו מחיר, אפשר להשתמש בכלם (load balancing).
- לכל חיבור מותר להיות כמה מטריות מחיר ל-SoT (Type of service) שונים. לדוגמה: מחיר חיבור ללווין מוגדר "מספר" בשביל SoT הכי טוב. גובה בשביל SoT בזמן אמת (בגלל שיש הרבה דילוי).

פרוטוקול intra-AS-היררכי

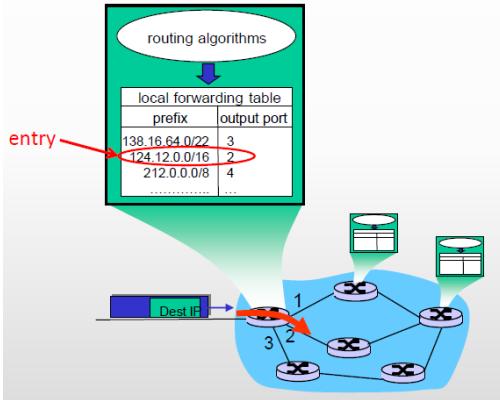
המנהל (administrator) של AS יבחר באיזה פרוטוקול ניתוב intra-AS למשמש בין הרואוטרים ב-AS. הפרוטוקול ניתוב קובע לכל הרואוטרים ב-AS את הכניסות בטבלאות הניתוב לכל ה-subnets בתחום ה-AS. הפרוטוקול ניתוב עוזר גם לפרוטוקול Inter-AS לקבע כניסה בטבלת הניתוב שלו ל-subnets או ה-prefixes שממוקמים מחוץ ל-AS.

ניתוב אינטרנט Inter-AS: BGP

הוא הпрוטוקול inter-domain הקבוע. "Glue that holds the internet together"

BGP מספק לכל AS:

- מידע על אפשרות הגעה לשכנים של subnet :external (eBGP).
- מפזר מידע על אפשרות הגעה לכל הראותרים ב-AS :internal (iBGP).
- קובע דרכי "טבות" לרשותות אחרות על בסיס מידע של אפשרות הגעה ותקנון מסוים.



איך נכנסה כניסה לטבלת הניתוב?

מניחים ש-prefix שורצים להכליל בטבלת הניתוב בנמצא ב-AS אחר.

ישנם 3 שלבים:

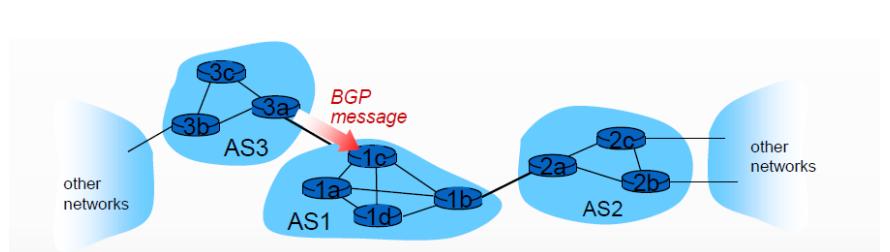
- הראותר נניה מודע ל-prefix.
- הראותר קובע את יציאת הפורט ל-prefix.
- הראותר מכניס את הפורט של ה-prefix לטבלת הניתוב.

שלב 1:

מתיקבת הודעת BGP שמכילה את מה שה-Route קורא לו "דרכי" (routes). זה ה-prefix והתכונות שלו: AS-PATH, NEXT-HOP

Prefix: _____ ; AS-PATH: _____ ; NEXT-HOP: _____

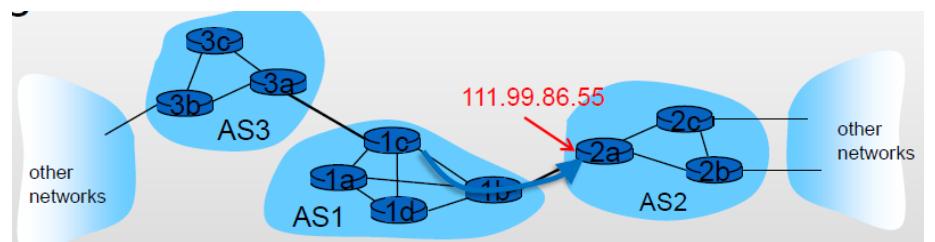
לדוגמה: הדריך התקבלה ע"י 1c מ-3a-3 AS3 route: 138.16.64.0/22; AS-PATH: AS3 AS131 AS201; NEXT-HOP: 201.44.13.125



הראותר יכול לקבל יותר דרך אחת עבור אותו prefix וצריך לבחור אחת מה דרכים. בהנחה שאין חוק בתקנון בדרגה גבוהה יותר לקביעת הדרך, הראותר בוחר את הדרך בהתבסס על הדרך הקצרה ביותר ל-AS.

כדי למצוא את ה-intra-route הטובה ביותר לrouteNEXT-HOP, משתמשים ב-NEXT-HOP של routeNEXT-HOP של routeNEXT-HOP. הוא כותבת ה-IP של הראותר שממנו מתחלים את הדרך ב-AS.

למשל אם ה-IP של הראותר הוא 111.99.86.55, אז הראותר ישתמש באלגוריתם OSPF כדי למצוא את הדרך הקצרה ביותר מ-1c-1a-1d-2a.



שלב 2:

הראוטר מגדר פורט ל-route ביחד עם הדריך הקצחה שהתקבלה ב-OSPF, החל מהראוטר עצמו ל-NEXT-HOP.

שלב 3:

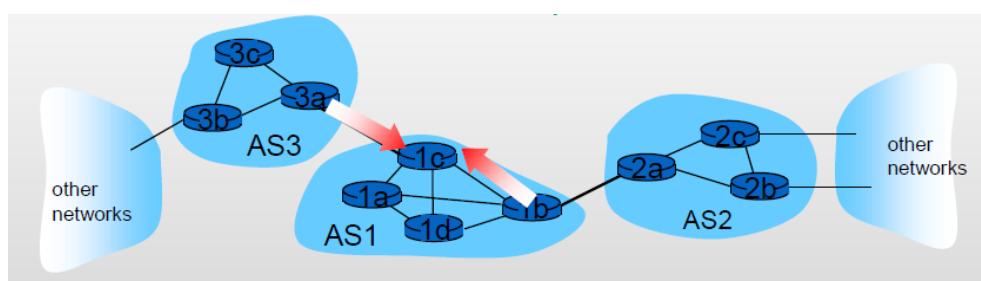
מכניסים כניסה של port (prefix) לטבלת הניתוב. למשל: (4 .138.16.64.0/22, port).

שכירת שווין: Hot Potato Routing

נניח שיש שתיים או יותר דרכי הći טובות ל-AS-inter, אז בוחרים את הדרך עם ה-HOP-NEXT הקרוב ביותר. משתמשים באלגוריתם OSPF ובודקים את ה-HOPs NEXT עצם כדי לקבוע מה הוא ה-gateway הקרוב ביותר.

למשל: נניח שמתќבל שווין, מ-1c בוחרים את AS17 AS3 AS131 או ?AS2 AS17 AS3 AS131

תשובה: נבחר ב-AS3 AS131 בגלל שהוא יותר קרוב.



שכבה 1 (IP/2 (TCP/IP) - שכבת הקישור

שכבת הקישור מספקת את האמצעים כדי להעביר מידע בין אמצעים לקלים. מחברת בין פרוטוקולים של השכבהعلילונה לאמצעים הפיזיים.

שולחת הודעות בין צמתים סמוכים.

יכול להיות אחראית על זיהוי שגיאות ותיקונים בין צמתים סמוכים.

משתפת את ערוץ broadcast-h.

יש מספר עצום של חיבורים פיזיים שונים לרשתות, לכל אחד יש את ה-*method* (MAC) שלו. פקעת IP אחת יכולה להיארך ולהיפתח (encapsulated and decapsulated) הרבה דרך פרוטוקולים שונים של שכבת הקישור לפני שהיא מגיעה ליעדה.

פרוטוקולים נפוצים של שכבת הקישור:

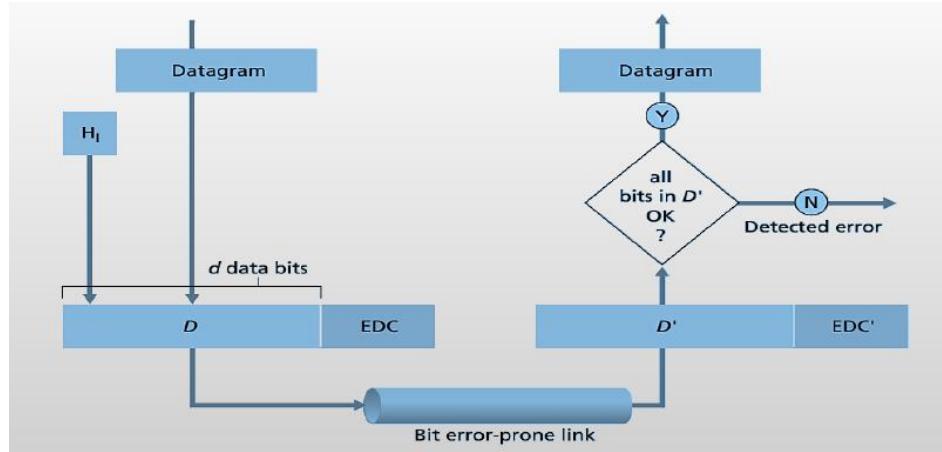
- Ethernet (802.3)
- WiFi (802.11)
- Point-to-Point protocol (PPP)

תת שכבות בשכבת הקישור:

- Logical Link Control (LLC): בקרת זרימה, הכרה (acknowledgement), התראות על שגיאות, multiplexing, כמו אפליקציה.
- Media Access Control (MAC): קביעה מי יכול לגשת למדיה בכל פעם, סנכרון פריימרים, ניתוב פיזי, אינטראקציית השירות (Quality of service).

זיהוי שגיאות ותיקונים

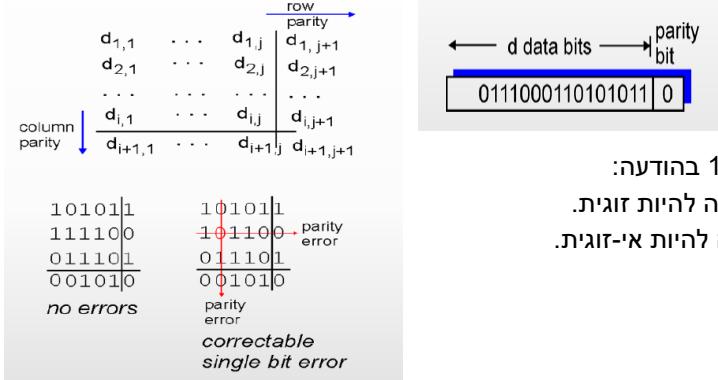
ניתן להוסיף אקסטרה ביטים לכל פריט כדי לחתם סכום (checksum) למטה שבמסגרת. הסכום יכול להיות מחושב אצל מקבל כדי לקבוע אם שלמות הודעה נשמרה.



זיהוי שגיאות הוא לא 100% אמין (יכול להיות איבוד מידע). ככל שנkazaה יותר ביטים ל-EDC (Correction) התוצאה תהיה יותר טובה. אבל זה מגע במחיר של גודל הפקטה וזמן החישוב.

:Parity Checking

הדרך פשוטה ביותר ל-EDC היא בדיקת parity. זה עולה רק ביט אחד, אומר אם החיבור של כל הביטים צריך להיות זוגי או אי-זוגי.



- ביט parity יחיד: מזהה שגיאות של ביט אחד.
- 2D parity (בשורה ועמודה): מזהה ומתקן שגיאות של ביט אחד.

ניתן להשתמש ב-parity זוגי או אי-זוגי. סופרים את מספר ה-1 בהודעה:

- Even parity – כל ההודעה, כולל הביט parity, צריכה להיות זוגית.
- Odd parity – כל ההודעה, כולל הביט parity, צריכה להיות אי-זוגית.

לדוגמא, עבור ההודעה: 11110011

- 111100110 :Even parity
- 111100111 :Odd parity

:Cyclic Redundancy Check
 רוב הפרוטוקולים בשכבה ה-2 משתמשים ב-CRC כדי לזיהות שגיאות.
 CRC משתמש בטכניקות שמאוד קלות לשימוש בחומרה.
 מזהה רצף שגיאות (burst errors) של עד $\geq h$ ביטים, כאשר h הוא אורך CRC. כלומר אם אורך ההודעה קטן מ- h הביטים של CRC, הוא יזהה את השגיאה.
 שגיאות שרוכות יותר מ- h הוא יזהה בהסתברות של (0.5^{h-1}).

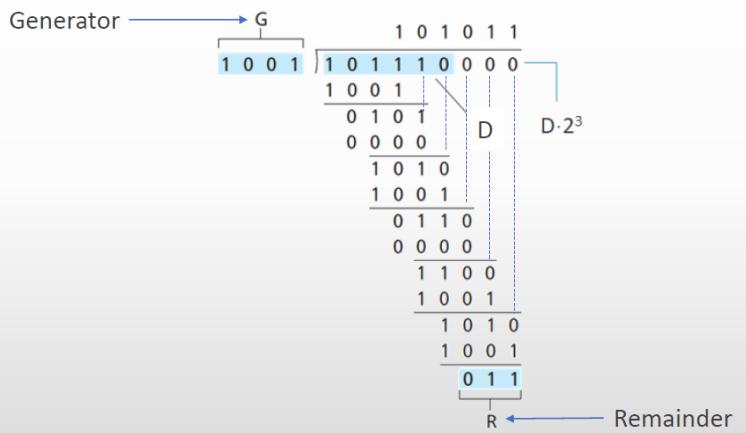
הסבר הרעיון הכללי של CRC בהקשר של מספרים דצימליים:

1. חשב על המידע שאתה רוצה להעביר בתור מספר.
2. חלק את המספר במספר ראשוני גדול (שידוע לשולח ולמקבל) וצרף את השארית לסוף המספר (המידע) ששולחים.

3. הצד מקבל יודע את המספר הראשוני. כל מה שהוא יעשה זה לחלק את המספר שהתקבל במספר הראשוני. ויראה אם השארית שהתקבלה זהה לו זו שקיבל.
4. אם השארית לא זהה, הtagلتה שגיאת. אחרת לא הtagلتה שגיאת.
5. הסתברות שהמגיד עובר עם שגיאות ועודין עבר את המבחן (כלומר לא הtagלו השגיאות), נהיה נמוך יותר ככל שהמספר הראשוני גדול יותר.

CRC Calculation Method

- **D** := The data to send
- **G** := The generator ($n + 1$ bits)
- Binary division in Galois Field (2) is used to determine the remainder (XOR)
- Remainder is appended to the original data and sent



פרוטוקולי MAC

באופן כללי ישנו שני סוגי של חיבורים:

- Point to point, משומש בין שני צמתים (nodes) -broadcast.
- אמצעי (רואטר) משותף (shared medium) בין כמה מארחים (כמו WiFi).

באמצעי משותף, צריך חוקים כדי לוודא שלכלום יש חזדנות שווה לשימוש באמצעי, והאמצעי משומש בצורה ייעילה. אם שני צמתים משדרים באותו הזמן, מקבל לא יכולlezחות אם אחד מהסיגנלים או שני הפרימרים נאבדו. תקשורת על ערוץ משותף צריכה להשתמש בערוץ עצמו.

סוגים של פרוטוקולי MAC:

- **חלוקת ערוץ (Channel Partitioning)**: מחלקים את הערוץ (חלוקת זמן, תדריות וכו'). כל צומת מקבלת חתיכת מהחלוקת לשימוש אקסлюסיבי.
- **גישה רנדומלית (Random Access)**: התנגשויות מותרכות, ההתקדמות היא על התאוששות מההתנגשויות.
- **לקיחת תורות (Taking Turns)**: צמתים לוקחים תורות, אבל צומת שיש לה יותר מהשלוח תיקח תור אחר.
- **ויתר**.

(Time Division Multiple Access) TDMA

eingשים לערוץ ב"סבבים" (rounds). כל צומת לוקחת סлот באורך קבוע (אורך=זמן שליחת הפקטה) בכל סבב. כל סлот שלא בשימוש נכנס למאובט idle.

לדוגמא: LAN של 6 צמתים, ל-4 יש פקטות, סлотים 2,5,6 הם idle.



יתרונות:

- מונע התנגשויות.
- הוגן: כל צומת מקבלת קצב של $sps N/R$ במשך כל זמן של פריים.

חסרונות:

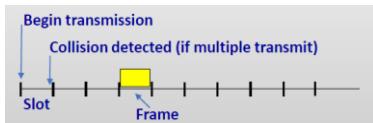
- צומת מוגבלת לקצב של $sps N/R$ גם אם רק לאחת יש פקודות לשולח.
- צמתים חייבות תמיד להמתין לתור שלהם אפילו רק אם צומת אחת צריכה לשולח פקודות.

Random Access Protocols

כאשר לצומת יש פקודה לשולח, שלוhim בקצב המלא של הערז, קצב R. אין תיאום בין הצמתים. אם יש שתיים או יותר צמתים שմשדרות בו זמן, יש "התנגשות". אם מזינה התנגשויות, שלוhim את הפקודות מחדש. פרוטוקול MAC random access מצין: מתי לשולח את הפקודות, איך לזהות התנגשויות, איך להתאושש מהתנגשויות (למשל ע"י דילוי באופן רנדומלי לשילוח מחדש).

Slotted ALOHA

הנחות:



- לכל הפריים יש את אותו הגודל.
- הזמן מחולק לסלוטים בגודל שווה (זמן לשילוח פריים אחד).
- צמתים מסונכרנים (יודעים מתי סлот מתחילה).
- צמתים מתחילה לשולח רק בתחילת כל סлот.
- אם 2 או יותר צמתים שלוhim באותו סлот, כל הצמתים המעורבים מזינים התנגשות לפני סוף הסлот.

אופן הפעולה:

כאשר צומת לוקחת פריים חדש, היא תשליח בסלוט הבא. אם אין התנגשויות - צומת יוכלה לשולח את הפריים החדש בסלוט הבא (אם יש לה משחו לשולח). אם יש התנגשויות - צומת מזינה באותו הסלוט ושולחת מחדש את אותו הפריים בכל סלוט עוקב עם הסתברותם (1<2). עד שהיא מצליחה.

יתרונות:

- צומת פעילה יחידה יכולה ברצף לשולח ב-rate full של הערז.
- מאוד מבוזר – הדבר היחיד שנחוץ הוא שלוטים בצתמים יהיו מסונכרנים.
- פשוט לימוש.

חסרונות:

- יש התנגשויות, שלוטים מבוזרים.
- שלוטים במצב idle.
- צמתים צריכים לזריז התנגשויות בפחות מהזמן שלוקח לשולח את הפקודה.
- מסונכרן לפי שעון.

Carrier Sense Multiple Access

ה-MAC שימוש ע"י רשתות WiFi-Ethernet. בודק אם מישו כבר משדר, אם לא אז משדרים, אחרת מוחכים זמן רנדומלי ושולחים שוב.

- Ethernet :CSMA Collision Detect – CSMA/CD ומנסים שוב. משתמשים ב- backoff אקספוננציאלי, מתחילה עם דילוי קטן שעולה בכל פעם.
- WiFi: CSMA Collision Avoidance – CSMA/CA לפניה של השילוח, מכשיר צריך לשלוח התראה שהוא מתכוון להתחיל לשולח. קשה לזרחות בשיטה זו אם מישהו אחר משדר בגל החזק של שידור מקומי.

מייען פיזי – Physical Addressing

כתובת MAC – 48 ביטים (6 בתים). נכתבם בהקסה-דצימלי.
לדוגמא: 1A-2F-BB-76-09-AD (ניתן נכתב גם: AD:09:76:BB:2F:1A).
צروب לתוך ה-ROM של כרטיס רשת (NIC), לעיתים מוגדר בתוכנה.
לכל NIC ב-LAN יש כתובת MAC ייחודית.

לכתובת MAC יש שני מרכיבים:

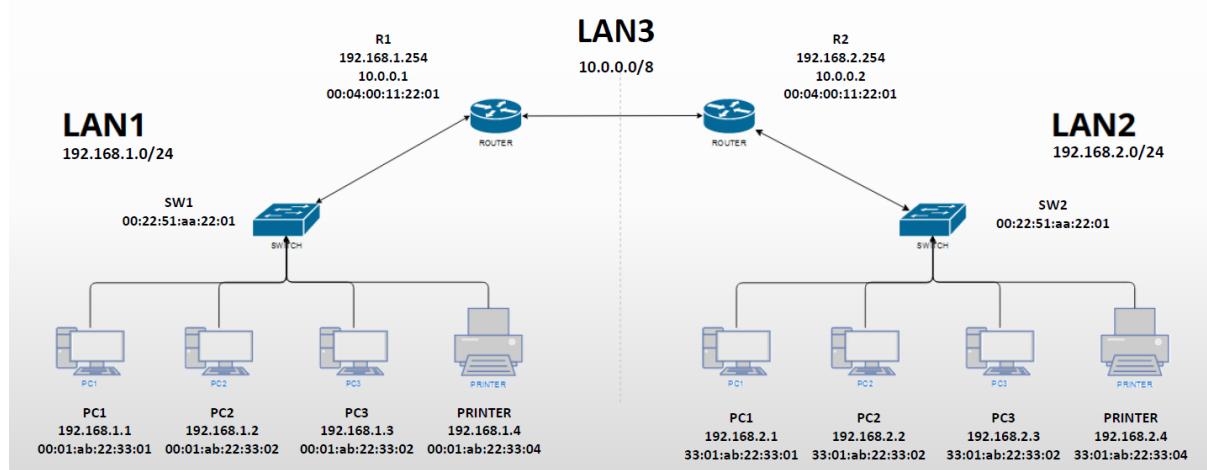
- Vendor (3 bytes)
- Device (3 bytes)

מנוהל ע"י IEEE. ל-NIC יש אותה כתובת לא משנה מאיזה מקום בעולם הוא מחובר.

סוויצ'רים – Switches

חומרה של שכבה 2. מתחזק טבלה של כתובות MAC ולאיזה פורטים הם מחוברים. כשהוא מקבל פריטים, בודק את כתובת ה-MAC ושולח אותה לפורט המתאים. אם ה-MAC לא ידוע, שולחים את הפריטים לכל הפורטים. מארחים לא מודעים ל-sw, הוא שוקף בשבילים. סוויץ' "מטומטם", שולח את כל הפקודות לכל הפורטים תמיד.

Fully Configured Two-LAN Network



Ethernet

Field	Bytes
Preamble	8
Addresses	12
Type	2
Data	1500
CRC	4
Total	1526



- 7 בתים עם תבנית 10101010 שאריהם מגיע בית אחד עם תבנית 11010110. משומש כדי לסנכרן את המקלט או מקלטים לקצב השעון של השולח.
- Type: 2 בתים, מסמן פרוטוקול של שכבה גבוהה יותר (בד"כ IP או IP אבל גם אחרים אפשריים).
- Data: 1500-46 בתים. אם ה-IP datagram עולה על 1500 בתים, אז השולח חיבר לחלק את הפקטה. אם יש פחות מ-46 בתים, השדה " ממולא" (stuffed) כדי למלא את הכל. ה-header של IP מכיל את האורך.
- CRC: 4 בתים, בדיקת שארית מחזוריית אצל המקלט (CRC-32). אם מזוהה שגיאה, הפריים מופל.

תכונות של Ethernet

- Connectionless: אין צורך בלחיצת יד לפני שליחת וקבלת NICs.
- לא אמין:
 - קבלת NIC לא שולחת ACKs או NACKs לשולח ה-CNIC.
 - אם CRC נכשל, הפריים מופל באופן שקט, ככלומר לא מתקבלת שום אינדיקציה על כך.
 - מידע בפרייםים שנפלו משוחזר רק אם השולח משתמש באמצעי שחזור של שכבות גבוהות יותר (למשל TCP), אחרת המידע נאבד לגמרי.
- פשוט וдол.
- ניתן להשתמש באמצעים פיזיים שונים: Fiber (סוגים שונים), כבילים (CAT5, CAT6 וכו').
- מהירותיות שונות.
- הכל מוגדר תחת IEEE802.3 סטנדרטי.

(Address Resolution Protocol) ARP

איך ניתן לקבוע את כתובת ה-MAC בהתבסס על כתובת IP? רק כתובות IP? משתמש ב-ARP.
כל צומת ברשות בשכבה 2 יש טבלת ARP. הטבלה ממפה כתובת IP לכתובות MAC.
ARP מלא את הטבלה, עובד רק באותו ה-LAN.

- שולח A רוצה לשולח מידע ל-B. A יודע רק את כתובת IP של B (בתוך אותה תת-רשת, "локלי").
- בשביל שהמידע יעבור ל-B, A חייב לדעת את כתובת ה-MAC של B כדי לעטוף (encapsulate) מידע של IP בתוך פריים של שכבת הקישור.
- ה-MAC של B לא נמצא בטבלת ARP של A.
- A שולח ב-broadcast פקטה של שאלת ARP, המכילה את כתובת IP של B (זה-IP של A עצמו):
 - source MAC address = A's MAC address
 - dest MAC address = FF-FF-FF-FF-FF-FF (MAC broadcast address)
 כל הצמתים ב-LAN מקבלים את שאלת ARP.
- B מקבל את ARP, רואה את IP שלו ועונה לכתובת MAC של A עם כתובת MAC של עצמו. נשלח פריים לכתובת MAC של A.
- צומת A שומרת בזיכרון (caches) את צמד ה-IP לכתובת MAC בטבלה ARP שלו (גם B שומר בטבלה שלו).
- עד שהמידע נהיה ישן (times out). אם הצמד בטבלה נהיה ישן ונדרש שוב, חוזרים על תהליך ARP.

ARP הוא "plug and play" – צמתים יוצרים את טבלאות ARP שלהם מלב' התערבות של מנהל הרשת או השרת. צמתים לא צריכים בטבלת ARP כניסה בשבייל כל שאר הצמתים בתת-הרשת, רק את אלו שאנו שוארים לתקשר איתם.

(Virtual LANs) VLANs

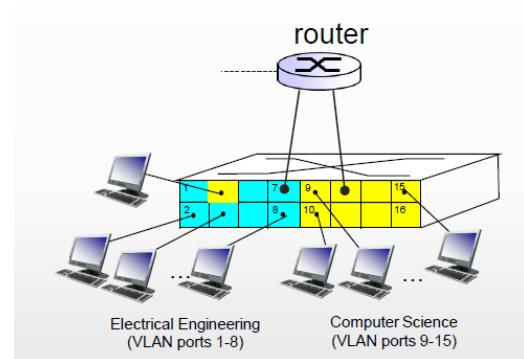
דומין broadcast יחיד: כל תבעורת broadcast של שכבה 2 (ARP, DHCP, unknown location of destination MAC address) יונן בעיות אבטחה/פרטיות ובעיות ייעילות.

.Virtual Local Area Network = VLAN אפשר לקנוף (configure) סוייצים שתומכים ביכולות VLAN כדי להגדיר כמה "virtual LANs" מעל תשתיית LAN יחידה.

VLAN מבוסס פורט: פורטים של סוייצים מקובצים (ע"י תוכנת ניהול של הסוייז) כר' שסוייז פיזי היחיד מתפרק ככמה סוייצים וירטואליים.

VLAN מבוסס פורט:

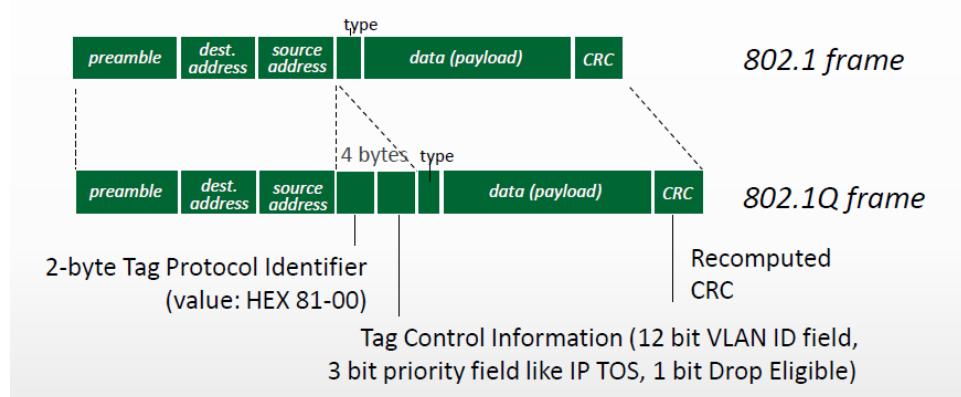
- בידוד תעבורת פרימיום אל או מפורטים 8-1 יכולם להגעה רק לפורטים 1-8. ה-ARP, DHCP וכו' מגודר רק ל-VLAN.
- Dynamic membership: פורטים יכולים להיות מוקצבים באופן דינמי בין VLANs.
- העברות בין VLANs: נעשה ע"י ניתוב (כמו עם סוייצים נפרדים). בפועל ספקים מוכרים רואוטרים משולבים עם סוייצים.



:VLANs spanning multiple switches

פורט Trunk: מעביר פרימיום בין VLANs שמנדרים בכמה סוייצים פיזיים. פרימים המועברים בתחום ה-VLAN בין סוייצים לא יכולים להיות פרימים פשוטים של Ethernet (כי הם חיברים להכלי מידע על ה-ID VLAN). פרוטוקול IEEE802.1Q מօיף/מוריד שדות נוספים ב-header בשבייל פרימים שמועברים ב-ports.

פורמט פרימ של 802.1Q VLAN



תג של VLAN נוספים ע"י הסויץ' בצד השולח של ה-trunk, מפורק (parsed), ומושר ע"י הסויץ' בצד המתקבל של ה-.trunk

דרכים נוספות להגדת VLANs:
ה-VLANs המפורטים מעלה הם VLANs מבוססי פורט. ניתן להגדיר VLANs מבוססי MAC.
הADMIN מגדיר סט של כתובות MAC ששויות לכל VLAN. בכל פעם שמקשיר מתחבר לפורט, הפורט מחובר ל-VLAN מבוססי MAC המתאים למקשיר.