15 פרוטוקול Datagram משתמש (UDP)

אם אתם אוהבים לשמור על הדברים פשוטים ואתם חושבים חיובי, UDP מתאים לכם. זה כמעט אולטימטיבי בהעברת נתונים קלה על גבי האינטרנט.

אתם שולחים מנות UDP ומקווים שהן יגיעו. אולי הן יגיעו, או שאולי מישהו שם טרקטור חפר דרך כבל סיבים אופטיים, או שהיו קרני קוסמוס, או שהראוטר היה עמוס מדי או כועס וסתם זרק את זה. בלי טקס.

זה החיים על קצה המידע באינטרנט! כל ההבטחות הנעימות והאמינות של TCP–נעלמו!

15.1 מטרות של UDP

לספק דרך לשלוח נתונים ללא טעויות ממחשב אחד למחשב אחר.
זה בערך כל מה שיש לזה להציע.

הבאים אינם מטרות של UDP:

- לספק נתונים בסדר
- לספק נתונים ללא אובדן
- לספק נתונים ללא כפילויות

אם אלו דרישות, TCP הוא בחירה טובה יותר. UDP לא מציע שום הגנה מפני אובדן או סדר שגוי של נתונים. ההבטחה היחידה היא שאם הנתונים מגיעים, הם יהיו נכונים.

אבל מה שהוא כן נותן זה עזרים מאוד נמוכים וזמני תגובה מהירים. אין לו שום דבר כמו חידוש מנות, שליטה על זרימה, חבילות ACK או כל דבר ש-TCP עושה. כתוצאה מכך, הכותרת הרבה יותר קטנה.

15.2 מיקום בסטאק הרשת

זכרו את השכבות בסטאק הרשת:

שכבה	אחריות	פרוטוקולים לדוגמה
יישום	נתונים מסודרים ליישום	HTTP, FTP, TFTP, Telnet, SSH, SMTP, POP, IMAP
תחבורה	שלמות נתונים, פיצול והרכבה של מנות	TCP, UDP
אינטרנט	ניתוב	IP, IPv6, ICMP
קישור	פיזי, אותות על חוטים	Ethernet, PPP, Token ring

אתם יכולים לראות את UDP בשכבת התחבורה. IP מתחתיה אחראי על ניתוב, והשכבה העליונה מנצלת את כל מה שיש ל-UDP להציע, שלא הרבה.

15.3 פורטים של UDP

UDP משתמש בפורטים, בדומה ל-TCP. למעשה, אפשר שיהיה תוכנית TCP שמשתמשת באותו מספר פורט כמו תוכנית UDP אחרת.

IP משתמש בכתובת IP לזיהוי מארחים.

אבל ברגע שהמארח מזוהה, מספר הפורט הוא מה שהמערכת הפעלה משתמשת בו כדי לשלוח את הנתונים לתהליך הנכון.

באנלוגיה, כתובת ה-IP היא כמו כתובת רחוב, ומספר הפורט הוא כמו מספר דירה באותה כתובת רחוב.

15.4 סקירה כללית של UDP

UDP הוא פרוטוקול ללא חיבור. אתם יודעים איך TCP לוקח את הרשת המנות והופך אותה למראה של חיבור אמין בין שני מחשבים? UDP לא עושה את זה.

אתם שולחים Datagram של UDP לכתובת IP ופורט. IP מנווט אותו לשם והמחשב המקבל שולח אותו לתוכנית שקשורה לאותו פורט.

אין חיבור. זה כל אחד בנפרד לפי מנת נתונים.

כשמנה מגיעה, המקבל יכול לדעת מאיזו כתובת IP ופורט היא הגיעה. כך, המקבל יכול לשלוח תגובה.

15.5 שלמות הנתונים

יש הרבה דברים שיכולים להשתבש. הנתונים יכולים להגיע לא בסדר. הם יכולים להיות משובשים. הם יכולים להיות כפולים. או שאולי הם לא יגיעו בכלל.

ל-UDP כמעט ואין מנגנונים להתמודד עם כל התקלות האלה.

למעשה, כל מה שהוא עושה זה גילוי שגיאות.

15.5.1 גילוי שגיאות

לפני שהשולח שולח את המנה, מחושב סכום בקרה עבור אותה מנה.

הסכום הזה עובד בדיוק כמו ב-TCP, רק שהכותרת של UDP משומשת. ביחס לכותרת של TCP, כותרת UDP היא פשוטה מאוד:

```
 31     24 23     16 15      8 7       0
+--------+--------+--------+--------+
|   Source      |   Destination     |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
                                     |
            ... data octets          |
                 ... ----------------+
```

כאשר המקבל מקבל את המנה, הוא מחשב את סכום הבקרה של המנה הזו.

אם שני הסכומים תואמים, הנתונים נחשבים כנטולי שגיאות. אם הם שונים, הנתונים נזרקים.

וזהו. המקבל אפילו לא יודע שהיה נתון שנשלח אליו. הוא פשוט נעלם לחלל.

הסכום הוא מספר בן 16 ביטים שנוצר על ידי העברת כל כותרת UDP והנתונים יחד עם כתובת ה-IP המעורבות לפונקציה שמעבדת אותם.

ההליך הזה עובד בדיוק כמו סכום הבקרה ב-TCP. (ג'ון פוסטל כתב את ה-RFC הראשון עבור TCP ו-UDP, אז זה לא מפתיע שהם משתמשים באותו אלגוריתם.)

הפרטים על איך הסכום עובד נמצאים בפרויקט השבוע. פשוט החליפו את כותרת UDP בכותרת TCP.

15.6 מִטְעָנֵי עומס ללא פילוח

זה קצת מוקדם מדי, אבל שכבות נמוכות יכולות להחליט לפצל מנת UDP למנות קטנות יותר. אולי יש איזור באינטרנט שהמנה צריכה לעבור בו ויכולה להתמודד רק עם נתונים בגודל מסוים.

אנו קוראים לזה "פילוח", כאשר מנת UDP מפולגת בין מנות IP מרובות.

הגודל המרבי של מנת נתונים שניתן לשלוח על כל חוט נתון נקרא MTU (יחידת שידור מרבית). ה-MTU הקטן ביותר האפשרי באינטרנט (IPv4) הוא 576 בתים. הכותרת הגדולה ביותר של IP היא 60 בתים. והכותרת של UDP היא 8 בתים. אז נשארים 576-60-8 = 508 בתים של נתונים שניתן להבטיח שלא יפולגו.