בס"ד

פתרון שאלות בתקשורת עם הסבר לפתרון

נתונה טבלת ההעברה (forwarding table) הבאה הנמצאת בתוך נתב A.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

אם מגיעה לנתב A הודעה שהיעד שלה הוא הכתובת:

11001000 00010111 00011011 10001111

על איזה יציאה תצא ההודעה?

- 3
- 0
- ✔ 2
- 1

נתונה טבלת ההעברה (forwarding table) הבאה הנמצאת בתוך נתב A:

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

אם מגיעה לנתב A הודעה שהיעד שלה הוא הכתובת :

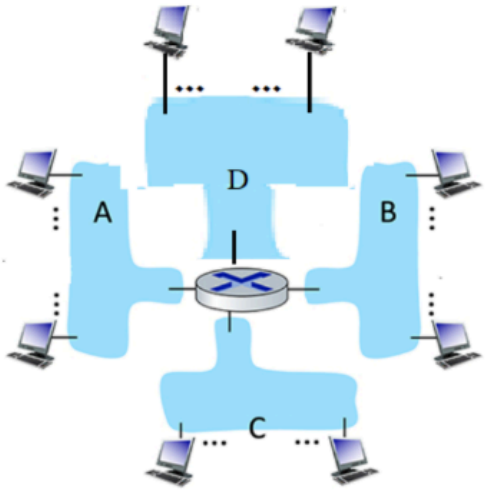
11001000 00010111 00010110 11100100

על איזה יציאה תצא ההודעה?

- 1
- 3
- 2
- ✔ 0

נתונה הרשת הבאה:

129.119.160.0

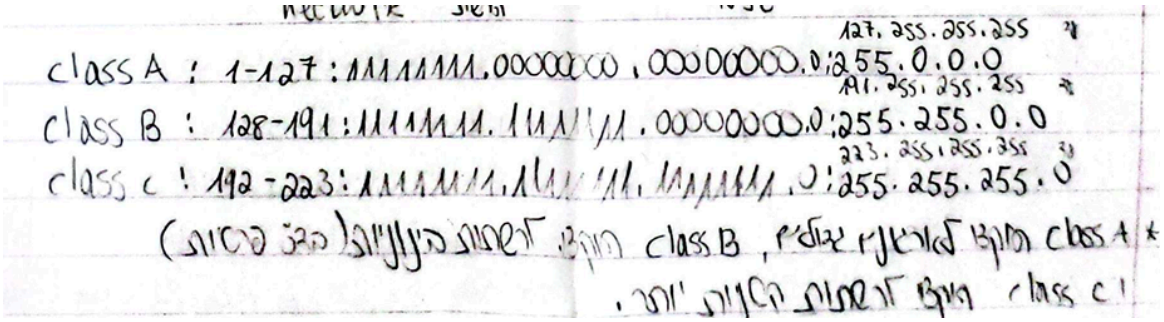


אם ידוע כי רשת זו נקנתה בתקופה של CLASSFUL, לאיזה class היא שייכת?

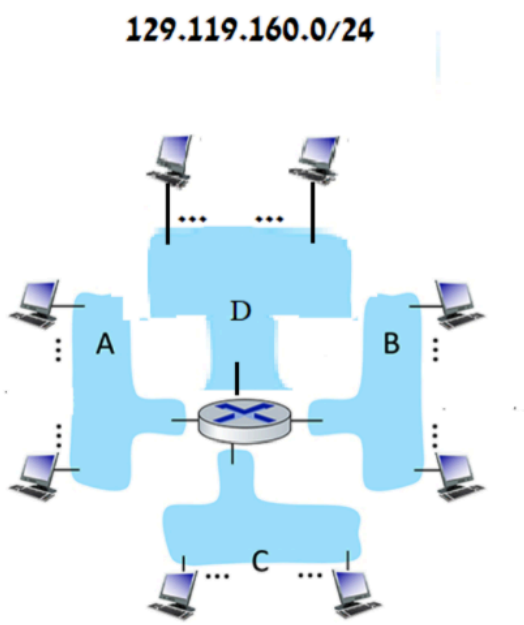
- [ ] A
- [x] B
- [ ] E
- [ ] D
- [ ] אי אפשר לדעת מנתונים אלו
- [ ] C

הסבר:

כתובת הרשת מתחילה ב129 שזו כתובת ששיכת לClass B



נתונה הרשת הבאה:



כמה **כתובות שונות** יש ברשת הגדולה - **<u>לפני שמחלקים אותה לתתי-רשתות</u>**.

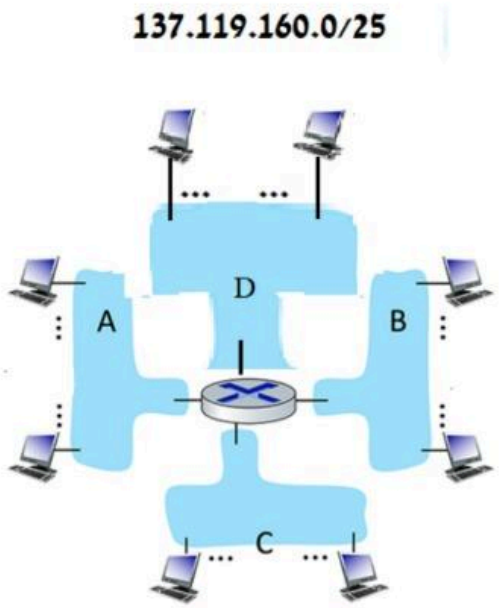
2^8 ✔

הסבר:

<u>מספר מקסימלי של **כתובות**: (מסכה-32)^2</u>

אצלינו המסכה בגדול 24 קיבלנו (8=24-32)^2 , כלומר 2^8

נתונה הרשת הבאה:



החליטו לחלק הרשת ל-4 תתי רשתות, כל תת-רשת בגודל שווה.

A הוא תת-הרשת הראשונה.

B הוא תת-הרשת השנייה.

C הוא תת-הרשת השלישית.

D הוא תת-הרשת הרביעית.

מה כתובת הרשת של תת-הרשת **B**?

- ☐ 137.119.160.128
- ☐ 137.119.161.0
- ☐ 137.119.160.64
- ☐ 137.119.161.128
- ☑ 137.119.160.32 ✔

הסבר:

<u>מספר מקסימלי של **כתובות**: (מסכה-32)^2</u>

אצלינו המסכה בגדול 24 קיבלנו (7=25-32)^2 , כלומר 2^7

מכוון שמחלקים ל4 תתי רשתות קיבלנו 2^5

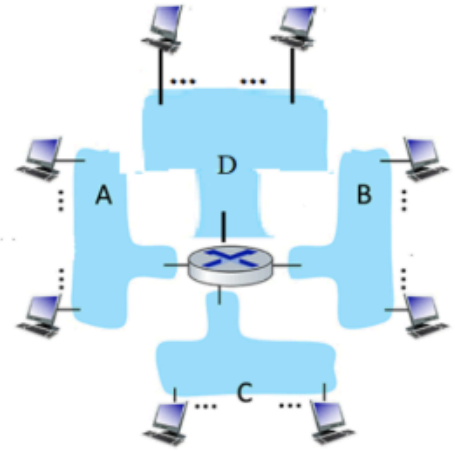
ז"א שכל תת רשת מקבלת 32 כתובות

A מקבלת מ0-31

B מקבלת מ32-64 וכו.

נתונה הרשת הבאה:

129.119.160.0/24



החליטו לחלק את הרשת ל-4 תתי רשתות בגדלים שווים.

A הוא תת-הרשת הראשונה.

B הוא תת-הרשת השנייה.

C הוא תת-הרשת השלישית.

D הוא תת-הרשת הרביעית.

מהו המספר המקסימאלי של **הוסטים** שונים שאפשר לשים בכל תת רשת?

- 2^6
- ✔ $2^6 - 2$
- 2^8
- 2^7
- $2^8 - 2$
- $2^7 - 2$

הסבר:

<u>מספר מקסימלי של **הוסטים**: 2-(מסכה-32)^2</u>

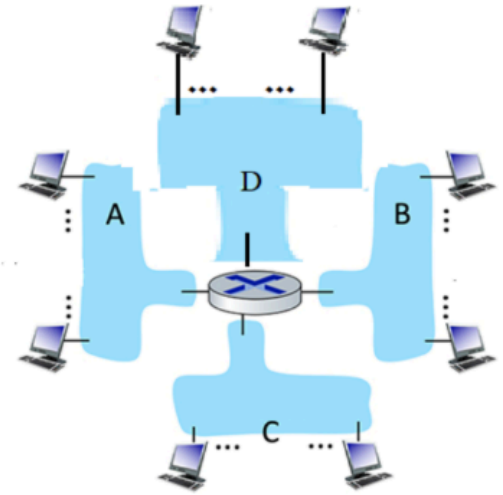
אצלינו המסכה בגדול 24 קיבלנו 2-(8=24-32)^2

מכון שהחליטו לחלק את הרשת ל4 תתי רשתות לכן צריך לחלק ב4.

סה"כ 2-6^2

נתונה הרשת הבאה:

137.119.160.0/25



החליטו לחלק את הרשת ל-4 תתי רשתות, בגדלים שווים.

A הוא תת-הרשת הראשונה.

B הוא תת-הרשת השנייה.

C הוא תת-הרשת השלישית.

D הוא תת-הרשת הרביעית.

מה כתובת ה- **broadcast** של תת-רשת **C**?

- 137.119.161.255
- 137.119.160.255
- ✔ 137.119.160.95
- 137.119.160.191
- 137.119.160.128

הסבר:

בכתובת שידור- ברודקאסט ניקח את כתובת הIP והמסכה, איפה שהביטים דלוקים במסכה נשאיר את הביטים האלו ככתובת הIP ואיפה שהביטים כבויים במסכה נדליק אותם. כך מחשבים:

הכתובת: 137.119.160.0/25

מבצעים המרה לבינארית: 10001001.01110111.10100000.00000000

המסכה: 11111111.11111111.11111111.10000000

לאחר הדלקת הביטים הכבויים: 10001001.01110111.10100000.01111111

המרה בחזרה לעשרוני: 137.119.160.127

עכשיו מכוון שנתון שחילקו את הרשת ל4 תתי רשתות, לכן את הביטים של ההוסט (הימנים ביותר לאחר הנקודה) מחלקים ב4 כך ש:

תת הרשת הראשונה היא 0-31

תת הרשת הראשונה היא 32-63

וכן הלאה כך שקיבלנו שתת הרשת C היא .95 (64-95)

נתון 2 שורות הבאות:

01000110 10110100
01101111 01001000

שימו לב: לכל שורה יש 16 ביטים - 8 הביטים בכל שורה מופרדת על ידי רווח.

מהו ה-checksum עבור 2 שורות האילו

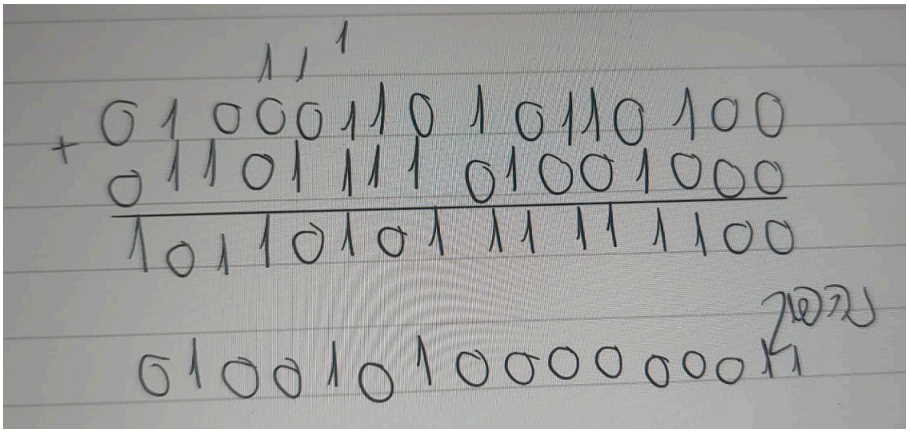
(בתשובה שלך, יש לשים רווח בין 8 ביטים הראשונים לבין 8 ביטים האחרונים)

- [x] 01001010 00000011
- [ ] 01001010 00011000
- [ ] 11111111 00000011
- [ ] 11001011 11000011

דרך הפתרון:

1. סוכמים את הביטים.
2. אם יש גלישה מוסיפים את ה 1 למספר שהתקבל.
3. הופכים את הביטים.

כך:



יש להתבונן באיור שלהלן, המתווה את התפתחות congestion window (חלון העומס) של TCP בתחילת כל יחידת זמן (כאשר יחידת הזמן שווה ל-RTT)

במודל המופשט לבעיה זו, TCP שולח פרץ של מנות (packet) בגודל החלון cwnd בתחילת כל יחידת זמן. התוצאה של שליחת פרץ החבילות היא או

א) כולם מקבלים ACK בתום הRTT, או

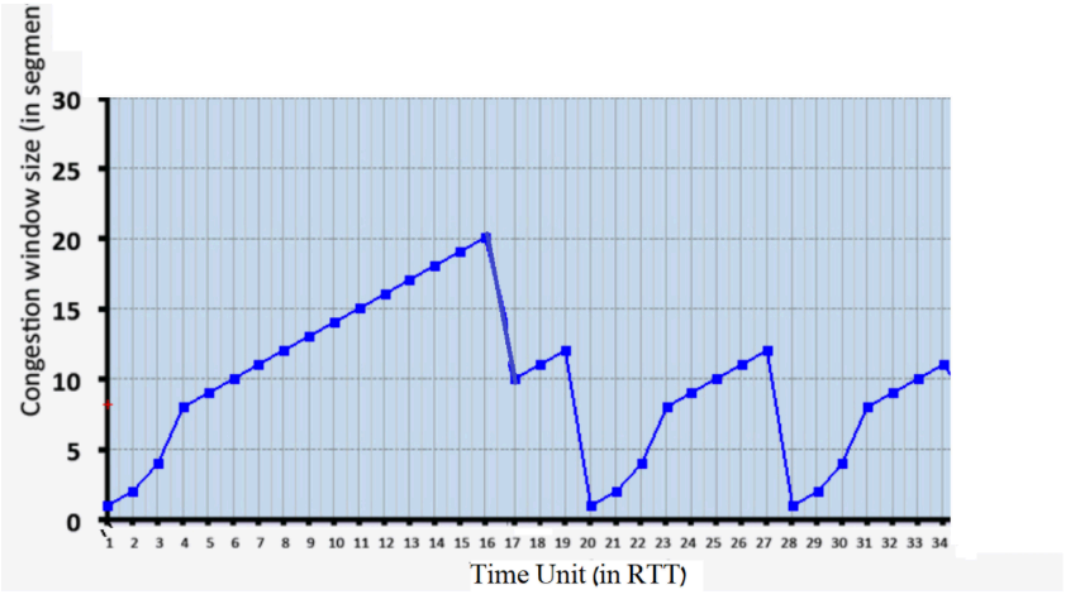
ב) שיש timeout על החבילה הראשונה, או

ג) מקבלים 3DUP ACK על החבילה הראשונה.

האיור להלן מתווה כמה מנות נשלחו בכל יחידת זמן.

במהלך איזה טווח הזמן הבאים TCP נמצא בשלב של **Congestion Avoidance**?

שימו לב: יכול להיות יותר מתשובה אחת נכונה. יש לבחור את כל התשובות הנכונות.



- [20,28]
- ✔ [4,16]
- [16,17]
- [1,4]
- [1,16]
- [27,28]
- ✔ [23,27]
- [20,23]

הסבר:

בשלב זה, גודל חלון הגודש (cwnd) גדל בצורה **לינארית ואיטית**, בניגוד לשלב Slow Start שבו הוא גדל מהר. המטרה היא **להימנע מעומס ברשת** על ידי בדיקה הדרגתית של כמה תעבורה הרשת מסוגלת להכיל מבלי לגרום לאובדן מנות. TCP נכנס לשלב הזה כאשר cwnd מגיע לערך הסף ssthresh. אם מתרחש אובדן מנות, TCP חוזר לשלב קודם (Slow Start או Fast Recovery).

בכל סיבוב RTT מוצלח (כלומר, כל המנות הגיעו ללא אובדן), גודל החלון גדל בערך ב-**MSS אחד**

יש להתבונן באיור שלהלן, המתווה את התפתחות congestion window (חלון העומס) של TCP בתחילת כל יחידת זמן (כאשר יחידת הזמן שווה ל-RTT)

במודל המופשט לבעיה זו, TCP שולח פרץ של מנות (packet) בגודל החלון cwnd בתחילת כל יחידת זמן. התוצאה של שליחת פרץ החבילות היא או

א) כולם מקבלים ACK בתום הRTT, או

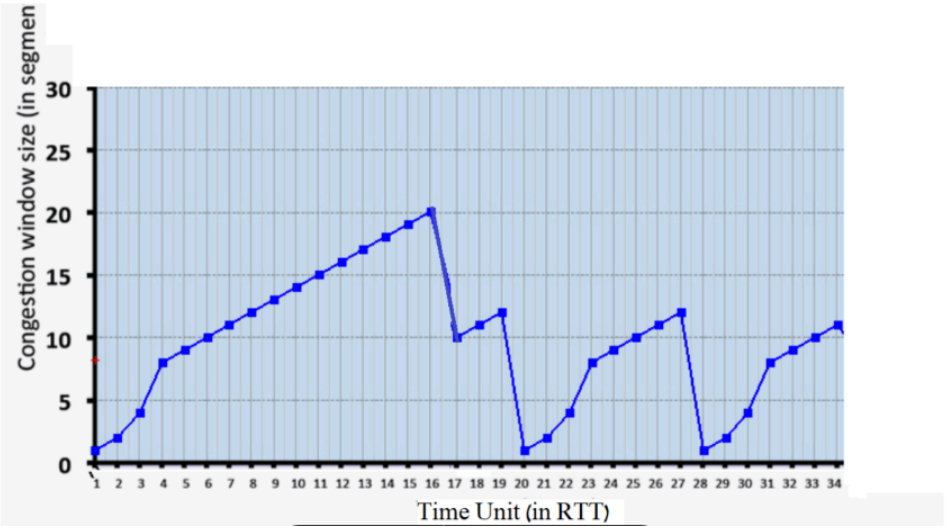
ב) שיש timeout על החבילה הראשונה, או

ג) מקבלים 3DUP ACK על החבילה הראשונה.

האיור להלן מתווה כמה מנות נשלחו בכל יחידת זמן.

במהלך איזה טווח הזמן הבאים TCP נמצא בשלב של **Slow Start**?

שימו לב: יכול להיות יותר מתשובה אחת נכונה. יש לבחור את כל התשובות הנכונות.

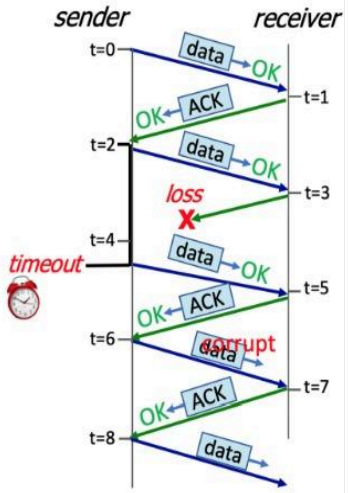


- [4,16]
- [1,16]
- [27,28]
- [16,17]
- ✔ [20,23]
- ✔ [1,4]
- [20,27]
- [20,28]

הסבר:

בשלב Slow Start של TCP, גודל חלון הגודש (cwnd) גדל **באופן אקספוננציאלי** – הוא מכפיל את עצמו בכל סיבוב RTT, כדי לנצל במהירות את רוחב הפס הזמין.

השלב נמשך עד שה-cwnd מגיע לערך הסף (**ssthresh**) או עד שמתרחש **אובדן מנות**, ואז TCP עובר לשלב Congestion Avoidance או מתחיל מחדש. מטרת השלב היא לבדוק כמה נתונים ניתן לשלוח מבלי לגרום לעומס ברשת.



יש להתבונן באיור שלהלן המראה העברת הודעות וACK-ים בין sender (משדר) לבין receiver (מקלט)

כך שה-sender שולח הודעות וה-receiver שולח ACK-ים

כל הודעה ו-ACK יוצאות תקינות אבל יכולות להשתבש בדרך.

הודעה/ACK שהתקבל בתקינות יהיה מסומן בירוק עם התווית OK

הודעה/ACK שהתקבל עם שגיאה מסומן באדום עם התווית corrupt

התעבורה עובדת לפי שיטת Stop and Wait - כך שכל הודעה מסומנת ב-0 או 1 ואחרי שה-sender שולח הודעה הוא מחכה ל-ACK לפני שליחת הודעה הבאה.

האיור הבא מראה שיחה בין sender ל-receiver ומסמנת 8 נקודות שונות בזמן כל יחידת זמן מסומנת על יד t = x כך ש- x הוא מספר.

רוצים לדעת מה היה ה-sequence number (מספר סידורי) של ההודעה/ACK בזמנים t = 4, t=5 t= 6 t=7

כל המספרים מופיעים משמאל לימין

לדוגמה עבור t = 0, t = 1

התשובה הייתה:

0 ,0

0 הראשון (שמאלי) כי ההודעה יצאה עם sequence number של 0

0 השני (הימיני) כי מספר ה-ACK היה 0

- 1, 1, 0, 0
- ✔ 1, 1, 0, 1
- 1, 1, 1, 0
- 0, 0, 1, 0
- 1, 1, 1, 1

הסבר:

האלגוריתם **Stop-and-Wait** הוא פרוטוקול תקשורת פשוט המשמש להעברת נתונים בצורה אמינה בין שני צדדים.

הפרוטוקול מבוסס על הרעיון שהשולח שולח חבילה אחת בלבד בכל פעם, ומחכה לאישור (ACK) מהמקבל לפני שהוא שולח את החבילה הבאה.

שלבי הפעולה של Stop-and-Wait:

1. **השולח שולח חבילת מידע (Packet)** למקבל.
2. **השולח עוצר וממתין** לקבלת אישור (ACK) מהמקבל.
3. כשהמקבל מקבל את החבילה בהצלחה, הוא שולח **אישור חזרה** לשולח.
4. רק לאחר קבלת האישור, השולח שולח את החבילה הבאה.
5. אם השולח **לא מקבל אישור בזמן סביר (Timeout)**, הוא מניח שהחבילה או האישור אבדו – ושולח שוב את אותה החבילה.

בדוגמא שלנו:

בזמן t=0 שלחנו הודעה ראשונה עם מספר סידורי 0

בזמן t=1 שלחנו ACK עם מספר סידורי 0 על ההודעה הראשונה שהתקבלה בהצלחה.

בזמן t=2 שלחנו הודעה שניה עם מספר סידורי 1

בזמן t=3 שלחנו ACK עם מספר סידורי 1 שאבד בדרך.

בזמן t=4 נגמר הTimeout ולכן שלחנו שוב את אותו המידע של ההודעה השניה עם מספר סידורי של 1.

בזמן t=5 שלחנו ACK עם מספר סידורי 1 שההודעה התקבלה בהצלחה.

בזמן t=6 שלחנו הודעה שלישית עם מספר סידורי 0 רק שהמידע לא התקבל.

בזמן t=7 בגלל שהמידע האחרון לא התקבל אז שלחנו שוב ACK עם המספר סידורי האחרון ששלחנו שהוא 1. לכן קיבלנו סה"כ 1,1,0,1.

התבוננו על האיור הבא:

TCP sender שולח 8 סגמנטי TCP בזמנים $t = 1, 2, 3, 4, 5, 6, 7, 8$.

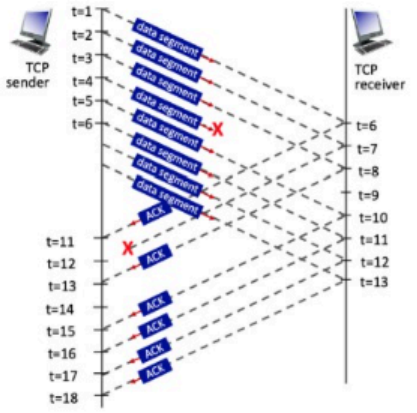
נניח שה-sequence number (מספר הסידורי) של הסגמנט הראשון הוא 0. השהייה בין שולח (TCP sender) ומקלט (TCP receiver) הוא 5 יחידות זמן, ולכן הסגמנט הראשון יגיע למקלט ב t=6

ה-ACK-ים שהמקלט שלח בזמנים $t = 6, 7, 8, 10, 11, 12$ מתואר באיור

אפשר להניח שאחרי זמן 11 אין לשולח עוד סגמנטים לשלוח

הסגמנט שנשלח ב t=4 הלך לאיבוד

גם ה-ACK שנשלח ב t=7 הלך לאיבוד



מהו ה-sequence number (מספר סידורי) של הסגמנט שנשלח ב t=5?

2
4 ✔
5
3

הסבר:
כאן אנחנו שולחים כמה הודעות ברצף ורק לאחר מכן מחכים לקבלת כל האישורים.
בזמן t=5 אנחנו עדין בשליחת DATA כלומר עדין לא נתגלה שאבד המידע ולכן אע"פ שהסגמנט שנשלח בזמן t=4 נאבד אנחנו עדין לא יודעים זאת.
מכון שהמספר הסידורי של הסגמנטים ממוספר מ0 לכן המספר הסידורי של הסגמנט שנשלח בזמן t=5 הוא 4.

נתונה הרשת הבאה:

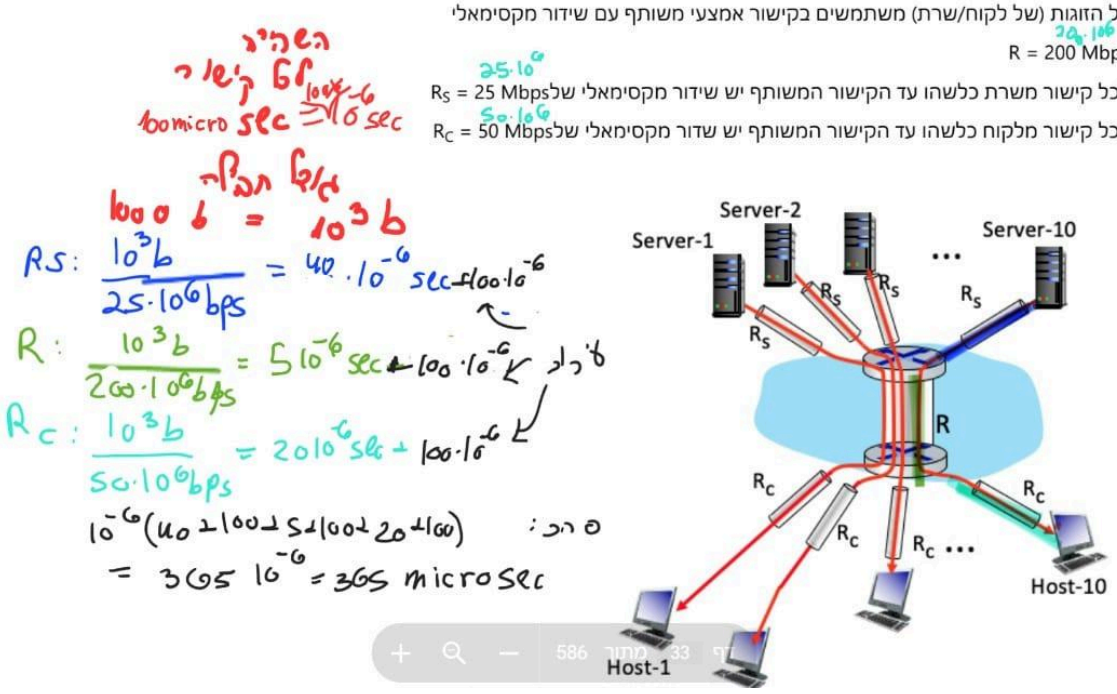
10 שרתים שונים (בתצוגה רואים רק 4 מתוך ה- 10), מחוברים ל- 10 לקוחות שונים על ידי 10 מסלולים שונים כך שבכל מסלול יש 3 קישורים.

כל הזוגות (של לקוח/שרת) משתמשים בקישור אמצעי משותף עם שידור מקסימאלי

R = 200 Mbps

לכל קישור משרת כלשהו עד הקישור המשותף יש שידור מקסימאלי של R_S = 25 Mbps

לכל קישור מלקוח כלשהו עד הקישור המשותף יש שדור מקסימאלי של R_C = 50 Mbps



נתבונן עכשיו על המסלול של שלשת הקישורים בין שרת 10 (Server 10) ללקוח 10 (Host 10) עם ערכים עבור R לפי המתואר למעלה. אפשר להניח שגודל הפאקט הוא 1,000 ביטים ולכל קישור יש השהייה של 100 מיקרושניות (תזכורת 1 מיקרושנייה הוא = 0.000001 שניות

(1 microsecond = 0.000001 seconds

אפשר להניח שהעיכובים האחרים (של התור ושל עיבוד צומת) הם זניחים. כמה זמן ייקח מהרגע שהשרת מתחיל לשדר עד שהלקוח יקבל את הביט האחרון של הפאקט האחרון אם ידוע שהרשת יחסית שקטה וכששרת 10 משתמש בקישור המשותף R אין אף מכשיר אחר שמשתמש בקישור בו זמני?

- 350 microseconds
- 555 microseconds
- 325 microseconds
- 450 microseconds
- ✔ 365 microseconds
- 735 microseconds

הסבר:

-נתון גודל הפאקט הוא 1000 ביטים. כלומר 10^3

-לכל קישור יש השהיה של 100 מיקושניות. כלומר 100^-6

-נוסחת החישוב: גודל חבילה חלקי שידור מקסימלי ומוסיפים את ההשהיה.

בס"ד

אוסף משפטים נכונים מתוך הבחנים:
*ההבדל בין forwarding (העברה) לבין routing (ניתוב):
-הכוונה של forwarding (העברה) היא להעביר datagrams (מנות) מהכניסה של הנתב ליציאה המתאימה.
-Forwarding (העברה) היא הפעולה הלוקאלית המעבירה פאקטים מהכניסה של הנתב ליציאה המתאימה בנתב ו-routing (ניתוב) היא
הפעולה הגלובלית המחשבת את המסלול בין מקור ליעד עבור פאקט.

*משתמשים בהודעות ICMP כדי להעביר הודעות הקשורות למצב של הרשת.
הודעות ICMP מועברות ישירות כ-payload בתוך datagram (מנה) של IP.

*ב-header (כותרת) של IP משתמשים בשדה version כדי לדעת האם משתמשים בפרוטוקול IPv4 או IPv6.

*ב-header (כותרת) של IP משתמשים בשדה TTL כדי למנוע את התופעה ש-datagram (מנה) תסתובב מספר פעמים אינסופי ברשת

*נניח שבנתב ב-buffer של פורט היציאה שלו יש מספיק מקום רק על מנת לשמור 3 מנות (packets), כולל המנה שהוא באמצע לשדר אותה.
ש. מה יקרה אם מנה 4 תגיע לפורט היציאה והמוצא כאשר ה-buffer הוא מלא?
ת. הוא יזרוק את המנה.

*משפטים נכונים לגבי כתובת IP:
-אם לנתב (router) יש יותר מממשק אחד אז יש לו יותר מכתובת IP אחת.
-אם להוסט (host) יש יותר מממשק אחד אז יש לו יותר מכתובת IP אחת.
-כתובת IP משויכת לממשק.

*בשיטה של N-Back-Go זורקים מנות (packets) שהגיעו בסדר לא נכון כי המימוש מצד המקלט (receiver) הוא יותר פשוט.

*ב-control flow (בקרת זרימה) שכבת התעבורה במחשב המקלט מודיע למחשב השולח את גודל חלון המקלט שלו

בס"ד

*עבור demultiplexing (ריבוי) נתונים ל- TCP socket נשתמש בשדות הבאות, הנמצאות ב-header (כותרת) של ה-segment (סגמנט):
כתובת IP מקור, כתובת IP יעד
כתובת PORT מקור, כתובת PORT יעד
ואף כתובת אחרת

*המימוש העיקרי של שכבת התעבורה (layer transport) הוא בקצה (edge) של הרשת (כלומר בהוסטים).

*מממשים את שכבה מספר 3 (שכבה של הרשת - network) גם בהוסטים (hosts) הנמצאים בקצה הרשת וגם בנתבים (routers) הנמצאים בליבת הרשת.

*הרכיבים נמצאים בקצה הרשת (edge network) אלו:server (שרת), client (לקוח).

*הגדרה "bolts and nuts" של האינטרנט. ההגדרות המתאימות למבט הזה:
-רשת של רשתות
-אוסף של המון רכיבים של מחשוב ומתגים שהם מחוברים על ידי קישורים.
-אוסף של חומרה ותוכנה המבצעים פרוטוקולים שהם מגדירים את הפורמט וסדר של הודעות הנשלחות בין 2 או יותר רכבים וגם כולל הפעולות שיתבצעו בזמן שידור/קליטה של הודעה או כל אירוע אחר.

*במבנה ה The peer-to-peer (P2P) paradigm כל מחשב יכול לבקש שירות וגם יכול לתת שירות.

*משפטים נכונים לגבי עוגיות (cookies):
עוגיות (cookies) משמשות כדי לשמור מידע על הלקוח ולהתגבר על העובדה ש-HTTP הוא **stateless** (ללא זיכרון מצב).
הלקוח אכן שולח את ה-cookie בתוך בקשת HTTP, אם קיים.
ה-cookies נשמרים במחשב של הלקוח (בדפדפן).
השרת גם שומר מידע מקביל שמתייחס ל-cookies (למשל session data).
-שומרים את הcookies במחשב של הלקוח
-הלקוח שולח את הcookie (במידה שקיים) בתוך הודעת בקשה
-משתמשים בcookies כדי להתגבר על הבעיה שHTTP הוא stateless (בלי מצב)

בס"ד

*שולח המשתמש בצינור (pipeline) יכול לשלוח הרבה מנות (packets) לפני שהוא מקבל ACK מהמקלט.

*כאשר משתמשים בצינור (pipeline) יכול להיות שנמצאים מנות (packets) שהם עדיין בתוך הערוץ שכבר נשלחו מהשולח אבל עוד לא הגיעו למקלט.

*TCP מבטיחה control flow (בקרת זרימה)

*הרכיבים הנמצאים בליבת הרשת (core network) הם: router (נתב), רשת ISP

*ל- control congestion (בקרת עומס) במידה שיש עומס ברשת, router (נתב) שתור שלו מלא, זורק הודעה. זה גורם לשכבת תעבורה של השולח (מחשב ששלח ההודעה שנזרקה) לקבל DUP-ACK עבור ההודעה שנזרקה ומזה השולח יכול להניח שיש עומס ברשת.