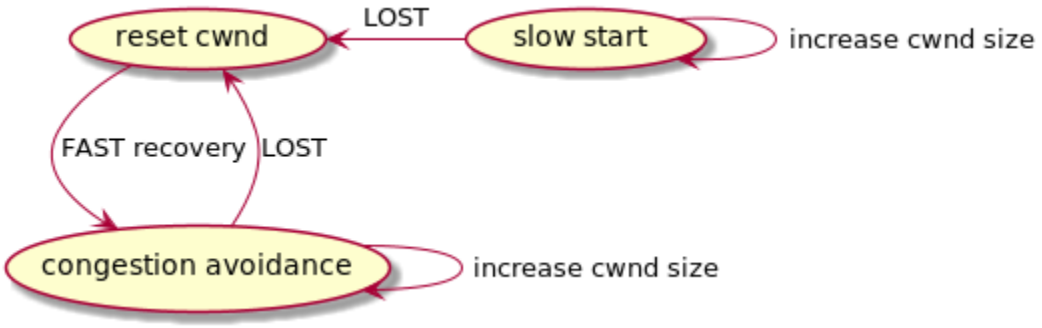
פרוייקט גמר
קורס תקשורת ומחשוב - 7028310
דניאל ציון 208640326
יאן נייגבבר 323906842

תשובות לשאלות של חלק ב:

- ציירו דיאגרמת מצבים בהם המערכת עובדת



- כיצד המערכת מתגברת על איבוד חבילות?

המערכת מתגברת על איבוד חבילות באמצעות שימוש בשלשה אלגוריתמים עיקריים: slow start, congestion avoidance, fast recovery, ובנוסף אליהם משלבת אלמנט של גודל חלון קבוע עד שמתקבלים ACKS על כל החבילות שנשלחו בחלון הזה.
המערכת מתחילה מ slow start בפעם הראשונה שהמערכת זיהתה שנאבדה חבילה, ה congestion window נחתך פי 2 ונקבע הss-threshold. לאחר מכן המערכת ממשיכה להגדיל את גודל החלון באופן לינארי באמצעות אלגוריתם Congestion avoidance אם אין איבוד חבילות. אם המערכת זיהתה שוב איבוד חבילות אז חותכים שוב את גודל החלון פי 2 ונקבע threshold חדש. וכך הלאה אבל לא חוזרים לslow start פעם נוספת. בנוסף המערכת שולחת באופן חוזרני וסלקטיבי את החבילות שלא התקבלו שוב לאחר זמן המתנה קבוע ל ACK.

- כיצד המערכת מתגברת על בעיית latency?

1) המערכת מפרקת את המידע לחבילות קטנות, 1024 בייתים כדי שזמן ההעברה שלהם יהיה יותר מהיר ברוחב פס קטן יותר ושולחת אותן.
2) המערכת מגיעה לשליחת כמות שליחת חבילות אופטימלית באמצעות אלגוריתם ה Congestion control.
3) המערכת מאפשר למספר לקוחות להוריד קבצים בו זמנית באמצעות threads ששולחים את החבילות ללקוחות באותו זמן. כך 2 לקוחות יכולים לקבל חבילות בו זמנית מהשרת במקום שאחד יוריד בזמן שהשני יחכה.
4) המערכת לא מחכה לקבל אישורים על כל חבילה בנפרד, אם המערכת קיבלה ACK על חבילה וגם גודל החלון מספיק גדול אז היא שולחת עוד חבילה, כלומר איננה תלויה בשאר החבילות שיגיעו, וזה מאפשר העמסה גדולה יותר של חבילות, וניצול מירבי של רוחב הפס.

חלק ג'

שאלה 1 תיאור הודעות שעוברות החל מהחיבור הראשוני למתג ועד שההודעה מתקבלת צד השני של הצ'אט:

הודעה ראשונה שולח המחשב 1m שהתחבר לרשת על מנת לקבל IP

MESSAGE: “Hello , any DHCP server available out there ? answer me if you hear me” TYPE: DHCP DISCOVER PROTOCOL: UDP **(Broadcast)		
	SRC	DEST
IP	0.0.0.0	255.255.255.255
PORT	68	67
MAC	m1	FF:FF:FF:FF:FF:FF

הודעה שנייה הנתב מחזיר הודעה (2m)

MESSAGE : “I can hear you , my IP address is 10.10.1.254 , i can allocate an IP address ‘10.10.1.5’ for you“ TYPE: DHCP OFFER PROTOCOL: UDP **(Broadcast)		
	SRC	DEST
IP	10.10.1.254	255.255.255.255
PORT	67	68
MAC	m2	FF:FF:FF:FF:FF:FF

הודעה שלישית המחשב 1m אל הנתב

MESSAGE : “Thank you for your response . then can you, at IP address 10.10.1.254 Allocate IP address for me” {Requested IP = 10.10.1.5”} TYPE: DHCP REQUEST PROTOCOL: UDP **(Broadcast)		
	SRC	DEST
IP	0.0.0.0	255.255.255.255
PORT	68	67

3

MESSAGE : “Thank you for your response . then can you, at IP address 10.10.1.254 Allocate IP address for me” {Requested IP = 10.10.1.5”} TYPE: DHCP REQUEST PROTOCOL: UDP **(Broadcast)		
	SRC	DEST
MAC	m1	FF:FF:FF:FF:FF:FF

הודעה רביעית בחזרה מהנתב

MESSAGE : “Sure i can allocate you , your IP address . allocate time 24hours” {Your IP= 10.10.1.5” , DNS = xxxx , allocate time =24 hours} TYPE: DHCP ACK PROTOCOL: UDP **(Broadcast)		
	SRC	DEST
IP	10.10.1.254	255.255.255.255
PORT	67	68
MAC	m2	FF:FF:FF:FF:FF:FF

כאשר מחשב נוסף יתחבר (4m) הוא יבצע את אותם הפעולות עד לקבלת IP

כאשר נרצה לשלוח הודעה ממחשב 1m ל 4m בהתחלה תתבצע לחיצת ידיים SYN ACK ולאחר מכן יפתח הקשר תועבר הודעה ותוצג על המסך

MESSAGE : SYNC TYPE: SYN PROTOCOL: TCP		
	SRC	DEST
IP	10.10.1.5	10.10.1.8
PORT	x	y
MAC	m1	m4

4

MESSAGE: SYN ACK TYPE: SYN ACK PROTOCOL: TCP		
	SRC	DEST
IP	10.10.1.8	10.10.1.5
PORT	y	x
MAC	m4	m1

MESSAGE: ACK TYPE: ACK PROTOCOL: TCP		
	SRC	DEST
IP	10.10.1.5	10.10.1.8
PORT	x	y
MAC	m1	m4

MESSAGE: PSH ACK {message = “something “} TYPE: PSH ACK PROTOCOL: TCP		
	SRC	DEST
IP	10.10.1.5	10.10.1.8
PORT	x	y
MAC	m1	m4

5

MESSAGE : ACK TYPE: ACK PROTOCOL: TCP		
	SRC	DEST
IP	10.10.1.8	10.10.1.5
PORT	y	x
MAC	m4	m1

שאלה 2 מה זה **CRC:**

CRC Cyclic redundancy check-במילים פשוטות, הוא אלגוריתם שמהווה גרסה נוספת לשיטת checksum בעת ביצוע העברת מידע, ניתן להשתמש ב CRC כדי לבדוק במהירות האם המידע שנשלח מהצד השולח הגיע בצורה תקינה לצד המקבל. האלגוריתם יוצר מניפולציה על הביטים של ה DATA שיוצר ערך CRC . כאשר כל שינוי קטן ב DATA ייצור לנו ערך שונה של CRC שבעזרתו ניתן לזהות ולאשר האם המידע שהגיע אינו תקין או האם המידע הגיע ללא שינויים.

<u>השימוש ב CRC</u> מאוד נפוץ מהסיבות שהוא יחסית פשוט להבנה , ומאוד יעיל בגילוי שגיאות. ההבדל העיקרי בין checksum רגיל לבין crc לדעתנו הוא ש checksum רגיל לא מסוגל לזהות את כל השגיאות ב2 ביטים או יותר במידע, ואילו crc מסוגל לזהות את השגיאות ב2 ביטים יותר טוב מה checksum.

<u>אופן פעולת האלגוריתם</u>: האלגוריתם פועל על קלטים בינאריים. ומחזיק עוד מספר בינארי הנקרא "פולינום". ניתן לבצע חילוק ארוך בין הקלט לבין הפולינום, וכן נקבל שארית. את אותה השארית נשלח לקליאנט יחד עם ההודעה. הקליאנט יבצע את פעולת החילוק הפעם על קלט מחובר עם השארית שהשרת שלח. כעת תוצאת השארית צריכה להיות 0 אם ורק אם התוצאה תקינה.

<u>בדוגמה הבאה:</u> בחלק העליון ניתן לראות את צד השולח: בחירה של הפולינום מצד שמאל כ x^3 + 1 ,1001. מצד ימין ניתן לראות את חישוב שארית על ידי ביצוע חילוק ארוך של הקלט בפולינום. ובחלק התחתון ניתן לראות את הצד המקבל, לוקח את הקלט ומחבר את השארית שהתקבלה, ואת התוצאה מחלק בפולינום. וכן התוצאה התקבלה כנדרש.

<u>דוגמא:</u>

suppose we want to transmit the message 11001001
using the CRC polynomial $x^3 + 1$

```
polynom
x^3+1
1001 ) 11001001000   message data
       1001
       01011
        1001
        001000
         1001
         0001100
           1001
           01010
            1001
            0011
```

message to be transimitted = original + CRC = 11001001011

Receiver transmitted message 11001001011

```
received message
1001 ) 11001001011
polynom
x^3+1  1001
       01011
        1001
        001000
         1001
         0001101
           1001
           01001
            1001
✓           0000
```

Reminder is 0000 message transmitted successfully

שאלה 3 HTTP1.0 VS HTTP1.1 VS HTTP2.0 VS QUIC:

HTTP Hypertext Transfer Protocol - פרוטוקול תקשורת ברמת האפליקציה בעצם " עמוד השדרה " של גלישה באינטרנט פרוטוקול של בקשה ותשובה. כאשר "סוכן האינטרנט" למשל דפדפן יוצר בקשה שאותה הוא שולח לשרת והשרת משיב את המידע כתשובה לבקשה. בעצם חיבור בין המחשב לעולם האינטרנט.

פרוטוקול HTTP 1.0 - עוד לפני כן הגרסא הראשונה של פרוטוקול ה HTTP נקרא לימים 0.9 היה מאוד מאוד פשוט כאשר בקשות הכילו רק שורה אחת אשר הכילה את השיטה היחידה GET ולאחר מכן הכתובת למידע המבוקשה הכתובת המלאה לא הייתה מוכלת בשדה וגם הפורטים לא היו הכרחיים ברגע שהתבצע חיבור לשרת.
בדומה גם השרת השיב תשובות קצרות ופשוטות מאוד שהכילו את הקובץ בלבד
(ללא HEADERS , ללא סטטוס קוד או קוד שגיאות היו יכולים להעביר רק קבצי HTML)
גרסא 0.9 הייתה מוגבלת מאוד ובמהירה דפדפנים ושרתים הפכו התפתחו ופיתחו את השיטה כאשר כאשר נוספו שיטות GET, HEAD,POST ,PUT,DELETE,LINK,UNLINK מידע נוסף נשלח בתוספת לבקשות (עבור בקשות GET נוסף מידע לגבי גרסאות) בתשובות מהשרת נוסף סטטוס קוד שסיפק לדפדפנים עוד מידע בנוגע להצלחה שגיאה ולהגיב לכך בהתאם (שימוש ב cache מקומי)
שימוש ב HEADERS של HTTP לבקשות ותגובות ויכולת להעברת מידע נוסף חוץ מרק קבצי HTML התפתח בזכות לקונספט של HEADERS

פרוטוקול HTTP 1.1 -בעצם תקנון הפרוטוקול יצא לא הרבה לאחר גרסה 1.0 ו 0.9 והתבסס על גרסאות קודמות לו, הבהיר כללים והביא שיפורים רבים : 1) אי צורך בפתיחת חיבור פעמים רבות עבור הצגת קבצים , אלא חיבור אחד ושימוש חוזר בו . 2) "צנרת" יכולת לשלוח בקשה נוספת עוד לפני קבלת תשובה על בקשות קודמות לכן. 3) שיפור בקרת קאש' . 3) משא ומתן כלומר נוספו שדות לגבי שפה "נתמכת" קידוד , וסוג בקשה. יותר מ15 שנה של התפתחויות

פרוטוקול HTTP 2.0 -כאשר דפי אינטרנט התפתחו נהיו מורכבים יותר גדולים יותר וכמויות המידע שהועברו גדלו באופן משמעותי פותח פרוטוקול HTTP 2
בשונה מגרסא 1.1 הוא יותר בינארי , אפשר לשלוח בקשות רבות במקביל על גבי אותו חיבור ללא האילוצים של פרוטוקולים בגרסאות 1 דילול ה HPACK" HEADERS" כלומר פחות כפילויות אפשר מכניקה חדשה בשימוש CACHE

עד כאן הפרוטוקולים השתמשו בחיבור TCP בין השרת ללקוח

פרוטוקול QUIC - תומך בסמנטיקה של HTTP 2.0 אך ההבדל העיקרי הוא שימוש ב UDP יותר גמיש ביחס ל TCP שדרוגים ועדכונים בפרוטוקול אינם קשורים למערכת ההפעלה כמו ב TCP כל חבילה מוצפנת בעצמה בשונה מTCP כאשר ההצפנה על כל הזרם של הביטים QUIC בא לשפר גם אירועים של החלפת רשת בין קווית לאלחוטית וסלולרית כאשר זה קורה ב TCP תהליך ארוך מתחיל והחיבור מתחיל מחדש על מנת לפתור את זה QUIC מכיל זיהוי עבור החיבור בנוסף ל ID .

שאלה 4 למה צריך מספרי פורט:

מספרי פורט: פורטים הם "תאים" וירטואליים במערכת ההפעלה, כל פורט משויך לתהליך מסוים והשימוש בהם עוזר למחשב בסידור המידע שמגיע אליו ברשת.
כאשר מחשב מקבל הודעה ברשת הוא צריך לדעת איזה תהליך מצפה לקבל את ההודעה , לשם כך צריך Port-Numbers כאשר תהליך ניגש לרשת האינטרנט הוא עושה bind למספר פורט מסוים אותו מספר מוכל בהודעה שנשלחת מאותו תהליך או בהודעה המתקבלת כאשר המחשב מקבל תשובה הודעה מהצד השני הוא מחפש את מספר הפורט ומעביר על פיו לתהליך שמוצמד לאותו פורט.
(טווח הפורטים נע בין 0 ל 65353)

שאלה 5 מה זה **SUBNET** למה צריך את זה :

מה זה subnet?
סאבנט תת רשת חלק מתוך הרשת הגדולה יותר כלומר אפשרות לחלוקת רשת גדולה לתתי רשתות קטנות יותר.
למה החילוק של הרשת לתתי רשתות (subnets) הופך את חלק מהתנועה ברשת ל "בלתי נראית "
(למשל 2 מחשבים המחוברים בתוך רשת של ארגון גדול רוצים לתקשר ביניהם במקום ששניהם יתקשרו דרך נתב אחד במקרה ובו יש תת רשת אחת בלבד אליה מחוברים כל המחשבים - עומס שיכול להיות גדול מאוד .
כל מחשב יהיה שייך לתת רשת שאחרת בתוך הרשת הגדולה של הארגון וכאשר שני המחשבים ירצו לתקשר הם לא יהיו חייבים לעבור דרך כל הרשת אלא רק דרך המסלול של הנתבים באותה רשת)
בעצם subnet עוזר לייעל ולנהל רשת גדולה יותר בעזרת תתי רשתות.

שאלה 6 -למה צריך כתובת **MAC**:

מה זה כתובת MAC ?
Media Access Control address כתובת MAC כתובת ייחודית הניתנת ומוטבעת על ידי היצרן (שמקבל טווח כתובות של מרשות IEEE) כתובת MAC נקראת גם "כתובת פיזית" והיא שונה מכתובת IP שכתובת ה IP יכולה להשתנות בעוד כתובת MAC נשארת קבועה ("כמו טביעת אצבע") .

למה צריך את כל זה?
סיבה ראשונה מכיוון שכתובות IP אינן בהכרח ייחודיות , לפני שמחשב מקבל כתובת IP הוא שולח בקשה (broadcast) תוך שימוש ב DHCP (פרוטוקול תקשורת שאחראי לחלק כתובת IP למחשבים ברשת) בשלב הזה של הבקשה ההודעה תהיה בסגנון של "היי , אין לי כתובת IP . הכתובת המזהה שלי היא *MAC*. האם השרת יכול לתת לי כתובת IP ולרשום אותי כדי שאוכל לתקשר ברשת"
כאשר הנתבים והמתגים מחזיקים טבלאות עם כתובת MAC וכתובת IP מקושרת לאותה כתובת.

שאלה 7 מה ההבדל בין **Router , Switch , NAT**:

ראוטר (נתב) - מכשיר המשמש לחיבור רשתות מכוון חבילות המועברות ברשת על פי כתובות יעד וכללים מוגדרים מראש

- ממוקם בשערי הרשת (gateways) היכן ש2 רשתות מתחברות אחת לשנייה (לדוגמא: הראוטר הביתי מחבר בין הרשת הביתית לרשת החיצונית)
- מנתב תעבורה בין מכשירים ובין רשתות
- יוצר רשת "חדר" בו אפשר לדבר אחד עם השני בלי לדרוס זה את זה ובנוסף יכול להעביר שיחות החוצה למחשבים ב"חדרים" אחרים (למעשה משתף כתובת IP פומבית בין המון מכשירים)
- שכבת ה network layer
- משתמש בNAT

סוויצ' (מתג) - מחבר כמה מכשירים ביחד (מזכיר מפצל) עם זאת הוא יכול לעשות קצת יותר "מסתם לפצל" (אפשרות ליצור VLAN רשת לוקאלית וירטואלית חלק מרשת ברשת מקומית שמוגדר מבחינה לוגית אוטונומי

- בדרך כלל בתוך רשתות פרטיות מאפשר
- מעביר תעבורה
- מחבר אנשים באותו "החדר" שיוצר הראוטר , מאפשר להגדיל את החדר
- עובד בשכבת ה data link layer
- אינו משתמש ב NAT

נאט NAT Network Address Translation - לכל מחשב יש כתובת IP ברשת כאשר אנו רוצים לגשת למקום מסויים אנחנו פונים לכתובת IP ציבורית/פומבית שלו ובכך אפשר למשל לגלוש באתר מסוים.
אבל מכיוון שיש אינספור מכשירים מחשבים וכו' המחוברים לרשת וכל אחד צריך כתובת IP , לשם כך פיתחו את הרעיון של כתובת IP פרטית ובכך להאט את הצמיחה של כמות כתובות ה IP , נניח למשל יש לנו רשת ביתית ובה 3 מחשבים לכל מחשב מקבל כתובת IP פרטית (DHCP) שלושת המחשבים באותה רשת פרטית יכולים לראות אחד את השני אבל אף אחד מחוץ לרשת הפרטית יכול לראות אותם מכיוון שאין להם כתובת IP ציבורית. לראוטר יש כתובת ציבורית ובכך כל אחד ברשת יכול לראות אותו כך שכל המכשירים המחוברים ברשת הביתית מדברים עם החיצון דרך הנתב.

כאן בדיוק נכנס ה NAT כאשר מכשיר ברשת הפנימית מבקש להוריד מידע מסוים מהאינטרנט הוא שולח בקשה , הראוטר מקבל את הבקשה ורואה אותה , הוא לוקח את אותה בקשה ושולח אותה בעצמו ומבקש בעצמו את המידע . כאשר הוא מקבל תגובה בחזרה הוא צריך להעביר אותה אל אותו מחשב שביקש וזה מה שה NAT עושה.

שאלה 8 שיטות להתגבר על מחסור ב **4IPv** ולפרט:

מחסור ה- 4IPv הבעיה: ברשת האינטרנט כל מחשב זקוק לכתובת IP כתובת ה IP בצורה של 4IPv היא כתובת המורכבת מ32 ביטים בינאריים כך שכמות הכתובות ב4IPv היא אינה סופית למעשה $2^{32} = 4{,}294{,}967{,}296$ קומבינציות אפשריות לכתובת IP אמנם זה מספר עצום אך הוא סופי. כאשר יש המון מחשבים מחוברים לאינטרנט כאשר לכל בן אדם יש לפחות מכשיר אחד ואפילו כמה בנוסף טווחי כתובות שניתנו לחברות וארגונים גדולים.

פתרונות:

- שימוש בכתובות בתצורה של 6IPv מוגדר על ידי 64 ביטים בכך מגדיל את כמות הכתובות 2^{64} מספר עצום (גם הוא סופי) הייצוג של כתובת מסוג 6IPv היא סדרה של 8 מספרים בני 4 ספרות בבסיס הקסדצימלי (2001:0db8:0001:0000:0000:0ab9:C0A8:0102)
- השימוש ב NAT מאריך את תוחלת החיים של 4IPv סוג של דוחה את הבעיה . בעצם השימוש בNAT נותן את האופציה להגדיל את התווח שימוש בכתובות 4IPv פרטיות
- שימוש ב CIDR Classless Inter-Domain Routing בעצם שיטה להצגת IP בצירוף למסכת רשת (Network Mask) לדוגמא 198.51.100.14/24 בעצם מייצג את הכתובת 198.51.100.14 ששייך לתחילית (prefix) של הרשת 198.51.100.0 כאשר ה Subnet Mask שלו הוא 255.255.255.0. כמות הכתובות באותה רשת או תת רשת יכולה להיות מחושבת כך $2^{address\ length - prefix\ length}$ (דוגמא נוספת 198.51.100.0/22 מייצג 1024 כתובות 4IPv בטווח של 198.51.100.0 ל 198.51.103.255)

מסכת רשת Subnet Mask - בעצם קובעת את כמות תתי הרשת (מתקשר לנו לשאלה של subnet) למשל יש לנו 300 מחשבים ברשת נצטרך 2 תתי רשת למשל 10.1.10.1-255
ו- 10.1.11.1-10.1.11.25 נצטרך עבור הדוגמא הזאת subnet mask של 255.255.254.0 כלומר (1111.1111.1110.000) כאשר המחשב יחפש כתובת IP הוא ישתמש ב subnet mask ובפעולת AND ויבדוק האם 11 הסיביות הראשונות דומות אם כן הוא שייך לתת רשת הזאת

שאלה 9:

א) הנתב c3 לומד על תת הרשת x בעזרת פרוטוקול OSPF הוא לומד על 4AS ומגלה את c4 , משם c4 מגלה את 4ש בעזרת BGP
ב) הנתב a3 לומד על תת הרשת x בעזרת פרוטוקול BGP הוא מגלה את b3 שבעזרת אותו פרוטוקול מגלה את c3 ומשם c3 על פי א'
ג) הנתב c1 לומד על תת הרשת x בעזרת פרוטוקול RIP הוא מגלה את 3AS בתוך 3AS פועל BGP ומשם על פי ב' וא'
ד) הנתב c2 מגלה בעזרת BGP את a2 משם 4AS מגלה בעזרת OSPF את 1AS מ 1AS הנתבים מגלים אחד את השני בעזרת BGP כאשר מ c1 על פי ג' ב' וא'