

סיכום מצגות IoT 2020 – אביב 236333/2

תוכן עניינים

2-5.....	הרצאה 1
6-8.....	הרצאה 2
9-12.....	הרצאה 3
13-17.....	הרצאה 4
18-20.....	הרצאה 5
21-25.....	הרצאה 6
26-31.....	הרצאה 7
32-35.....	הרצאה 8
36-39.....	תרגול אRDDoIIm ומעגלים שימושיים
41.....	안дрואיד
42.....	Serverless Computing
43.....	Comm Tech & Protocols

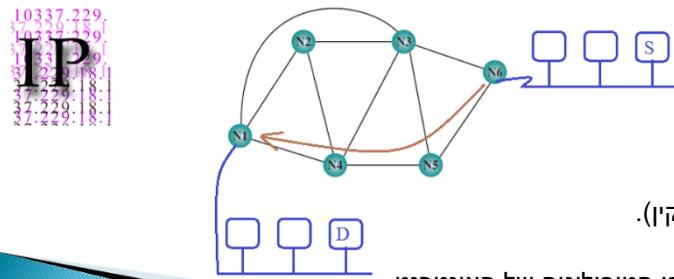
מה זה האינטרנט? אוסף של רשותות מחשבים. האינטרנט, הוא בעצם חיבור בין רשתות. בצורה הזאת כל מחשב בכל רשת יכול לדבר עם כל מחשב בכל רשת אחרת. החוקנים הראשונים:

1. IP - פרוטוקול הניתוב.
 2. TCP - פרוטוקול התעבורה.

פרוטוקול זה אוסף כללים המקבול על שני הצדדים על מנת לסייע את תפקידם. כלים מאוד מסודרים, בהיקש שלנו: תוכנה שיזענעם מה مصدرיהם, איך מקבלים, מה מקבלים, איך מזינים ויזענעם לפעול בהתאם לפרוטוקול מסוים.

פרוטוקול הניתוב IP

מחשב ייעד דרך האינטראקטיבי. כאשר המקור והיעד מחוברים דרך נתבים ▶ ה-IP לומד את הטופולוגיה של הרשת וועשה כמייטר יכולתו להעביר את נתונים לרשותות שונות. יתכן שנ לצורך דרכם כמה נתונים כדי לחבר האינפורמציה מתחנות מקור לתחנות יעד.



כשמו כן הוא, מתבגר את המידע עם מחשב מקור (source) אל מחשב יעד דרך האינטרנט. כאשר המקור והיעד מחוברים דרך נתוני לרשותות שונות. יתכן שנציגך לעברך דרך כמה נתונים כדי את הרשותות של המקור והיעד.

הPA "בוחר" דרך נציגים להעביר את המידע. הבחירה נעשית בדף "לפי המסלול הcé קצר (מספר קשתות מינימלי). בפועל זה מה שקיים כיום, אבל אפשר לחלוטין להגדיר אופן אחר, כמו המסלול הקל ביותר, המסלול עם רוחב הפס הגדול ביותר, המסלול הקשה ביותר וכו' בלאל.

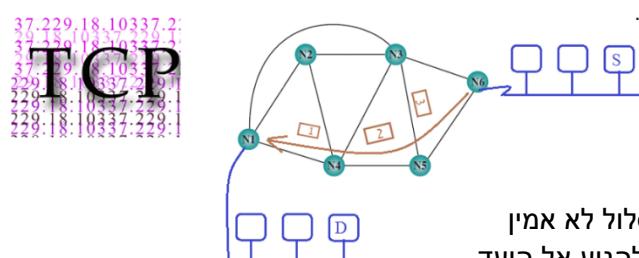
בכך הוא כמו הווי' של האיגניטרנט. הוא גורע את המסלול הטוב ביותר לפיו. הטופולוגיה של הא'ברטנט.

TCP

דווגה להבהיר את המידע בין שתי אפליקציות ללא שגיאות, וכך גם מאשר ריבוב מידע בין כמה אפליקציות על אותו מחשב. זה פרוטוקול שנכתב בשנת ה-70 של המאה ה-20.

פרוטוקול התעבורה TCP

ה- TCP דואג להעברת האינפורמציה בין שתי אפליקציות ללא שגיאות, שפכול או יציאה מסדר ומאפשר ריבוב אינפורמציה בין כמה אפליקציות על אותו מחשב.



אםنم הוא עובד מעולה, אבל ביום הוא מאבד מהיוקה של
כי הוא לא מתאים למשל לתקורתם לוויים, ולכן
כן געשים שיפורים ונכתבם פרוטוקולים יותר מודרניים.
כיוון שאי אפשר ולא רצוי להזכיר קבצים גדולים דרך

הainternet בבת אחת, הTCP/IP בעצם חותך את המידע לחבילות קטנות של 1500 בתים בדרך כלל ומתיחיל לשדר אותן (בתוספת מידע), על מנת לשמור על סדר וכי לדעת מה משודר). החבילות משודרות מהמקור אל היעד לפי IPו. בדוגמא, המידע מופרד ל-3 חבילות. חבילה 1 ו-2 יוצאות דרך המסלול המוכתב ע"י IPו. בגלל סיבה כלשהי, IPו מחליט שהמסלול ולכן מחליט לשדר את חבילה 3 במסלול אחר. אבל עדין היא אמורה למצוא בעיות זה פותח?

1. יציאה מסדר: חבילה 3 תגעה לפני חבילה 2. פתרון: כמו אמרנו, הTCP משדר את החבילות עם עד מידע (metadata). המידע הזה יכול להכיל אינדוקס של החבילות. כמו אמרנו, המקור והיעד מכירים את אותו פרוטוקול ולכן אם המידע נשלח לפי פרוטוקול שמתכוון את סדר החבילות, היעד מכיר את הפרוטוקול וידע להרכיב

2. חבילה נאבדה או החבילה לא תקינה. הTCP יכול לzechות א-תקינות (למשל לפי bit parity או לפי זמן שהיה מוגדר). במקרה ההפוך, היעד יכול להציג שיש בעיה עם החבילות ולבקש מהמקור לשילוח מחדש את החבילה.

3. שכפול: יתכן שהabilia 2 התעכבה מספיק בשבייל היעד לבקש שידור חוזר, ונוצר מצב שהabilia 2 הגיעו פעמיים אל היעד. לפי פרוטוקול TCP/IP, הוא יזהה שיש לנו שתי חבילות עם אותו אינדקס ויזרק אחת מהן.
4. ריבוב מידע: מידע מקבילי הגיעו בו זמני מכמה ערוצים (למשל, סרטון ביוטוב ומיל). הTCP יודע לאיזה אפליקציה מיועדת כל חבילה כדי שהחבילות שיועדו למיל לא יערבו לנגן יוטוב והפוך.

Wide Area Network – WAN
האינטרנט שיר למשפחה של רשתות לאיזורי נרחבים. ה-WAN מקשר רשתות מקומיות. כל רשת מקומיות מכילה יחידות שונות: נתבים, מחשבים וכו'. האינטרנט דיבר את כל משפחת WAN.

Local Area Network – LAN
 מבוססות יכולות פיזיות. כבלי נחושת עבור חיבור קווי, אויר עבר WIFI, אופטיqua עבור סיב אופטי וכו'.

Body Area Networks & Personal Area Network – BAN & PAN
רשתות של סנסורים עליינים. ההבדל:

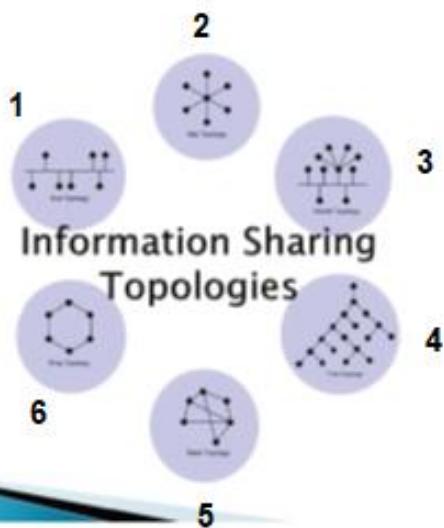
PAN- רשתות שלובשים עליינו, כמו שעון חכם או بد עם סנסורים. (נמצאים פיזית על הגוף)

BAN- מחוברים אליו. סנסור רמת סוכר או קוצץ לב. (נכנים פיזית לגוף)

רשתות אלו יכולות לדבר עם יחידות שליטה (כמו סמארטפון או טאבלט) שנמצאות ב-LAN. בעצם, הקוצץ לב מדבר עם הסמארטפון, והסמארטפון מדבר עם האינטרנט. זה יכולות חישוב וביצועים וגם מקל על רשת האינטרנט. באופן זהה, לא באמת כל המכשירים מדברים עם רשת האינטרנט. יחידת השליטה משתמשת מהם.

שיתוף מידע
יחידות קצה יכולות לדבר בין עצמם (P2P) על רשת מקומית או דרך בקר שמתווך ביניהן.
יכולות לשתף שרטטים או יחידות אחרות על רשת WAN או MAN.
ה/devices שלנו, יכולים להיות יחידות קצה ברשת סנסורים עצמאית, שמדוחת למחשב קצה, סנסורים המתקשרים ביניהם ומאותם רק במידה יש צורך, או ממש יחידות P2P.

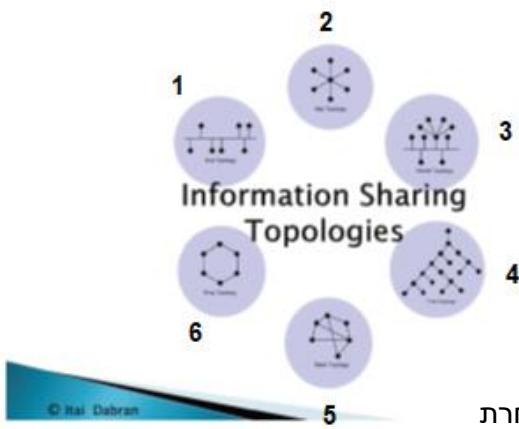
טופולוגיות חיבור



1-רשת של דברים שמחוברים על BUS אחד. רק מחשב אחד יכול לשלдр ברגע נתון, וכל האחים יכולים להאזין ואם רלוונטי ללקחת את המידע המשודר. כל היחידות חולקות אותו קוו. ה-Ethernet עובד ככה.

2- כוכב, כל היחידות מחוברות ליחידה מרכזית שמעבירה את המידע ביניהן. מעבר בין כל שתי יחידות הוא 2 קפיצות. בהינתן שהיחידה המרכזית היא יחידה חזקה. החסרון שהנקודה המרכזית היא נקודת כשל. רשת WIFI עובדת באופן דומה.

3- היברידית, שילוב של 2+1. רשת מחשבים על BUS משותף. אחד מהרכיבים על ה-BUS הינו כוכב, יחידה מרכזית. לדוגמה, שילוב של Ethernet ו-WIFI.



6- טבעת. רשיימה דו כיוונית, מסלול מעגלי. כל יחידה יכול לפנות ליחידה אחרת דרך המעגל. היתרון, שאם יש כשל באחד הרכיבים, עדין קיימں מסלול בין היחידות (כיוון שני של המעגל).

5- רשת Mesh. נפוצה בעולם IoT. כל מחשב יכול להיות מחבר למחשב אחר. לכן כל רכיב יכול לדבר באופן ישיר עם רכיב אחר. למשל, אצל Drones שמדוברים אחד עם השני.

4- עז. בד"כ כדי לחבר כמה רשתות מקומיות אחת לשניה ולראש המקומיות גדולה יותר. היתרון, שהגעה בין רשתות הוא aglog. הכשל הוא כאשר צומת כלשהו נכשל, יתכן שהוא כשל בעז.

דרכי תקשורת של הדברים

בין לבין עצם – דלת הכניסה מודיעה למזגן שאדם נכנס הביתה למשל. לטלפון חכם אצל המשמש – נספק דרך הטלפון או התאבלט מתכוון לעוגה, והמרקער יענה איזה מצרכים יש ואיזה ציריך לנקות (ואולי אף יבנה כבר רשימת קניות וזמן בעצמו). שליחת מידע למרחוק – קריית מונה החשמל, נשלח את המידע לחברת חשמל. שליחת מידע לקבוצה דרך הענן – למשל אוסף מקרים רפואיים. קיימים פה עניין של פרטיות ובטחת מידע. למשל, במהלך ההתקנות יכולים לוחץ על agree זהה מאפשר למסר לשלווח מידע לאם ענק של חברות מסחריות, כמו תנובה לדוגמא. תנובה יכולה ללמוד את הנתונים ולהשபיע על מחيري המוצרים שלא לפיה איזוריהם לפחות המידע הזה. זה יכול להיות טוב וזה יכול להיות רע.

היכן הדברים נמצאים?

היכן יש דברים – הדברים שבדרך

- חישבי טופוGRAF, מצלב בכיבשים, עומס בכיבשים, מכוניות ללא נהג, חישבי תנועה על מעבר חיצה, חישבי אורות וכו'.
- דברים באנווטות המקשרים לדברם בנמל ומרחוק לחברות שנوع שיגיעו.
- דברים לדיווח מיקום תחבורה ציבורית.
- דברים לדיווח על פקקים למרחוקים.
- רכבים אוטונומיים ברמות שונות:
 - רמה 0 – שליטה נגה מלאה
 - רמה 1 – פעילות בסיסית (לשלול כוונת האצה) נשלטת ע"י הרכב
 - רמה 2 – הרכב יכול לספק שליטה של מושל cruise control ומכרז נתיב אבל הנהג חייב להיות מוקן לקחת שליטה של הרכב בכל רגע
 - רמה 3 – אפשרה התקשרות לא רצפה על הכביש
 - רמה 4 – אפשר להסיר תשומות לב מהכביש
 - רמה 5 – רכב ללא נהג

היכן יש דברים

- רכבים חכמים
- מסחר
- שינוי חכם
- בת' מלון
- בריאות
- ועוד
- מדע
- ועוד.
- אנרגיה
- בניינים פרטיים
- בניינים תעשייתיים

כמה דוגמאות:

1. הרכבת התתית בסינגפור. על אף שהיא מאוד מסועפת, היא נשלטת לחלוטן באופן אוטונומי. אין סיבה שלא, שכן המסלולים קבועים, התחנות ידועות מראש ובהתאם לשעות הלחץ אפשר לחשב אלגוריתם אופטימלי לסנכרון רכבות.
2. כפר העובדים של סיאו. הכבישים בין הConfigurer למפעל הם אוטונומיים. חוסכים הרבה מקומות חניה, כי אין אנשים ברכבים אז חוסכים חניה כי המכוניות חונחות צמוד.
3. רכב שמחנה עצמאית בסין. האוטו רואה לפי סנסורים של מרחק ומצלמות של ראייה ממוחשבת. בנוסף של דבר נכנסים הרבה ביטים של מידע. כל הדברים האלה מבקרים ע"י מחשב. קלט של ביטים ויש אליו wrapper שידוע להוציא לפולט. אם כך, למה לשוב את ההגה של האוטו? למה לבזבז כוח חישוב על סיבוב הגה (שלא רלוונטי, שכן ההגוי מתבצע אוטומטית). זה מבחינה פסיכולוגית, להראות לנו שהכל עובד והכל תקין.

הדברים בעולם:

1. בעיר - חניות, רמזוריים, השקייה וכו'. יש כמה מכשוריהם: אורות בגנים ציבוריים שנכבים כאשר אין אנשים. אפשר לאנשים לא לגיטימיים להסתתר בשיחים ולתקוף אנשים לא חושים. צריך למצאו את האיזון.
2. בעבודה – רובוטים במחסני ענק (כמו באזזון) שיודעים לנוט את עצם בתוך מחסן ענק וմדברים עם מערכת בקרת מלאי (אם צריך חoser, יודעת להזמין עצמה עוד מוצרים).
3. בפאב – למזוג משקה לפי מרשם נפוץ או אישי מהסмарטפון ועוד.

IoT Near Future (I)



© Itai Dabran

הרצאה 2

נתה השוק של IoT צפוי לגדול עם השנים, במיוחד לאחר הקורונה. מצד אחד שוק הצלכנים ירד, מצד שני השוק למוצרי IoT נטול גיגפים כנראה עלה.

از איך בעצם הנושא התעורר? במשך הרבה שנים דיבור על בתים חכמים ומוכנות שנוציאות לבד.

בשנים האחרונות, שיפורים של מהירות אינטרנט (5G), פיתוח clouds, שיפורים אלגוריתמיים, פיתוח טלפונים חכמים ועוד עודדו את פיתוח IoT בשנים האחרונות ונתן לזה רוח חדשה. יש דימיון לחוק מוש: אינטל ו-AMD מייצרת מעבדים יותר חזקים, ובתגובה מיקרוסופט ואפל מייצרת מערכות הפעלה יותר חזקות יותר משאבם ומסוגלוות יותר יכולות.

שיפור האינטרנט מהירות האינטרנט מושפעת באופן פיזי (כבלים אופטיים, פיתוח שיטות wireless ב מהירות גבוהה, שידור בצלבים שונים ואפשר לשדר במקביל כמה ביטים ב מהירות האור על אותו כבל) וגם באופן לוגי (SHIPOR פרוטוקול TCP, מערכות caching שונות, דחיסת נתונים חבילות, אנטנות שמקבלות מידע מסווג מקומות ומשדרות מידע למספר מקומות).

פרוטוקול TCP לא מתמודד טוב עם תקשורת לוינים בגלל הפרעות מגז אויר וזמן שהייה ארוך יותר של חבילות. פותחו TCP ייעודיים ללוינים שיאפשרו מעבר מידע למרחק יותר גדול.

מערכות caching שונות, למשל שני שכנים שגוליםים לאתר CNN. השיכון השני ירצה לגלוש, והוא יוכל להשתמש במידע שבמطمון כדי לגלוש יותר מהר ולהחסוך רווח פס. יש מערכות גדולות של מטמון בשירותים שמוחברים אחד לשני אצל ספק האינטרנט. כמובן שישanza המון בעיות (אם זה אמר מחקרי זה מעולה, אבל אם זה מבזק חדשות, נראה שעד שיתמשו שוב במידע מטמון זה כבר לא יהיה רלוונטי). צריך לדעת לאיזה מידע לשים במטמון).

דחיסת נתונים חבילות- יש פרוטוקולים שלמים שמתעסקים בדוחסה. בלתי אפשרי לשדר תמונות גדולות או סרטים כי כל פיקסל מרכיב 32 ביט, ובתמונה 1080 יש מהו 2 מיליון פיקסלים, זה 66 מיליון ביטים עבור תמונה אחת. שלא לדבר על סרט, שלא לדבר על 4k. דוחסים נתונים ומשדרים רק את הנתונים ההכרחיים (למשל אם פיקסל נשרך קבוע במשך הרבה שנים, לא צריך לשדר אותו מחדש).

פיתוח הסמארטפון

ברגע שהטלפון התחבר לאינטרנט, הוא הפרק למחשב שנגish לכמות עצומה של נתונים ובעל יכולות גבוהות (לאו דווקא אצל היחידה, הוא יכול לשלווח מידע לען שיבצע חישובים). נוצר שטף של מכשירים שמתקשרים עם האינטרנט (SENSORS בטלפון) וקיבלונו עוד מידע שניtan להעביר בין הטלפון למשתמש ובין הטלפון לאינטרנט ולדברים אחרים.

SENSORS בסמארטפון (II)

- חיישן אוור - בודק את תאות הസבה
- חיישן GPS - בודק לחץ אטמוספרי ובמוצעות גובה ו współ-zeit (ה-GPS).
- תרמומטר - בודק טמפרטורת המכשור (ב- S4 S5 ועוד משתמש).
- חיישן לחות - בודק הסביבה הפיזית למשתמש
- חיישן זופק - Pedometer - למדידת עוצם וMOVE.
- חיישן כביש אבטחה (HTC).
- בדיקת קירינה - בשארם בגראם הפינה.
- מגנטומטר ומלכודת כוחות.

SENSORS בסמארטפון (II)



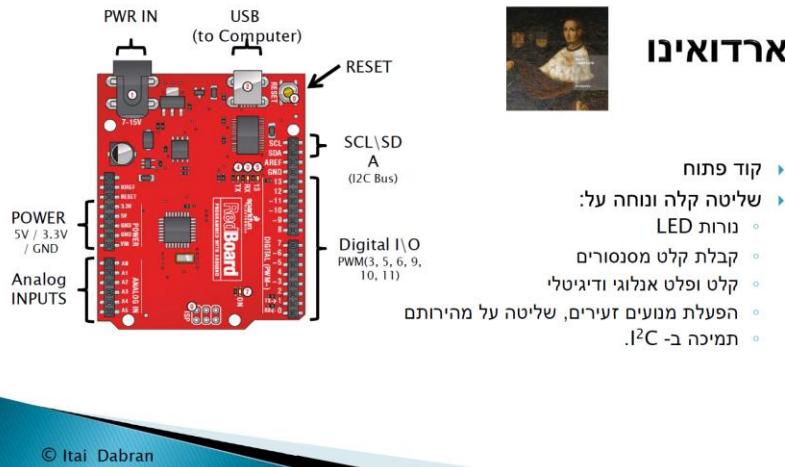
SENSORS לדוגמא:

- Accelerometer - חיישת האזנה: חיישת ליפוליה וחושפה, מרים את תנועת הטלפון לגדלים.
- Gyroscope - פאגת אוינטיציה בדיק ר. ב. שימוש ע. Google's Sky שאנון את קוקה המהיבן.
- Magnetometer - אברון שדות מגנטיים, לטעון או לאפליקציות לנצח את תנועה להשל.
- Proximity - מושלב עם אופטוא-יד LED וחישון ה-IR. מתקדם לוחות קירינה און להישר. סוכום מוקין רום משוחרר עיגום קירוב ובקלע ע. החישון

© Itai Dabran

גורמים נוספים:

- שיפור מערכות המחשב
- פיתוח סנסורים צעירים ופרוטוקולים אלחוטיים
- צרייכת חשמל מזערית של מכשירים חדשים
- פיתוח הארדואינו



לוח קל, קוד פתוח בחומרה ותוכנה, שליטה מודרנית על קבלת ושליחת מידע. נקרא שם של פאב שנקרו על שם של מלך איטליה.

BIG DATA

מילאardi אנשים מחוברים דרך הרבה דברים ונוצרת כמות אדירה של מידע בכל יום. את המידע ממתקות IoT ניתן לנתח ולעבד ולשפר את איכות החיים שלנו. למשל, עיבוד מידע לפני פקקים בדרכיהם (waze), יומניהם, לוחות זמנים, בריאות, תחבורה, אנרגיה. זאת כמובן בהינתן שהכל מתנהלה כשרה ולא באזדון, תמיד יש מקום לחשד. איסוף מידע שעצמו לא מביג ערך. כמות המידע שיש היום לא ניתנת לניתוח ע"י אנשים. לכן, תוצריו Big data נוצרות ע"י AI ו-ML. ככל שנאוסף יותר מידע איצוטי אפשר להגיע לתובנות עמוקות יותר.

איך מנתחים מידע? זיהוי תבניות
מיין תבניות של ביטים שחוזרים על עצם או נמצאים ביחד הרבה פעמים. ברגע שמנזה את התבניות נוכל לקבל החלטות. למשל, שני סוג של קבלת החלטות
- התערבות מוקדמת, לזהות תבנית שמנבאת משזה שיקרה בעבר ולכן ניתן להעירו אליו.
- אוטומציה חכמה, שני פס יוצר בנקודות זמן, כי לפי תבניות מסוימות השינוי יפיק יותר רוחים.
ברגע שיש מסקנות, אפשר לשלב ייחדות IoT בכך לשרף את המרכיבים.
תבנית לדוגמא, הבנת המילה "ירוק" בין "טירון ירוק" מול "תפוח ירוק". התבנית הראשונה מתארת תוכנה של בן אדם, התבנית השנייה מתארת תוכנה של פרט.
כל שימושי הוא deep learning. למשל בזיהוי פנים.
האתגר הגדול הוא כמון למצוא תבניות ולהפיך מסקנותabytes של מידע.

CLOUD COMPUTING

נותן אפשרות לאחסן מידע במערכות גדולות ולקבל שירותים של מערכות ירטואליות ושירותי אפליקציה בכל זמן ובכל מקום. זה מאפשר לדברים לבקש מהענן לבצע פעולות חישוב כבדות שהוא לא מסוגל לבצע בעצמו. מנגנון הענן זה פועליה יותר קלה. וכך להציג את התשובה למשיר.

יש מקרים בנוסא "בהינתן ענן מידע ואפליקציה", האם עדיף להביא את המידע מהען לאפליקציה ולחשב במכ Shir שלנו, או להביא את האפליקציה מהמכ Shir אל הען ולבצע את החישובים בען ורק להחזיר תשובה".

פריצת הדרך בתחום הייתה שירות AWS של AMAZON בספקה חכילה לאחסון מידע וקובלת שירותים מחשב ב2002. בהמשך התפתחה בתחום Elastic Compute cloud EC2 ב-2006 שאפשר לחברות קטנות לשכור מחשבים עבור העסק שלהם (EC2). זה חשוב הען הראשון שאפשר Software as a SaaS (aaS).

כמובן, שהדבר גורר התוצאות באבטחת מידע ופרטיות. הלוקה יכול לגשת למידע בכל זמן ומכל מכשיר, וכך הוא מאוד חשוב. צריך לוודא שמידע לא נמחק, נוסף או משונה לא במכoon. או חילילה, מישוה זדוני שהתחבר מרוחק ומכוון מוחקים קבצים שלא לצורך. הספקים עצם יכולים לשתף את המידע עם צד שלישי, בהתאם להסכם עם הלוקה. בענין שימושותם לכמה לוקחות יש צורך שאין זליגה ודילפת מידע מלוקה אחד ללוקה אחר. כאמור, בגל הטבע שליהם, הם מחזיקים המונן מידע ולכך מטרה למתיקות פריצה וכופרה.

IoT & Security

הנדסה חברתית – גניבת זהות ומידע שMRI על הפרטיות והביטחון בתקילה הייתה נמוך בסדר העדיפויות, בכך להוציא מוצר לשוק כמו שייתר מהר. ע"י צבירת מידע אפשר לעקב אחריו אדם: מידע כללי שנמצא באינטרנט, מדיה חברתית, שיעונים חכמים/צמדי כשר, מקרר חכם, שעון חשמל וכו'. ברגע שיש לנו את הזנות של הנאדם אפשר לחולב יותר מידע עליו. ניתן לסתוח את האדם, לעקב אחריו הרגלי הקנייה שלו ולהפעיל אליו מניפולציות, לגונב לו אפלו את הזהות. אם בנוסף גילו מידע עסק', הרוח של התקוף רק גדרה. ריגול תעשייתי. בנוסף לתקיפות תוכנה, אפשר פשטן לראות את הבן אדם מכניס את הסיסמא שלו למיכשיר. ברגע שגונבים את הטלפון או השעון החכם אפשר להוציא את כל המידע באופן קל. סمارפונטים באופן פרטני יותר מאובטים, עם טביעת אצבע, ID Face, והיכולת לחסום מכשיר מרוחק. כל שאר הדברים עדין נמצאים מאוד מאחור בנושא.

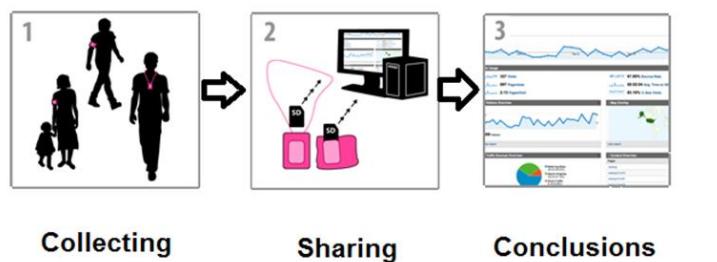
דוגמא: סטודנטים MIT תיכננו מודל של צב שמערכות AI זיהו כרובה. זה בקלהות היה יכול להיות הפוך
ורובה היה מזזהו ממשהו תמים.

הרצאה 3 – מבוא לBIG DATA ו-IOT

יש המון מידע שמסתובב בעולם שmagיע מהרבה מקורות שונות. למשל, רשותות חברות, תמנוניות, סרטונים GPS, RFID, הרב קוו שלנו. כל המידע הזה נאגר ונוסף וונצירות מערכות חדשות שתומכות בקבלת החלטות.

בג DATA יוצר מערכות חדשות לחלוּטן לתמיכת בקבלה החלטות ומגדיר כמיות נתונים עצומות ממקורות שונים. בגין כמהו המידע העצומה והסיבוכיות שבה (מגעה מהרבה מקורות, סנסורים של מג אוור, תగובות בראש חברותית, תמונות, וידאו, קבלות, שמירות נתוני GPS ועוד) לא ניתן לנתח אותו ולחשיך מסקנות בדרכים המקבילות. התפענה התרחשה בעקבות העלייה בקבלה נתונים, התפתחות רשותות חברותיות, גול, שירות ענן של AMAZON ומיקרוסופט וכו'. הרבה ארגונים, חברות וממשלות אוספים ומנתחים כמיות עצומות של מידע, לרוב לצורך רוח.

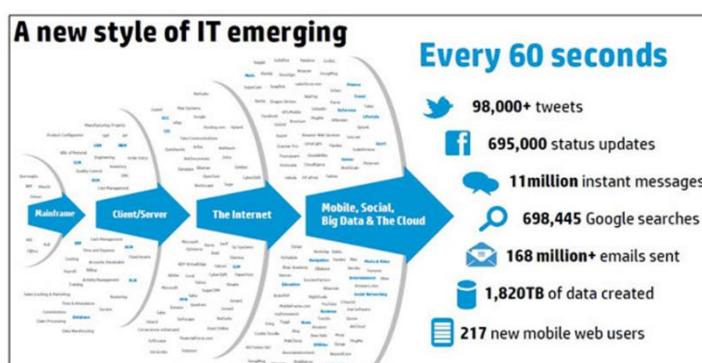
90% מהDATA בעולם נוצר בשנתיים האחרונות. כל ים אנחנו מייצרים 2.5 קוונטליון בתים של DATA



איסוף מידע איסוף של מידע יבש. מצלמות, סנסורים, GPS, תשלומים, מאגרי מידע על האנשים. כל המידע זה יישם לחלוטן. הרבה הרכבה נתונים ועובדות.

המידע זורם לבניין (חברות ביטוח, חברות מכירות, ממשלה וכו'). כמובן שיש סוגיות אתיות בנושא, שיתוף המידע בעיקר בנושא צנעת הפרט. יש הבחנה בין נתונים אישיים (תאריך יום הולדת, שם, גיל וכו'), שלרוב לא נרצה לתת אותם. נתונים התנהגותיים, זה מה שאנו עושים. אנחנו לומדים בטכניון, אנחנו נוסעים לאיביזה. גם זה מנותח ולרוב את המידע הזה נספק יותר בנדיבות.

הסקת מסקנות
ביחסות המידיע, האלבה בין מקורות שונים ומיפויים מסוימים.



از מה ההגדרה של ביג דטה? לפי WWW:

1. High volume – כמות המידע שקיים

2. High velocity – המהירות/קצב שהמידע נוצר

3. High variety – מגוון סוגי הדatos הנוצרים

IBM הוסיף עוד 7: Veracity – כמה המידע נכון או לא נכון. למשל, פיק ניוז, סקר שرك סימנו בו תשובה בלי לקרוא. בערך שלישי מבערל העסקים לא סומכים באופן מלא על המידע בשילוב קבלת החלטות.

האתגרים

פיתוח היכולות. תכנון אלגוריתמים, מודלים ותבניות שאכן יעצימו את התובנות. זאת כМОן עם כמות>Data אדירה. האתגר הגדול, זה קבלת החלטות בזמן אמיתי (אנליזה לökחת זמן).

צריך לדעת להציג את המידע (בין פורמטים שונים).

כאשר המידע רב מדי, צריך לדעת כיצד לנחות אותו (שגיאות, ערכים חסרים, פורמטים שונים של תאריך ועוד).

הרבה שיטות של כריית נתוניים קשות למימוש מבזורי, ונעשים מחקרים בנושא.

ניתוח של ביג דטה

כשיש לנו הרבה מידע בלבד, צריך לעשות לו ניתוח (Analytics). יש 3 סוגי של ניתוח:

1. Descriptive – מתאר את מה שהוא. תוצר הנתוח מגיע לגורם האנושי שמעיר מה יקרה בעתיד וקובע החלטות באופן אנושי, בהתאם לתיאור שספק ע"י הנתוח.

2. Predictive – ניסיין להעיר מה הולך לקרות, למשל ע"י ML. מבין וצופה מה יהיה בעתיד לאור הנתונים שנוחתו מהביבג>Data.

המטרה העיקרית היא שיווק, הבנת צרכי לקוחות והעדפותיהם.

3. Prescriptive – מגיע למסקנה וגם מנסה להציג מה לעשות. למשל, הצעת מחיר לינה לפ' איזור, עוניה. אם יש כנס גדול באיזור בפברואר, אז יעלה את המחיר של הלינה באיזור.

בגלל WWW, יש בעית ממשית באחסון וניהול הנתונים באמצעות מערכות מידע מסורתיות.

לכן, הטיפול נעשה בשיטות שונות:

1. Aggregation – למשל אם יש לנו מידע מהרבה סנוסרים, במקומות להציג את המידע מכל סנסור אפשר להציג רק את סיכום המידע. מינימום, מקסימום, ממוצע, ספירת כמה סנסורים.

2. ML Clustering – איחוד עצמים בעלי תכונות משותפות לפי קритריונים מסוימים. מחזיקים מידע אחד עבור כל העצים בקלאסטר זהה.

3. ML Classification – מראש מינים עצמים לקטגוריות שהוגדרו מראש, ואז מחזיקים את המידע לכל קטgorיה. המידע בין עצמים שונים יכול להיות שונה בין עצמים באותה קטgorיה.

Big Data - Analytics

לדוגמא:

בגלל של-4 החישנים זהים ובאותו איזור,

אפשר להבין שככל האיזור היה מתנהג

דומה, ואין צורך באיסוף מידע של כל 4

החישנים, מספיק ממוצע שלהם (כאיו חישן

(1). ברגע שאצל אחד החישנים, הכמות

חרוגת בצורה לא הגיונית מהממוצע אפשר

להתריע בהתאם ע"י מערכת גלובלית או לוקלית.

Time	temp	humid	light	CO	Time	temp	humid	light	CO
4/9/2017 11:23	40	30	100K	0.2	4/9/2017 11:23	40	30	100K	0.2
4/9/2017 10:23	29	40	100K	0.2	4/9/2017 10:23	29	40	100K	0.2
4/9/2017 9:23	28	40	90K	0.2	4/9/2017 9:23	28	40	90K	0.2
4/9/2017 8:23	22	50	90K	0.2	4/9/2017 8:23	22	50	90K	0.2
			10K	0.2		10K	0.2		
					4/9/2017 9:23	28	40	90K	0.2
					4/9/2017 8:23	22	50	90K	0.2

Descriptive Analytics

כורה מידע, ומבצע aggregation Data. מאפשר גילוי תכניות התהagaות כדי להציג נתונים. אפשר לדעת מה קרה. למשל, סיכון אשראי של לקוחות (לפי התנהגות בעבר, מצב חשבון קודם, רכוש נכסים וכו'). בסופה של דבר, התוצר מהניתוח מגיע לבנקאי, והבנקאי עצמו מחליט האם לחת אשראי לקוחות. הניתוח פה הוא רק כדי עזר לפישוט וסיקור הנתונים עבור הבנקאי.

Predictive Analytics

מנסה לנבأ מה הולך להיות, לפי מודלים ופונקציות סטטיסטיות, על סמך נתונים שקרו בעבר. אפשר לדעת "מה צריך/כדי לעשות". למשל, חברות שונות שבודקות נתונים מכירות, פרסום, תוצאות בפרסוק ועוד, ולפי זה החברה יכולה להסיק מסקנות לגבי העתיד.

Prescriptive Analytics

משתמש בסימולציות ואופטימיזציות על נתונים מה עבר על מנת לתת תשובה מה יהיה בעתיד. למשל, חברת שמייעלת את המלאי שלה. היא יכולה לדעת את צפי הצריכה בעתיד ומראש להזמין מלאי. עדין לא בשימוש נרחב, רק 3% מהחברות לפי גרטנר.

דוגמאות הסטארטअפים:

סטראט אקס ובקפה החדש

היום:

- למחמת התחלת המכירות שלו נדגמו תוצאות בבלוגים, טוויטר ופורומים של קפה.
- בצהרים אופיין שטעמו טוב אבל מחירו נראה יקר.
- תגובה תוך שעה: המחר יריד ובסוף היום לא ציוו יותר תוצאות חדשות.

בעבר:

- המתנה לתוצאות מה שיטה שיעברו ע"י אנשי המכירות
- אפיון הבעיה
- תגובה רק אחרי מספר שבועות.

דחיסת נתונים

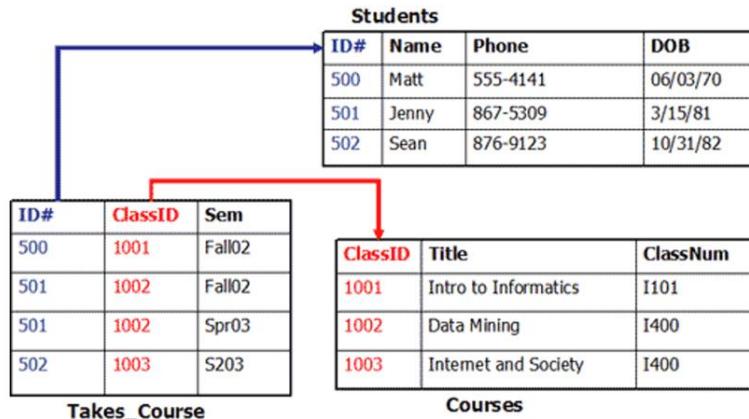
- ▶ דחיסת נתונים תקטיים כМОון את גודל המידע.
- ▶ הדחיסה מצומצמת בצורה טוביה מאוד את צריכת האחסון.
- ▶ הדחיסה חוסכת זמן רב עבור העברת המידע ועבור הגיבוי שלו.
- ▶ הדחיסה חוסכת משאבי תקשורת יקרים ומשאבי אחסון יקרים עבור גיבוי המידע
- ▶ המחיר הוא זמן העבודה כМОון של האינפורמציה.
- ▶ אחד הפתרונות לצמצום עבודה המידע הוא שימוש בדגימות בלבד אשר מספקות מבחינה סטטיסטית.

ניהול מערכות מידע בעולם הדיגיטלי

בגלל WWW, מסדי נתונים רגילים היו יקרים מדי לתפעול ולא הגיעו לביצועים טובים. מסדי נתונים רגילים הפכו להיות צואר בקובק. בגול פיתחו את שיטת Map Reduce.

את המידע עם מחלקים ל chunks (reduce), וכל חלק צזה מיפוי למחשב אחר (map), כאשר המחשבים עבדו במקביל. לבסוף, כל החלקים התאגדו לטור מחשב יעד שסיפק את המידע הנדרש.

מערכת נתונים רגילים (Relational Data Base)



3 הטבלאות מוגדרות הרטנית מראש. מה זה אומר? IDI הוא בדיקת 3 ספרות למשל. הטלפון זה 3 ספרות, מקף, ועוד 4 ספרות. המרכיבות אלה יורדות לרוחוץיה של מיקום בדיסק, שהוא בדיקת 4 ספרות, שהוא גודל שהוא גודלה ב-IDI.

NoSQL

מאגרי מידע הקיימים נתונים במבנהים גדולים, מקורות רבים, נפחים משתנים, סוגים שונים. המידע אינו מאורגן לפי שיטה כלשהי, Unstructured.

בניגוד למערכות נתונים רגילים, זה מבוסס SQL.

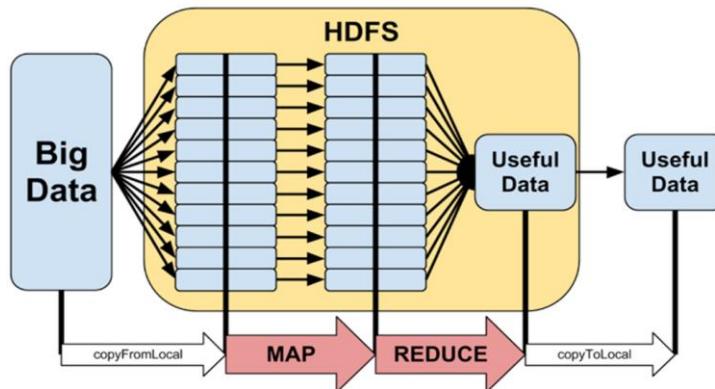
.Hadoop Distributed File System ,Document Oriented Data Base (XML) **תיגמאות**

```
<CATALOG>
<CD>
<TITLE>Hide your heart</TITLE>
<ARTIST>Bonnie Tyler</ARTIST>
<COUNTRY>UK</COUNTRY>
<COMPANY>CBS Records</COMPANY>
<PRICE>9.90</PRICE>
<YEAR>1988</YEAR>
</CD>
<CD>
<TITLE>Greatest Hits</TITLE>
<ARTIST>Dolly Parton</ARTIST>
<COUNTRY>USA</COUNTRY>
<COMPANY>RCA</COMPANY>
<PRICE>9.90</PRICE>
<YEAR>1982</YEAR>
</CD>
<CD>
<TITLE>Still got the blues</TITLE>
<ARTIST>Gary Moore</ARTIST>
<COUNTRY>UK</COUNTRY>
<COMPANY>Virgin records</COMPANY>
<PRICE>10.20</PRICE>
<YEAR>1990</YEAR>
</CD>
```

דוגמא ל-XML:
מואייפים מרקרים למסמן.
<Marker></Marker>
בגלל שאנו חנו שמים את המרקרים, כל מידע יכול להיות
באיזה אורך או פורמט שנרצה. אין הגבלה.
החזקת מידע שלא מאופין מראש.
המחיר הנוסף הוא כמובן המarkers עצם, אבל
הם בטלים בשישים מול גודל המידע.

הרצאה 4

כעת נדבר על Hadoop. Hadoop הוא מיזם המידיע מוחלט, שהמ מידע בformatter-blockים, שהם בעצם מפוזרים לחוליות. ישנו מספר עותקים לכל בלוק בשבייל גיבוי מידע זמין במקורה של נסילה. מבנה זה מאפשר עבודה בסגנון Map Reduce על העותקים השונים. מבחינת חישוב ענן וSQL, זיכרנו זה דבר אינסובי (תמיד אפשר להשאיל עוד ולקחת).



היתרון של Hadoop נובע מכך שהמידע מוחלט לא נכתב ממקבילי ולא סריאלי. הניתוח מתבצע ב-Java בתופסת כלים. חברות תעבור לעבוד עם hadoop כאשר יש לה הרבה מידע (בערך 10tera) או שאליות שדורשות הרבה משאבי הhard disk. ההבדלים העיקריים אל מול SQL הם בסמכות הקוריאה והכתיבה, ובאחסון המידע.

כתיבה ב-SQL בהינתן שני מסדי נתונים של SQL, A ו-B. כאשר נרצה לכתוב מ-A ל-B נצטרך לדעת הרבה דברים. כמו מבנה השדות של A, מבנה השדות של B, איך להתאים מידע מ-A ל-B (למשל, תאריך אירופי לתאריך אמריקאי). בנוסף צריך שהמידע המועבר יוכל את כל סוג המידע ש-B מיפה לקבל, אחרת נקבל שגיאות.

קוריאה ב-Hadoop כאשר כתבים מידע, פשוט מעבירים אותו ללא כללים. כאשר רוצים לקרוא את המידע, יש תוכנת Java שמאפיינת את הקוריאה, ולא מאפיינים את המידע עצמו.

לSQL, ב-Relational Database, המידע מאוחסן ב-DB גדול. הטבלאות, מידע, קוריאה וכתיבה, הכל מוגדר מראש.

ב-NoSQL המידע דחוס, ומשוכפל לבסיס מידע קטן יותר. אמן זה תופס יותר מקום, אבל בגישה "מקומ אינסובי" זה לא משנה, והיתרון זה הזמן עיבוד מקבילי על המידע.

- גישות להעברת מידע-Hadoop:
- אם באחד השירותים לא תפרד באופן זמן, המידע ממנו לא יגיע. יש שתי גישות אפשריות:
 1. העברת המידע שכן יש לשאר השירותים. מתוך הנחה שהמידע החסר יגיע בסופו של דבר או שהוא שיש זה מספיק.
 - בסופו של דבר, הכל מתבצע במעטפת קוד Java, لكن אין בעיה לתוכנת קר שזה יכח עד קצט עד קבלת המידע או לנסות להשיג את המידע החסר משרת אחר.

גישה להעברת מידע- RDB

מסדי נתונים רגילים יעבירו מידע אך ורק אם הוא מושלם. יש להזה שימושים (כמו בבנק).

יש כלים, כמו Hive של פיסבוק שמאפשרים כתיבת שאלות בSQL מעל Java או Hadoop, ובעצם מביאים את הלוגיקה SQL לעולם המבזיר. יש גם שירותים שמאפשרים שאלות בשפות חופשיות יותר, שמתרגמות JAVA ו- Hadoop.

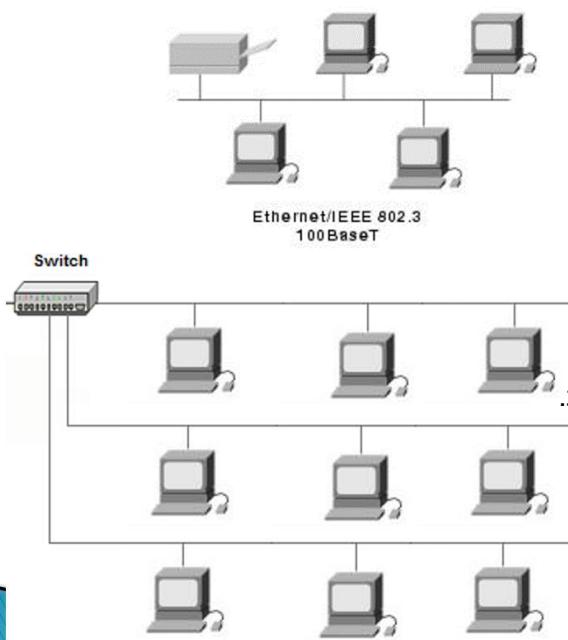
רשתות ותקשורת מחשבים

התקשורת בין דברים לבין עצמם ובין אנשים יוצרת הרבה אפשרויות שונות של ייצור מידע. מידע זה מאפשר לתחקoot וללמוד תהליכיים שונים. כמו, מעקב אחרי חוליה משוחרר מבית חולים, התנהגות של נהגים ע"י סנסורים, סטטיסטיקה ועוד.

סוגי התקשורת יכולים להיות יחיד לחיד, יחיד לרבים, רבים לאנשים, סנסורים ומערכות מידע. האינטרנט, כפי שהוא מתוכנן כיום, לא יודע להתמודד עם כמויות תעובה גדולות כמו אלה שמצוות. כ-75% מהתעובה באינטרנט היא בתחום Data Centers. עומס של תעובה עצומה בין יחידות הקצה לבין עצמן ולבין לקוחות להפעלה ויתוח המידע. בעתיד יוצר עומס רק על התקשורת ועומס רק ב-Data Centers. עומס של תעובה עצומה בין יחידות הקצה לבין שני סוגי תעובה: שוטפת או בפרצים (burst), כמו שכולם מתחברים בעבר לנטפליקס אז יש פרץ של תקשורת שידור).

דרכי ההתמודדות אפשריים:

1. תכנון נכון.
2. ניהול תקשורת-B-Trees שונים, כמו fog, שזה ענן קטן ונמוך יותר. למשל מעין ענן קטן לאזרם מסויים ואז רוב הבקשות של האזרם יונחו עמו fog ולא יעלו לאינטרנט.
3. תכנון Data Centers מבוזרים.
4. ניהוליעיל של Web caching.

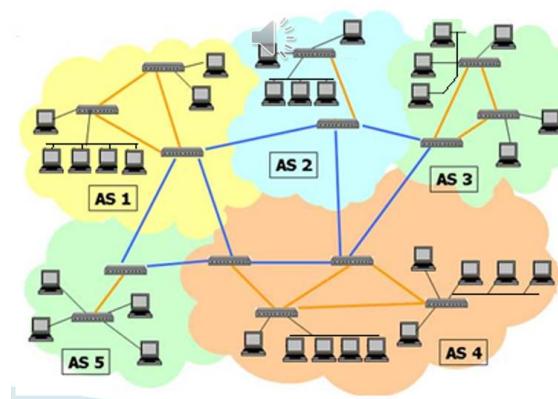


מבנה האינטרנט- רשת מקומית
אתרנט-יש פה 5 יחידות שמחוברות על בס. כל יחידה יכולה לשלו
ביטים על הבס, אבל רק יחידה אחת בכל רגע יכולה לשדר. הכל מתנהל
לפי פרוטוקול כאשר כל יחידה יכולה להקשיב. ההבדל בין התקן האינטרנט
لتקן IEEE 802.3 זניח (ביט וקצת נוספת) ולכן תומכים בשניהם. זהו רשת
מקומית. מספר רשתות מקומיות ניתן לאיחוד ע"י Switch, שזו קופסה עם
חומרה חזקה ותוכנה חזקה שמודעת לנכון הودעות בין רשתות LAN קטנות.
לדוגמא, אם יחידה בשורה הראשונה תשדר לשורה השנייה, ויחידה
בשורה השנייה תשדר לשורה השלישית, זה יבוצע במקביל.
אבל אם יחידה בשורה הראשונה תשדר ליחידה בשורה השלישית,
הסוי' ידע לסנן את זה בכר שיפסיק שידורים אחרים לשורה השלישית.
Extended LAN

רשת ה-WAN

כמו סוויצ'ים שմדברים אחד עם השני, מוצעים את זה דרך נטב (ראוטר). זה מחשב ליחידות שלא נמצאות באותו LAN לדבר אחת עם השניה.

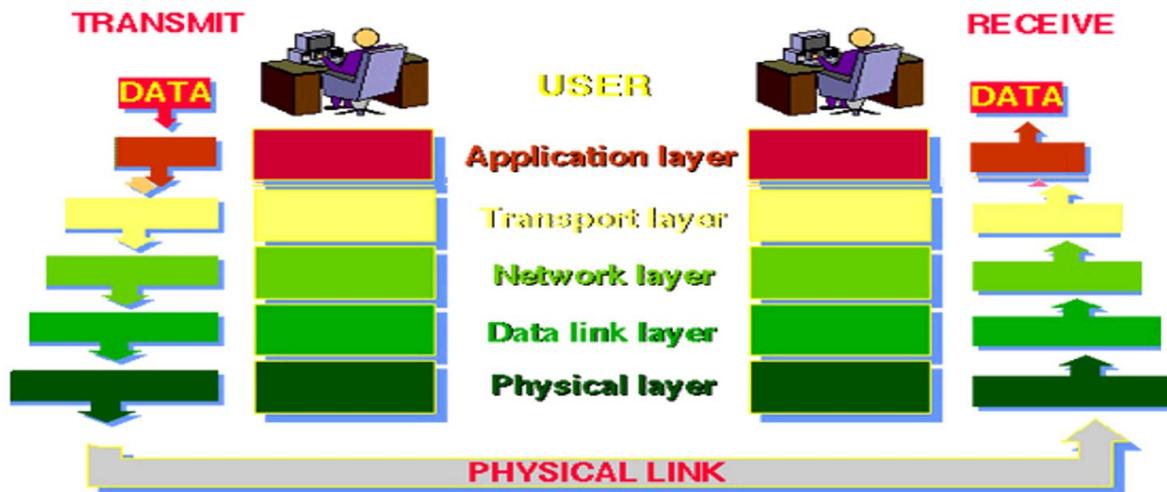
אז מה זה בעצם האינטרנט? זה חיבור של נטבים. ואז כל יחידה יכולה לדבר עם כל יחידה אחרת דרך נטבים שונים. האינטרנט נהיה גדול מדי לשכיב להסתמך רק על תקשורת נטבים, ולכן חילקו את האינטרנט למיצוקות אוטונומיות (AS). בעצם, כמה נטבים יוצרים AS. בתחום AS יש שני סוגי של פרוטוקולים: פרוטוקוליים פנימיים שמאפשרים לנטבים לדבר אחד עם השני בתחום AS שלהם. בנוסף, נטבי גבול (border gateway) הם נטבים שיודעים את הדרך החוצה מ-AS ולהיכנס ל-AS אחר. מרגע שנכנסו ל-AS חדש, שאר המסלול יהיה לפני הנטבים המקומיים. נטבי גבול רק מאפשרים לחצות AS, אבל לא מעבירים את המידע את כל הדרך, רק את הגבול בין AS.



כਮון שיכולה להיות המונע בעיות כאשר משדרים מידע, במיוחד כשהוא חוצה הרבה רשתות. איבוד מידע, אמינות מידע, איך לשחרר תמונה דחוסה, איך לשדר מחדש מידע וכו'. لكن תוכנת התקושרת מוחולקת למודולים שנקראים שכבות (layers). כל שכבה עוסקת בפתרון בעיות ספציפי. של שכבה מעל, פותרת בעיות אחרות אבל נסמכת שהשכבה מתחתיה פתרה את הבעיה שלה.

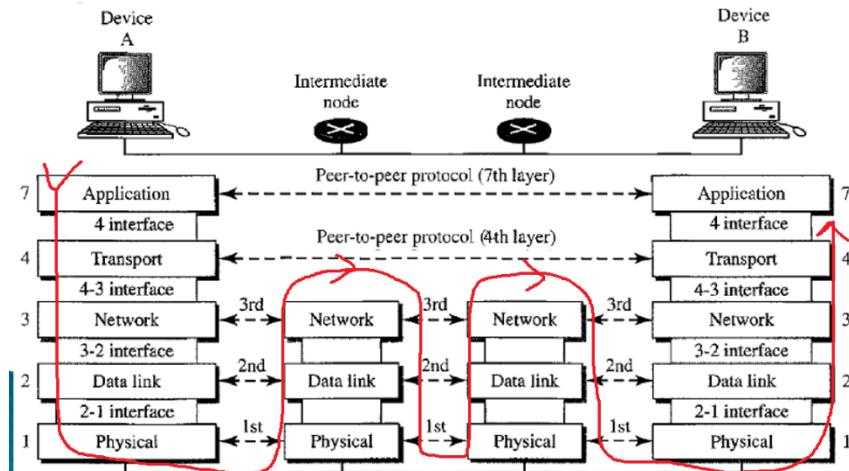
בעיות לדוגמה: אפיון 0 ו-1 פיזיים, זיהוי שגיאות ותיקון, מניעת התנגשות בשידורים הדדיים, ניתוב הודעה בaczורה נכונה על מחשב היעד, תמייהה בהצלחה השידורות ובאפשרות לקבל הודעה של מספר אפליקציות.

מעבר ונסביר על השכבות.

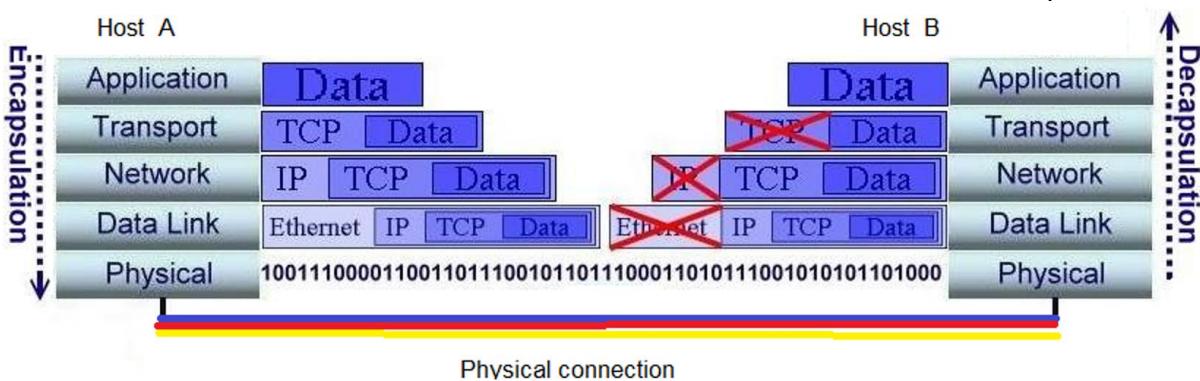


- Physical - חומרה, תדרים. אחראית על שידור הביטים.
 - Data Link - לגנות ולתקן שגיאות. להיזהר ולהימנע מהתנגשויות על קו משותף.
 שתי השכבות הללו, מספקות כדי לחבר מחשבים ב-LAN.
 - Network - תdue לקשר בין מחשבים בכל האינטרנט, מחשבים שנמצאים ברשתות שונות. מכירה את הטופולוגיה של המקור, נסמכת data-link עובדת טוב ותdue לשולח את המידע אל היעד.
 - Transport - לדאוג לעומסים באינטרנט ולדואג לעומסים בין תחנות. סמכת על קר שה-network יודעת לאיפה צריך להעביר את המידע. תdue לעשות שידורים חוזרים אם צריך. תיאום ביטים (האם זה מיועד למייל? האם לכרכום? האם ל��区?).
 - Application - האפליקציות. יכולות להיות אפליקציות רגילות או שמקבלות לפי פרוטוקולים מיוחדים כמו FTP או HTTP.
 הקשר בין שכבות מקבילות (למשל Network אצל מקבל ואצל השולח) הוא ששתייהן צריכה לתמוך באותו פרוטוקולים.

הקשר בין שכבות
 כל שכבה מדברת עם זו שמתחילה וזו שמעליה לפי API מסוים.
 שכבות 4 ו-7, לאחר הקישור, יכולות לדבר ישירות בין המקור ליעד.



מבנה חבילת התקשורת

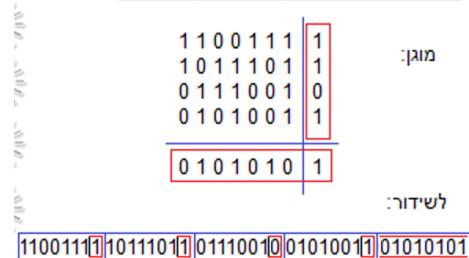
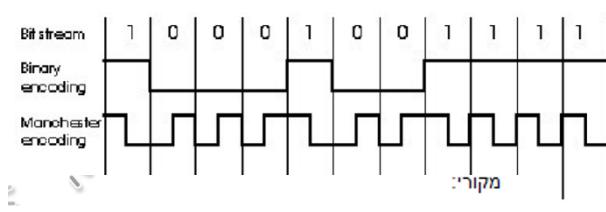


ברמת העיקרון, כל שכבה מוסיפה עוד header לקובץ. כשהميدע מגיע לצד השני, הוא מתחילה למחוק את headers אלה. ככה הוא יודע בכל שלב האם הוא צריך להמשיך לפענוח את המידע, או לעזור כיון שהميدע לא מיועד אליו ולהמשיך לשולוח אותו ולאן.

בקודנה מעניינת: רמת ה-Link מסיפה חלק מהميدע בהתחלה וחלק מהميدע בסוף של הקובץ, כדי לאפשר קראינה יותר מהירה ולשפר ביצועים. לכן, ברמת ה-DL, לעיתים קוראים לחבילה frame.

תפקידי השכבות

השכבה הפיזית, משדרת את האינפורמציה על הרשות, לפי קידוד מסוים. למשל קידוד מניטסטר, שדורש שבכל בית יהיה שניי במתח (אם הביט זה 1, אז המתח יהיה גבוה ואז נמוך, והופך אם זה 0), כדי לתunken שגיאות שיכולות להיווצר בכלל אי סyncron מדויק של שעונים.



שכבה ה-Data Link דואגת לתקינות השידור. מוסיפה מידע בקרה לגילוי או תיקון שגיאות, ואם צריך עשויה גם שידורים חוזרים. למשל, בית זוגיות דו מימדי.

שכבה הרשות, תפקידה להעביר חבילות תקשורת מתחת המקור אל תחנת היעד. דורשת ניהול מיפוי של הטופולוגיה של האינטרנט, קבלת החלטה באיזה מסלול יש להעביר חבילה (לרוב, לפי מספר הקשרות המיניימי'), ניתוח חבילות לפי כתובת IP ומחליטה לאן לשולוח, קישור בין רשתות פיזיות שונות, ופירוק והרכבת של חבילות תקשורת על פי מגבלות (למשל, אם מרשת אחת חבילה היא בגודל X וברשת היעד חבילה היא בגודל Y, אז צריך לדעת להתאים).

שכבה התובלה, דואגת לתקינות המידע ולהעבրתו לאפליקציות ביעד. שכבה האפליקציה משתמש במידע עבור שימוש כלשהו, ולעתים עבור פרוטוקול אפליקטיבי כמו HTTP.

הרצאה 5

כתובות זהה מספר בינהר שמאפיין את תחנת המקור ואת תחנת היעד. כרטיס הרשות מażin לקו (אופטי, נחושת, אויר) יוכל לקרוא את כל ההודעות שמתכולות אליו אבל יבחר לחתך רק את ההודעות שכתובות היעד שלון זהה לכתובות שצרכוה או מוקנפת לכרטיס. או בקרים מיוחדים, Multicast (שידור לקבוצה). יש כתובות שקובצתן מסויימת מתחשפת בכתובות הزادת וכל מה שימושה אליה, כל מי שבקבוצה יזכה את זה, -Broadcast (שידור לכלולם), לרבות הכתובת הزادת מורכבת רק מ-1.

כתובות MAC

צרובות בכרטיס התקורת, מאופיניות ע"י 48 ביטים (למשל B8:9B:41:80:6F:45) כאשר 24 הביטים הראשונים זה יצרkan הכרטיס. יצרן שגמר את כל ה-²⁴ אפשרויות שלו, פשוט יקבל כתובות יצרן חדשה.

כל כתובות של כל כרטיס רשות ייחודית בעולם!

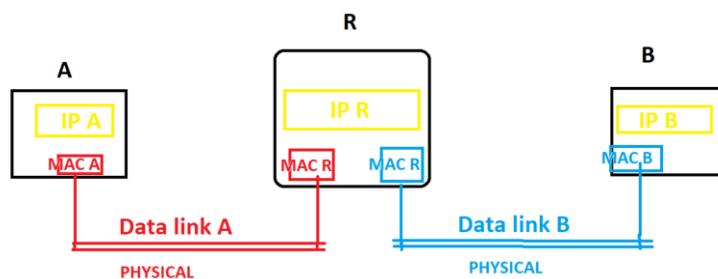
הכתובות האלה נראות אותו דבר גם באתרנט וגם בוIFI וכן פיתוח תוכנות לפי הפרטוקולים שלהם יחסית קל. (לא צריך לבצע התאמות).

הכתובת הزادת היא בפועל כתובות LAN, וכך כל התחנות שוממות כל שידור, וכל תחנה בוחרת את ההודעה שמכונת אליה. בפועל, מחשב מדבר ברמה 2 בכתובת MAC (בLAN שלו) אבל מכיוון שהאינטרנט מחובר דרך נתבים, והמעבר בין נתבים הוא קפ"יות, ברמה 3, המחשב מדבר בכתובת IP (מדובר על כל הכתובות באינטרנט).

כתובות IP

כתובות לוגיות (ולא צרובות בכרטיס, אין פה כרטיס באמת) בפורמט זהה עברו כל היחידות המוחברות לאינטרנט דרך פרוטוקול IP. גרסא 4 נפוצה, קיימת גם גרסא 6. הכתובות מקודדות ע"י מעטפת קוד באופן ידני, אבל רוב הפעמים באופן אוטומטי ע"י DHCP. מקבלים את IP בדרגה 3, באופן וירטואלי לחלוין. אין תלות באיזה רשת נמצא או איפה מכשיר. לאחר מכן, כשיורדים לדרגה 2 חוזרים לדבר עם כתובות MAC שיכולה להיות שונה אם זה אתרנט או רשת סלולרית. כל מחשב שיש לו כתובות MAC מקומית, יש לו כתובות IP גlobilit.

IP מאפשר לחבר רשתות מטיפוסים פיזיים שונים או שווים ע"י נתבים שמחברים בין רשות לרשta.

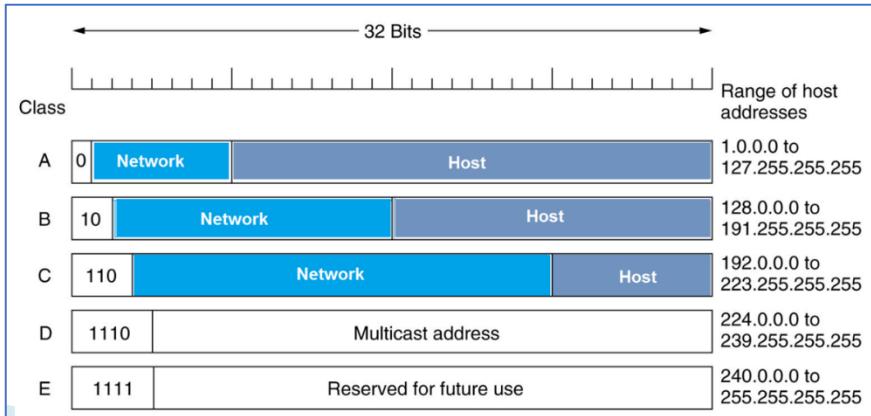


יש פה שתי יחידות (A,B) שמחוברות דרך אותו נתב. לכל אחד מהיחידות יש כתובות IP (שכן כולן מדברות עם האינטרנט). בנוסף, לנtb יש כתובות MAC ברשת A וגם כתובות MAC ברשת B.

IP של A רוצה לשלוח הודעה לIP של B. הוא יוצר הודעה ורושם שהיא מיועדת לIP ברמת IP (רמה 3). כת רמה 3 מבקשת רמה 2 לרשום הודעה לנtb ברמת MAC בשביל של B שזה יצא לאינטרנט. A MAC A יעביר הודעה ל- R MAC R, שיעביר את הודעה ל- R IP. R IP יזהה שה הודעה מכוונת לIP והוא יבקש מIP של רשת B שישלח הודעה ל- B MAC B. MAC B מעביר את הודעה ל- B IP שמקבל אותה. נשים לב שלשליחת ההודעה, ברמה הלוגית היא בכתובות IP, אבל ברמה הפיזית דרך MAC.

כתובות IPv4

כתובת IP גרסה 4 בנויה ממיליה של 32 ביט שמחולקים ל-4 חלקים של 8 ביטים. הכתובת מחולקת ל-ID Net ID ו-ID Host. ID Net זה כמו שם המשפחה, מקדים את כל המחשבים שנמצאים באות הרשות. ID Host זה הכתובת הlokality של כל מחשב. החלוקה של ID Net ו-ID Host היא לא קבועה, זה תלוי ב-Class של הכתובת.



על E ו-C לא נרחב. בהתאם ל-class, הכתובת של ID יכול להיות 8, 16 או 24 ביטים. יכול להיות 8, 16 או 24 ביטים. לפיקוד על כל class, נתב יוכל לדעת מראש אם המידע מכון אליו או לא (למשל, אם הוא ברשות של 8 ביטים אבל ההודעה מיועדת לרשות של 16 ביטים. את זה הוא יכול לגלו מידית לפי הкласс). בהתאם, מספר הביטים למחשב יכול להיות 16 או 8. כאמור, ב-A Class אפשר לשימוש ברשות אחת 2^{24} מחשבים.

	כמות רשות אפשרית	כמות מחשבים ברשות
class A	$2^7 - 2 = 126$	$2^{24} = 16777216$
class B	$2^{14} - 2 = 16382$	$2^{16} = 65536$
class C	$2^{21} = 2097150$	$2^8 = 256$

חברות "מקורבות לצלהת" כמו IBM ו-T&T קיבלו כתובות A. הטכניון קיבלו B.

חברות יותר קטנות קיבלו C. לצורך העניין, הווIFI אצלי בבית הוא C. הורדנו 2, כי הכתובות שהכל 0 או הכל 1 שמורות.

איך ניתן לחלק 32 ביטים? נחלק אותם ל-4 מילימים של 8 ביט וונשים נקודה.

1100000010101000000000111101001

11000000.10101000.00000001.11101001

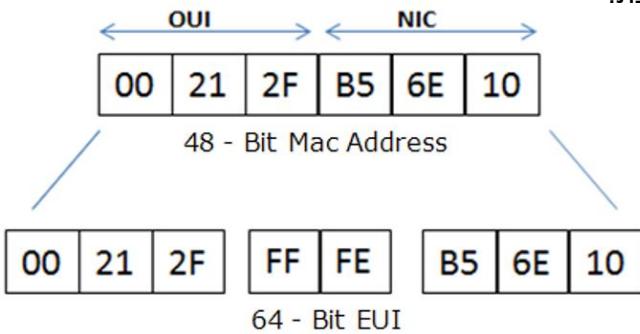
לאחר מכן, נהפוך כל מיליה לדסימלית. להתרען על בינהו לדסימלי.

192.168.1.233

סה"כ כתובת IP גרסה 4 יש מראה כמו 2,113,658 רשותות, שכמוון זה לא מספיק.

כתובות IPv6

ייצוג ע"י 128 ביטים. מיוצג ע"י 8 ספרים HEX בני 4 ספירות. משמשים 0 מוביילים, ובולוקים רק של 0 גם משמשים. FEFB:2BC1:0000:0000:0000:DAA1:0063:0000:0000 → 2BC1:0000:3268:DAA1:0063:0000:0000:FEFB הגרסאות הראשונות מנגננת את העבודה ש-MAC הנקודות, ובעזרת כתובת MAC בונה את הכתובת. כתובת MAC (ביחד עם תוספת בשביל להשלים) תהיה ה-LSB 64 של הכתובת.



הבעיה: רוב הנטבטים לא תומכים בתקן זה, ולכן יש מצב שהודעות לא יגיעו לידי שלחה.

Network Address Translation

כל שטחן כתובות IP, נועד לפחות את מצוקת הכתובות. מוגדר ב-RFC-2663.

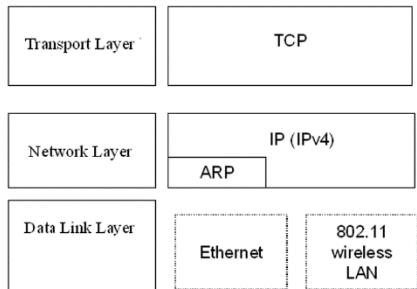
מוקם בקצתה של רשות מקומית (יכול להיות ממוקם קצר במחשב של הנטב), ומחזיק אוסף של כתובות IP חוקיות שמקצות לרשות המקומית. למשל, עבור רשות מסוימת מוקצות X כתובות, אבל זה פה X3 מחשבים. אז לכל מחשב ברשות המקומית תהיה כתובה IP "אילו" חוקית, על מנת שיוכלו לדבר אחת עם השניה. כאשר מחשב ירצה לשולח הודעה החוצה, NAT יקצתה לו כתובה אמיתי רק בשבייל שליחת ההודעה, וכשתחזיר תשובה, הוא ידע לאיזה מחשב ברשות המקומית ההודעה מכוונה.

בעצם, ל-NAT יש בנק כתובות שקטן מספר המשמשים תחת הנהנה שלא כל המחשבים מתמחברים לאינטרנט בו זמינים. עוד פתרון חח"ע הוא להשתמש בכתובת אחת עם המרת ports.

לסיכום, איך עוברת הודעה?

- ▶ באינטרנט שלוחים מכתובת ה- IP של המקור לכתובת ה- IP של היעד.
- ▶ בפועל, תחנת המקור שלחתה מכתובת ה- IP שלה, דרך שכבה ה- MAC Data Link שלה לכתובת ה- IP של הנטב שלה (ומזכירה בחבילת ה-IP מה כתובתה ומה כתובת היעד באינטרנט).
- ▶ הנטב שלח את ההודעה (הוא רק נותן שירות ומשאיר את הכתובות ב- header של ה- IP כמו שהן) דרך שכבה ה- Data Link שלה לכתובת ה- MAC של הנטב הבא.
- ▶ הנטב הבא (אחרון כאן בדוגמא) שלח את ההודעה דרך שכבה ה- Data Link שלה לכתובת ה- MAC של היעד.
- ▶ שכבה ה- IP ביעד תזהה שזה עבורה לפי כתובת היעד ב- IP.

הרצאה 6



Address Resolution Protocol

מתרגם כתובות IP לכתובות MAC. בtower אותו LAN, כאשר שכבת רשת תשלח הودעה לIP אחרת, ה-IP של היעד יתורגם לMAC ע"י ARP. המידע זהה יועבר לדינמי שידע לאיזה כתובות לשולח את המידע. כאשר רוצים לשולח הודעה למחשב ברשת אחרת, ARP יביא את כתובת MAC של הנטב.

תחנת המוקор שואלת את ה-ARP: מהי כתובת ה- MAC של הנטב שלי (שmagder אצלי, בכתובת IP שהוא Z.Z.Z.Z)? ARP: כתובות ה- MAC היא tt:tt:tt:tt:tt:tt.

תחנת המוקור: שלוחת את ההודעה לכתובת ה- MAC של הנטב (ומזיכה בהביבת ה-IP את כתובות ה-IP שלה ושל היעד באינטרנטן). נתב ראשון (אחרי שמחה לאיזה נתב לשולח) שואל את ה-ARP: מהי כתובות ה- MAC של הנטב הבא (שmagder אצלי, בכתובת IP שהוא y.y.y.y)? ARP: כתובות ה- MAC היא zzz:zzz:zzz:zzz:zzz:rrr.

נתב ראשון: שלוח את ההודעה לכתובת ה- MAC של הנטב השני (ומזיכר בהביבת ה-IP את כתובות ה-IP שלתחנת המוקור ושל היעד באינטרנטן). נתב שני (אחרי שמחה שתחנת היעד ברשת המקומיות שלו לפי כתובות ה-IP) שואל את ה-ARP: מהי כתובות ה- MAC של תחנת היעד (שmagderת אצלי, בכתובת IP שהוא x.x.x.x)? ARP: כתובות ה- MAC היא www:www:www:www:www:www.

נתב שני (אחרון כאן בדוגמה) שלוח את ההודעה דרך שכבת ה- Data Link שלה לכתובת ה- MAC של היעד. שכבת ה- IP ביעד תזהה שזה עברורה לפי כתובות היעד ב-IP.

איך ARP עובד?

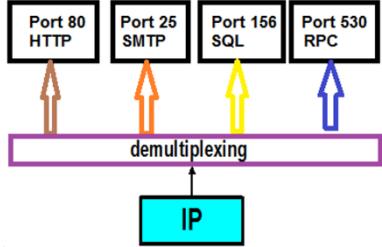
תחנה שרצה לדעת את כתובות MAC של כתובות IP מסויימת שלוחת הודעת Broadcast מיעודה שידועה בתור הודעת ARP (כל התחנות באותו LAN, כתובות היעד כולן). רשומות הכתובות הפיזיות והלוגיות של המוקור, כתובות לוגיות של היעד, ומקומות ריק (בשביל הכתובת הפיזית של היעד).

תחנת היעד תזהה שזה עברורה והיא תטפל בזה. היא תמלא בהודעה את הכתובות הפיזית שלה ושלוחת חזרה למקום, ועכשו הטבלה של השולח מעודכנת. באותו מחיר, גם הטבלה של היעד מעודכנת (הר' היא קיבלה כתובות לוגית ופיזית של המוקור). הטבלאות נשארות מעודכנות ל-20 דקות. הן נשארות מעודכנות בשוביל לא לחפש עוד פעם דברים שכבר גילינו, אבל לפחות עדכניות. כתובות IP עלולות להשתנות בזמן זהה.

שכבה התובלה, *Transport*

הפרוטוקולים העיקריים הם *TCP* ו-*UDP*.

UDP עוטף את *P/I* בחבילת ביטים שמאפשרת גילוי שגיאות, ריבוב אפליקציות. איך הוא ידוע לרבות אפליקציות?



אם למשל מחשב רוצה לשלוח ל-*HTTP*, אז מה *TCP* שלו הוא שולח בפורט 80, ו-*TCP* הידע יזהה את זה ויעביר את זה לאפליקציה שמשתמשת ב-*HTTP*.

איך בניו *TCP*? מה הוא מוסיף?

1. הגעת חבילות לפי הסדר

בעזרת השדה של *sequence number* הוא מציין את מספר החבילה הנוכחי, ובעזרת *acknowledgement number* הוא מציין את מספר החבילה הבאה שהוא מצפה לקבל.

2. הגעת מידע ללא איבוד

בעזרת *checksum*. שדה שבו מבצעים מניפולציות על הביטים ולפי התוצאה אפשר לבדוק. בדומה לביט זוגיות. בנוסף שדה זה מכיל גם את כתובת *P/I*.

3. שמירה על עומס המקלט ע"י דוא"ל

בעזרת *window*. השדה הזה מתאר "חלון" של כמות חבילות שאפשר לשלוח בקצב אחרி השניה. אם המקלט עמוס, הוא ישלח דרך הרודה בקשה להקטין את החלון.

4. שמירה על עומס האינטרנט ע"י גילוי עומס והורדת קצף

ע"י בקרת הצפה (*congestion control*). לכל *TCP* שיש, יש אלגוריתם מבוצר שעוזר להם לשמור על עומס של כל האינטרנט. כאשר יש עומס על האינטרנט, והבאפרים מתמלאים יש סכנה שחבריות שיגיעו יזרקו. *TCP* יזהה זאת ע"י א' קיבלת *acknowledgement* וכניסה למצב של *time out* בו הוא יעשה שידור חוזר. אבל מה הבעיה? שהה שידורים חוזרים של חברות שנזרקו, لكن גם השידורים החזירים יזרקו. וכך יהיו שידורים חוזרים לשידורים החזירים, גם יזרקו. ואנחנו תקועים בLOOP של שידורים חוזרים שמעמיסות עוד יותר על הרשת.

Slow Start & Congestion Avoidance

שני תהליכי שמנטים לגשות ולנוח כמה שייתר מהר עד כמה עמוסה הרשת.

בעזרת שני משתנים, (*CWND*) - *Congestion Window* ו-*SSThresh*.

הרעין הכללי - ככל שאפשר, מאייצים את קצב השידור. עד שגיעה לנקודת מסוכנת ואז ניזהר ונוריד את הקצב.

בהתב楼下 מוזריה לנמלים:

kan namlim la sholch at kol namlim shlo lo lchafsh mazon, achraet han yomotno. mspur katan shel namlim nashlch oam han chzro um mazon sholchim uod uod, bishorot mosedorot. am pataom kvara msho (nemar haocel, lid mutzben um zcachit magdlat) sholchim pchot v pchot.

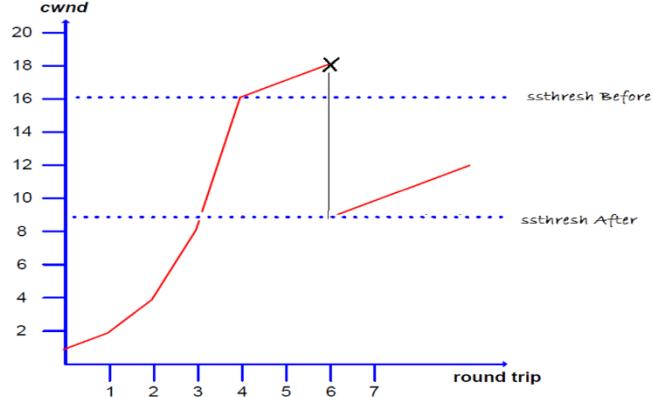
Slow Start

נניח, שכל חבילה בעצם מתנהגת כמו 2 חבילות: שליחה וקבלת *Ack*. נגיד *round trip* בתור ממשך הזמן בין רגע השידור עד קבלת תשובה מהצד השני. لكن, אם בזמן 0 נשדר חבילה אחת, בזמן 1 שידרנו 2 חבילות. בזמן 2 שידרנו 4, בזמן 3 שידרנו 8 חבילות (למה? שידרנו 4 חבילות ועוד 4 *ack*). לכן אנחנו מושרים בקצב 2^x .

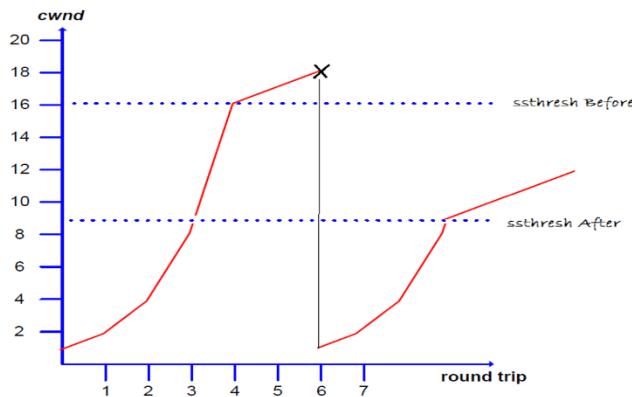
כמoven הקצבזה מאוד גדול, ולכן נגיד *ssthresh*, שכאשר נגיע אליו נתחיל להאט את הקצב החבילות בקצב 2^x ליניארי. למשל, (שידרנו עד כה 8 חבילות והגענו לחסם. אז בזמן הבא, נשלח עוד חבילה, 1, סה"כ בזמן 4 שלחנו 9. לאחר מכן בזמן 5 נשלח 10 (הרי זו *ack* של ההודעה הקודמת) וכן הלאה).

אם יש אובדן חבילות, מוריידים את גודל החלון. אם נכנסים ל-*time out* מוריידים את גודל החלון אגרסיבית. מתי מזיהים אובדן חבילות? הרי יכול להיות שהן מתעכבות. בשיטת *4 strikes and your out*:

אם קיבלנו 3 הודעות שהחביבה לא הגיעו, אז בפעם הרביעית נऋוץ עליה שנאבדה. כת, *TCP* מבין שאינטרנט عمום וצריך לטפל בה. לכן ישנה את שני הפרמטרים שלו: *CWND*-*SSTHRESH*. נסמן ב-*X* את ערך *CWND* בו זיהינו אובדן חבילות, אז נגיד $CWND/X = SSTHRESH$. נתחיל מחדש לשדר ליניארית החל מערך $CWND/X = X$. רק שעתה, זה יהיה ליניארי אבל עם שיפור יותר מאשר המפעם הקודמת. כך נמשיר את התהילה שוב עד שנגיע לאו ושוב נאבד חבילות, וכל פעם נשדר ליניארית בקצב יותר קטן עד שנמצא את הקצב נכון.



ומה קורה ב-*time out*? במצב הזה יותר קשה, כי אנחנו אפילו לא מקבלים הודעה שהחביבה לא הגיעו. לכן הפתרון פה הוא יותר דרמטי. תיקון *SSTHRESH* יהיה זהה, אבל במקום להתחיל עם *CWND* ליניארי, נתחל את התהילה מהתחלת, ונמשיר אותו אופן.



שאלה: האם ה-*SSTHRESH* יכול לעלות? כן! ע"י שיפור החיבור (כמו לוחץ רענן כשאטר תקוע). בפועל, אם בעלייה הלינארית נתחל לאבד חבילות בערך יותר גבוה (למשל $2 + X = new_X$) אבל נמשיר בתהילה אבל נבצע $SSTHRESH=X/2$ ואז $X=new_X$.

מתקפות סייבר ב-DOS

- למה ייחידות DOS היא חשופת לפגיעה?
1. הרבה מאוד ייחידות מואתו סוג. אם פרצת לאחת, פרצת לכלן.
 2. נפוצות במקרים שונים בעולם.
 3. יחידה אחרת יכולה להיות בהרבה מקומות שונים במהלך חייה (למשל שעון חכם).
 4. בעיות זיכרון, CPU, יחידות חישות ולכך קשה לקיים אליה הגנה טובה.
 5. אמורות להיות זולות מאוד, ולכן יש זולן בעניין של ההגנה.
 6. נמצאות בסביבה קשוחה, לתוכף יכול להיות קשר פיזי עם היחידות. (תשווה התקפה דרךIFI לעומת התקפה דרך USB)

התקפת DoS - Denial of Service

אוסף של מתקפות בקטגוריות התקשרות השונות. נועד להשבית מערכת ע"י העמסת המשאים שלה. המטרה הינה מניעת גישה למערכת שספקת שירותים למשתמש. בכך המשתמשים לא יכולים לקבל דבריהם מהמערכת והיא תהיה לא שימושה.

זה מתבצע בכך שהתקפה תדרוש משאב חשוב של המערכת באופן אגרסיבי ובעצם תעמיס עליו.

תוצאות ההתקפה הן:

1. משתמשים לגיטימיים לא יקבלו משאבי (המוחזקים ע"י ההתקפה) ולכן המערכת תהיה לא שימושית.
2. המערכת תגיע לירisa כולל.

דוגמא טובה לכך, זה בעונה 1 פרק 4 של ריק ומורטי, כשהם לכודים בסימולציה VR עם הרבה סוכני AI. ריק מתחילה להציג שאלות ותשובות יותר למערכת (החל מ-"כל הנשים להגיד היי" כל מי שמועל גיל 32 יששה כהה עם היד! כל מי שעם חולצה אדומה שיקוף על מעלה ולמטה" ועוד "כל מי שהשם הפרטיו שלו מתחיל ב- ל' והוא לא הייספני לו בمعال לפי המספר של השורש הריבועי של הגיל שלהם כפול 10") ואכן הם הצליחו לתקוע את הסימולציה ולברוח ממנה.

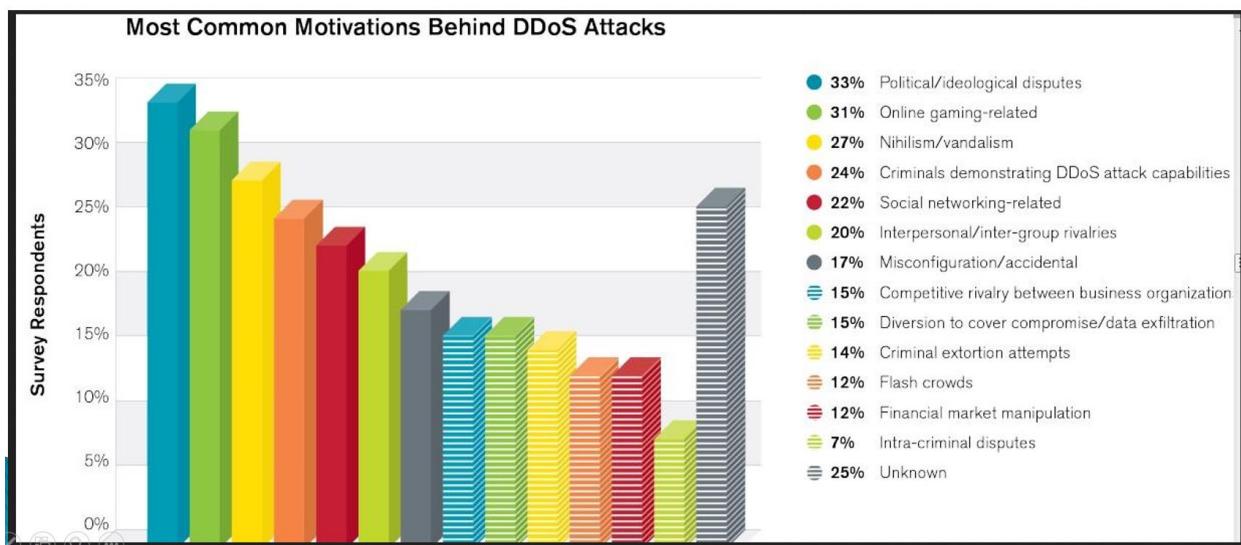
התקפה DDoS היא התפתחות של DoS, ונוסף לה Distributed DoS שמתבצעת ע"י מספר מערכות.

סוגים של התקפות DoS

Prevention- מניעת שירות. מטרתה למנוע משתמשים לגיטימיים לקבל גישה למשאב או שירות מסוים, ע"י העמסת אותו המשאב או הרשת שספק את השירות.

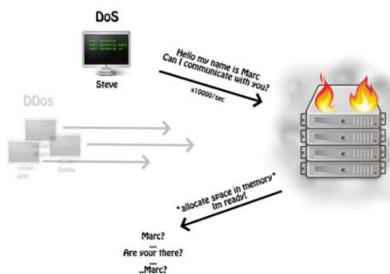
Disrupting- שיבוש התקשרות בין מחשבים כלשהם בתוך הרשות המותקפת (שרתיים, נתבים, קצה, ועוד).

Flooding- שליחה של מספר אדיר של חבילות查明יציות את הרשת, וכתוצאה לכך נוצר גודש בתעבורה הנכנסת ויוצאה מהרשת.





SYN/TCP Flood



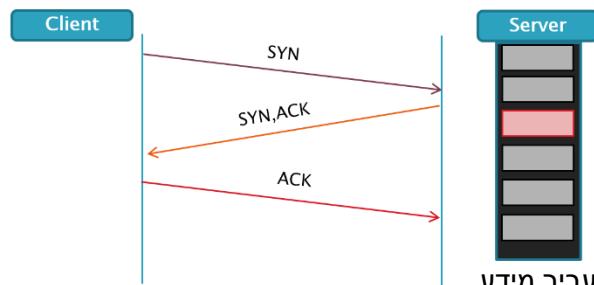
גישות שונות להתקפה:

1. התקפה מבוססת פרוטוקולים - SYN Flood

2. התקפה מבוססת רוחב פ-UDP Flood

3. התקפה ברמת האפליקציה - HTTP Request Attack

.Prevention UDP Flood זה הצפה בחבילות, 1-3 הן יותר מסוג של Prevention



TCP- Threeway handshake

קשר TCP בין מחשבים נוצר ע"י לחיצת יד משולשת.

השרות מהכח באזנה. ל��ווון מססה לצור חיבור ע"י

הודעת SYN. השירות עוברת למצב RECEIVED-SYN.

מה זה אומר? הוא קיבל בקשה לՏՆՉՈՒՅՆ. הוא יגיב בהודעה

כפולת ACK (קיבלתי את הבקשה ואני מאשר) SYN+ACK.

(אני רוצה להՏՆՉՈՒՅՆ איתה גם). בשלב זהה הוא מקצה

משאביו זיכרונו עברו הבקשה.

הלקוון קיבל את הודעת ה-SYN של השירות ויחזיר גם

הוא ACK. השירות יקבל ACK מהלקוון והחיבור יוקם וניתן להעביר מידע.

TCP SYN Attack ?TCP SYN Attack

התתקפה ברמת ה-TCP, מtabסס על לחיצת יד משולשת.

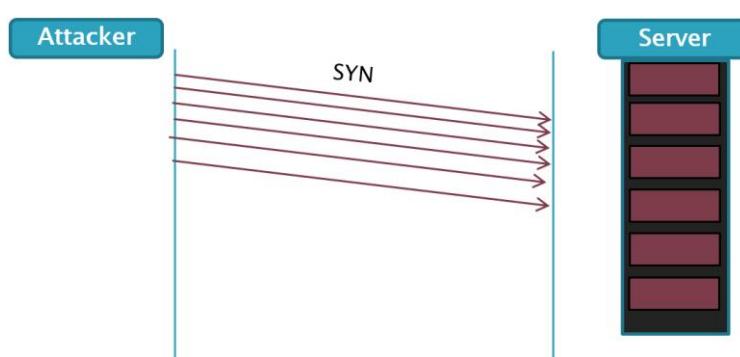
המטרה היא לסתום חיבורים: לגרום לשרת להקצחות הרבה משאבים ולגרום לכך שהוא לא יוכל להציג לחיבורים

לא ניתימים. התקוף, ישלח הודעה SYN, וכשהשירות ישלח הודעה SYN שלו, התקוף יתעלם ממנו וישלח הודעה

SYN חדשה. באופן זהה, השירות ייכה להרבה ACK שלא יגיעו בו בזמן יקצת עוד ועוד מקומות. לעיתים 1000 בקשות

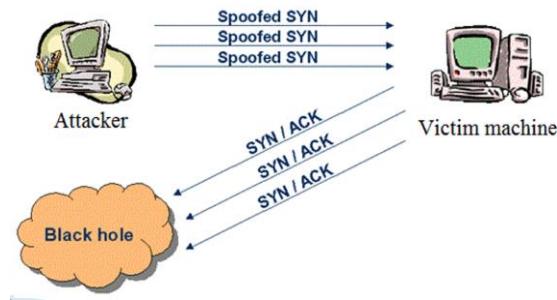
כolumbia בשנייה מ-1000 מחשבים שונים, תעמיס על השירות כך שמשתמשים לגיטימיים לא יקבלו שירות והשירות

יהיה בסכנת קריישה.



התקפת SYN Attack – IP Spoofing

התוקף יוזם מספר גדול של בקשות SYN מזויפת של IP. השרת ישלח SYN-ACK בחזרה לכתובות השגויות. גם אם ההודעה תגיד ליעד, היעד לא ישלח ACK כי הוא לא התחיל את התהליך. השרת מגיב לאווסף כתובות IP מזויפות שאין יעד עברו השרת. יש פה גם אלמנט של התקפה וגם אלמנט של התהבות של התקוף.



שיטות הגנה של SYN Attack

1. מגדירים מספר חיבורים מקסימלי עבור תחנה כך שהיא לא תקרוס מעומס בקשות. זה לא פותר את הבעיה שימושים לגיטימיים לא יקבלו שירות, שכן השרת לא יענה לאירוע מעבר לסף, אבל לפחות הוא לא יקרוס.
2. הגדרת פרק זמן מרעג שליחת ה-SYN-ACK שם הוא לא יקבל ACK מהצד היוזם את הקשר בפרק הזמן זהה, הוא ישחרר את המיקום שהקצה, יפסיק להמתין ויחזול את החיבור.
3. קבועת firewall או תוכנה על השרת שנרכזה להגן – ימוקם בכניסה לרשת הפנימית ומטרתו לזהות את התקפות האלה. מעין משגית. זהה כמהות וקצב בקשות לא חריג אז הוא יחסום ובכלל לא יעביר את זה הלא לשרת.
4. מצטום הטיימר של Received-Syn, בדומה ל-2.
5. ביצוע cache ל-SYN – לא מטפלים בבקשת כshan מגיעות, אוגרים אותן ואז אחורי כמה זמן מטפלים בהן. לא מקרים מיקום ולא מתחילה תהליך. התהליך יתבצע רק אחרי שיחזור ACK מהמשתמש. אם וכאש יחזור ACK ישלף המידע מהcache ויתבצע תהליך רגיל.
6. מחזר ה-TCP Half Open (הכי ותיק) – פותחים חיבורים רגילים, עושים סוג של תור מעגלי. אם חיבור במצב עובי (choked, מחכה ל-ACK) נכנס אותו לתור. התור יהיה מוגבל במקומות, נגיד 1000 מקומות, ובכך שיכנס הבקשה 1001 תזרוק את הבקשה הראשונה לחלווטין. (כנראה זאת שכני הרבה זמן במצב עובי).

מתתקפות רוחב פס – UDP Flood

החולשה של UDP שהוא connectionless, אין הקמת קשר מוקדמת. לכן אפשר להעמיס דרכו כמויות אדירות של חבילות ללא בדיקה. התוקף שולח חבילות IP עם UDP datagrams ופונים לפורטים אקרים. השרת בודק את האפליקציות (לפי הפורטים) ולא מוצא, ושולח בחזרה הודעת ICMP, יעד לא נגיש, destination unreachable. בעצם, התוקף מכריח את השרת להגיב לו והמערכת נעשית עמוסה. כמובן, אפשר לשלב את זה עם spoofing כדי להסתיר את התוקף וגם כדי שההודעות לא יחוzu אל התוקף ויתקעו אותו בחזרה.

הגנה UDP Flood

קובעים חומת אש לפני השרת. חומת האש יכולה באופן חופשי לחתט במידע הנכנס. ניתן להציג פורטים שאליהם לא מעבירים הودעות (ובפרט עבר פורטים שלא קיימים) וכן אין יסוננו ויזרקו בرمת חומת האש ולא יגיעו לשרת.

בנוסף, חומת האש יכולה לזהות מקרים חשודים (נפח או קצב גבוה מדי של חבילות UDP, מעקב אחריו חבילות ICMP למשל לבדוק אם חוות הרבה יעד לא נגיש). כמובן, שאפשר לתקוף את firewall עצמו.

התקפות ברמת האפליקציה - HTTP Request Attack

HTTP הוא פרוטוקול שיושב ברמת האפליקציה ומהווה את הפרוטוקול גלישה באינטרנט שלו.

תקשורות עם HTTP מתבצעת באמצעות שתי בקשות: GET, POST.

-GET - פעולה קלה יחסית, להיא תוכן סטנדרטי וטיטי, כמו תמונות וטקסט שנכתב והוא סטטי.

-POST - נועדה לדפי דינמיים, מכילה יותר פרמטרים וונטה לנצל יותר משאים של זיכרון ועיבוד.

לדוגמא, פירוט היתרונות לפי חדש באתר של הבנק. זה דורש מהשרת של הבנק לבצע חישובים ולהקנות זיכרון. התקפות מהסוג זהה, מנצלות בקשות GET וPOST לגיטימיות כדי לתקוף שרת או אפליקציית WEB.

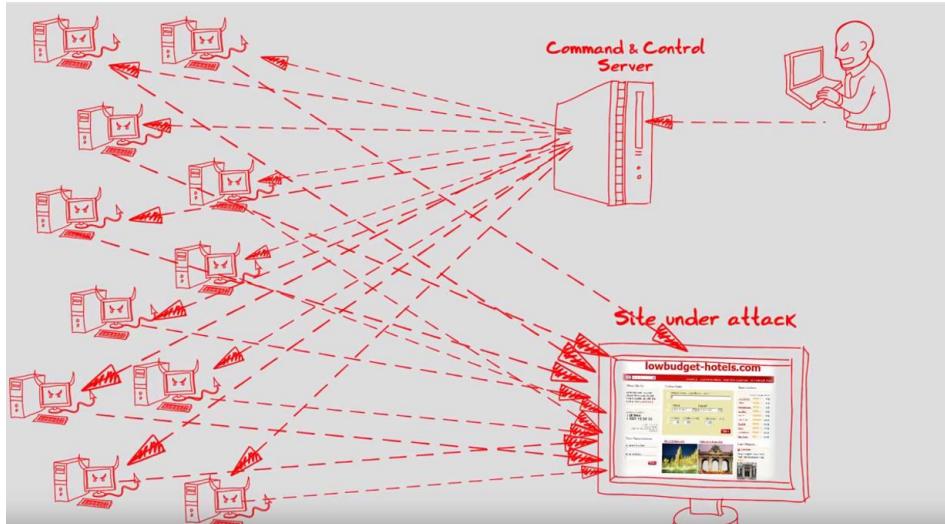
לרוב משתמשים בו-BOTNET, שהוא מספר מחשבים שהושתל בהם קוד זדוני. בעצם רשות של בוטים. ההתקפה תהיה אפקטיבית כאשר השרת יאלץ להקנות הרבה משאים עבור ההתקפה. אך לרוב מוצעים בקשות POST כי הן מראש יותר כבדות.

BOTNET ומתקפות DDoS

BOT זה תוכנה, שלרוב מותקנת באופן זדוני (מורוץ דרך מייל או מסתור בקבצים) או על פי רצון המשתמש. מסוגל להזיך פנימית למחשב או לשמש למטרות אחרות.

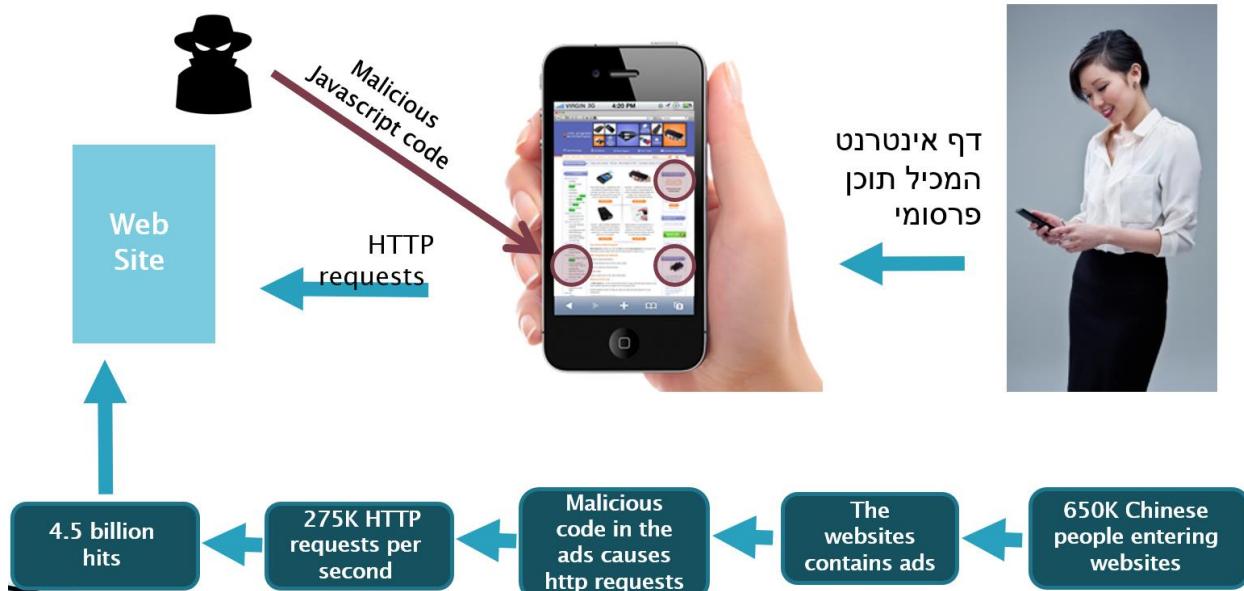
BOTNET היא רשות של בוטים שיכולים לבצע תקיפה יחיד. הבוטים נשלטים ע"י המחשב המדביק ומקבלים ממנו פיקודות. BOT MASTER הינו בעל השליטה הראשית, והוא יכול לשולט במחשבי התקיפה משרת מרכז.

כמובן שברכבי IoT, ההפצה וскопול של בוטים כאלה נהיה יותר קל ומהיר.



עוד קצת על DDoS:

DDoS הינה מתקפה של מספר רב של מערכות zdonyot נגד מטרה אחת, בדרך כלל קורה דרך Botnet שמנסים לבקש שירות מהיעד באותו הזמן.
מתקפות כאלה לא מנסות לגונב מידע או לפרק אבטחה, אבל פגיעה בזמן יכול לעלות לחברות כף רב, פגעה באמנות ובדימוי.
משתמשים לגיטימיים יעדו לשירותים של מתחרים בעקבות שירות שלא זמין.
בדרכם כלל מתבצעת ע"י אקטיביסטים או סחטנים.
לדוגמא, התקפה מ-2015 של 650,000 סمارטפונים סיניים:



לאחר 4.5 ביליאון בקשות, האתר של הבנק קרס.

- ▶ **מכשיiri Tso**, שמראש לא תוכנו עם מחשבה של אבטחה בהכרח, ה-
מטרה קלה מאד.
 - לפי סטטיסטיקות אחוז ההדבקות שנעשו בשנת 2018 עלה ב-203% לעומת 2017.
- ▶ **לדוגמה - BrickerBot**
 - נזקה שנועדה לגרום למכשיiri Tso להפוך ל-Bricks, או חסרי תועלת
 - לפי הערכות, BrickerBots הרסו כבר מעל 3 מיליון מכשיiri Tso.
- ▶ רשת של Bot'ים יכולה לפי פקודה להפעיל שרתים קרייטיים, למנוע גישה
משתמשים לאתרם מסוימים ולעוד הרבה מתקפות שונות.
- ▶ כיום, מנסים לשלב ליחידות Tso פרוטוקולים של אימות המחשבים
שאיitem מדברים, הצפנה של המידע הנשלח והבטחת שלמות המידע.

מתוקפה על עורך תקשורת בין שתי מערכות נפרדות, בו התקופף מתחזה לאחד מצדדי התקשרות. התקופף יכול לגרום לאחד מצדדי התקשרות למסור לו מידע רב, שייתכן שלא אמורה להיות לו גישה אליו. לדוגמה, תוקף מתחזה לסנסטור קורא טמפרטורה וגולם למוניטור לחשב שמכשיר מסוים מתחכם מאוד ובכך להשיבת פס ייצור שלם.

עוד דוגמא זה פריצה לרכב של Jeep. התגלתה פירצת אבטחה שאיפשרה לכך שלישי להתחזות לנגן עצמו, ועוד. אפשר לו לשנות על פונקציות בסיסיות כמו לחיצה על בלמים סיבוב הגה ושליטה על תאוצה. באופן כללי, ברכבים יש המון מקום לפירצאות. זה עקב שרכיב לא תוכנן במקור כאמצעי דיגיטלי וכל המחשב שנכנס אליו נכנס בהדרגה וబת אחת נפתח לאינטרנט, מה שלא נתן מספיק זמן לעבודה על אבטחה. בנוסף, תהיליך ייצור של אותו לוקח כמה שנים וכן הכל צריך להיות במחשבה שנים קדימה.

הנדסה חברתית – גניבת זהות ומידע (יש פה חזרה על עמוד 8 ותוספות) הגנה מפני גניבת זהות:

- ▶ **שיתוף של מינימום המידע שצריך באינטרנט.**
 - חשוב לדעת למי יש גישה לאיזה מידע
 - במקרה גישה שימוש
 - באיזו מדיניות משתמשים לצורך שמירה על מידע.
- ▶ **הצפנת המידע**
 - מבטיח גישה של אנשים מורשים בלבד למידע הרלוונטי.
- ▶ **שימוש באימות חזק**
 - קביעה למי ולמה מותר להתחבר למכשיר שלך, למידע שלך ולרשת שלך.
- ▶ **Authentication Multi Factor - אימות הדורש קומבינציה של אלמנטים לצורך קבלת גישה, בדרך כלל ע"י שני אמצעים או יותר.**
 - ע"י מהו שיש לך (לדוגמה, הפלפון)
 - מהו שאתה יודע (לדוגמה, סיסמה)
 - ומהו משלך (לדוגמה, טביעת אצבע).
- ▶ **שימוש בסיסמה שונה בשבייל כל מכשיר/משתמש.**
 - במקרה ותוקף מצליח לפרוץ למשתמש מסוים, הוא לא יוכל לפרוץ לכל שאר המשתמשים גוף-גוף.

הנדסה חברתית הפעלת מניפולציות על אנשים כדי שיחפשו מידע חסוי. לרוב המטרה זה אנשים ללא ידע באבטחת מידע, אשר יש להם גישה למידע הרצוי. בד"כ, הטעיה וניסיון לرمות את המטרה ולגרום לה לתת להם סיסמות לדברים רגילים. בנוסף, ניתן לגרום למטרה לתת גישה למחשב ולהתקין תוכנות זדוניות שיכלות לאוסף מידע ולתת גישה לתוקף.

לדוגמה, Phising. נתינת כתובות לאתרם אשר דומות לכתובות האמיתית (ולפעמים גם האתר נראה כמו האתר האמית). אבל ברגע שמנגנים מידע נשמר אצל התוקף.

סימנים זיהוי:

1. כתובות מייל/אתר שדומה מאוד לכתובות אמיתנה, עד כדי הבדל בתו בודד.
2. האצת הקורבן להיכנס לילינק כמה שיותר מהיר בלי לחשב.
3. אתרים שנראים ממש כמו האתר המקורי.
4. טעויות כתיב או רמה לשונית לא מתאימה או ירודה.

הסכנות בהנדסה חברתית

1. פריצה לדברים שמחוברים בעולם המטרה. בין אם זה כניסה לחשבון בנק האיש, או פריצה למכשירים יותר חזקים.
2. אובייקטים חכמים לא מוגנים יכול להיות שער לתוך אובייקטים יותר מוגנים בעלי מידע רגיש. לדוגמה: מוניטורים של תינוקות, שמרаш נמכוו בזול, היכלו backdoor עבור פורצים. הפורצים פרצו לצלמות וכיון המצלמות היו מחוברות לרשות הביתית היה להם יותר קל לחדר לרשות הביתית.
3. התקפות של הנדסה חברתית דרך DDoS מأتגרות את ביטחון המידע ופרטיות הסובבים אותם.
4. ללקוח יש שליטה על Devices בסביבתו. במידה אחד נפרץ, לפורץ יכול להיות גישה לכל הסביבה, לבית עסק למכוון ולמידע אישי.
5. פריצה אל הרשות הביתית תוך שליטה על שער חניה חמלי, טליזיה, אורות וכו'.

אם לפורץ יש גישה לרשות ולמחשב, יש לו גישה לכל המידע האישה, מה שיכול להוביל לגניבת זהות.

דרכי התמודדות עם הנדסה חברתית בעולם העסקי

1. חינוך העובדים והלקוחות
2. מדיניות סיסמאות חזקות והחלפתן כל פרק זמן מוגדר
3. שימוש certificates לכל מכשיר בנפרד, ובכל להבטיח שכמensch יתקשר רק עם מכשירים בטוחים אחרים, תוך שימוש בזהות מבוססת מפתחות להצפנה, אימות והבטחת נכונות המידע.
4. התקנת עדכונים, לצורך תיקון חולשות ופירצות.
5. התקנת Multi Factor Authentication.



חולשות בDOI – רמת הממשק

מיושן ושיימוש DOI גורמים להרבה ערכאים תקשורת. התקשרות דורשת ממשך. על מנת להבדיל בין משתמשים, הממשקים דורשים פרטי התחברות. החולשות העיקריות במשק הינם: 1. אונומראציית חשבון 2. אי נעלית חשבון. אונומראציית חשבון, זה כאשר עברו כל כניסה כניסה כושל מקבל תשובה מה לא הצלח (שם משתמש לא נכון), סיסמה לא נכון) ואין הגבלה על כמות הניסיונות. מכך מאוד על פריצה באמצעות brute force cooldown. בעילת חשבון, הגבלה על מספר פעמים שניתן לניסויים להתחבר, עד שהגישה לחשבון נחסמת או attack brute-force. שיטת ההתקפה העיקרית. מתקפה שמחשבת את כל צירופי הסיסמאות האפשריים בהינתן הגבלות. קלה מאוד לשיאן הגבלה על מספר הניסיונות, וכשהסיסמה יחסית פשוטה. לדוגמה, סיסמה של 4 ספרות, יש סה"כ 10000 צירופים. לא בעיה לכתוב תכנית שתנסה את כלם. סיסמא בת 8 תווים שח"יבת להכיל אות קטנה, אות גדולה ומספרה, זה כבר $64^8 = 27 * 27 * 10 * 10$, קשה.

חולשות בDOI – רמת האימוט

הרמה בה המערכת דורשת זיהוי ואיומות של המשתמש.

- דרישות עצמת סיסמא – כמו שהסבירנו לעל, דרישות לגבי אורק, סוג התווים והగבלות כמו איסור רצפים. DOI יכולים אין דרישות מינימליות על הסיסמא וכן קלות לגילוי ב brute force.
- דרישת החלפת סיסמא – בד"כ לא ממומש. בנוסף, לרוב מכיר DOI מגיע אם סיסמא default ראשונית ופשתה כמו 0000 או admin ובד"כ משאירים אותה! לכן לא צריך לעובוד קשה ולא צריך brute כדי לפרוץ.
- דרישות איפוס הסיסמא – כאשר יש לשחרר או לאפס סיסמא, אפשר לדרש אימוטות חזות ע"י כל אחר (כמו שכחתי סיסמא במיל, אז הוא שולח הודעה לסמארטפון). כמובן, דרישת חלשה בDOI.

חולשות בDOI – רשות הרשת/העברה המידע

- פורטי תקשורת פתוחים – יתכונו פורטים פתוחים שאין אתם תקשורת, או שמאזינים להם ומגיבים לקלט מהם. פריצה לפורט זהה מאפשרת שליפת מידע ממכשיר או שליחת פקודות שיקריiso את המכשיר.
- חומר בדיקת נכונות וकצוות פותחים בקלט – ניתן לשולוח פקודות שיכולות להפיל את המכשיר. למשל, fuzzing, הוספה מידע רנדומלי או מכון שיכל ליצור overflow buffer. היה נמנע אם הקלט היה נבדק.
- העברה מידע לא מוצפן – שליחת מידע מסווג בצורה של clear text או קידוד פשוט מאוד.

הרצאה 8

במחשוב ענן, האפליקציות והשירותים עבורים ל"ען". ענן זה חווות שירותים גדולה ומוגבה, גם פנימית וגם ע"י חוות נספנות. חוות כלל נמצאות במספר מקומות בעולם. כל הנושא התחל לפני בערך 20 שנה. החשיבה הייתה שלעבד עם תוכנות משרדיות כבדות מצרי משאבי מחשב וקבוצות תמייה טכנית בשביל להתקין, לתמוך, להחליף תוכנות, אז עדיף ל"הוריד" את התוכנה לכמה שעות, לעבוד ולשלם לפי שעה. התפתחות האינטרנט הפכה את הרעיון לגlobלי ופיתחה אותו ממשמעותית.

זהו צורת מחשב הנשנה על חלוקת משאבם פיזית או ירטואלית המועברת דרך הרשת, ללא שימוש ותחזקה של ציוד פרטי.

למעשה, מכל דבר שניתן להתחבר לרשת ניתן להשתמש בשירותי ענן. רוב האנשים משתמשים בענן אפילו מבלי לדעת את זה.

נקודה מעניינת: פרוטוקול בשם infiniband, מתעסק עם ה-SUS של המחשב. כמו שברשת מקומית מחשבים יכולים לדבר אחד עם השני, באינטראנס, רוב רכיבי המחשב יכולים לדבר אחד עם השני, ובפרט הזריםות והמעבדים. אז בעצם, במקום לדבר עם מחשבים, וכל מחשב עשו את החישוב הפנימי שלו, אפשר לדבר ישירות עם המעבדים והזרים וליצור מעין מגה-זרד של מעבדים זיכרון וליצור סופר מחשב. פרי פיתוחה של מלונקס.

תכונות המפתח בענן:

1. גמישות ואמינות
 - אפשר להשתמש רק בישומים ספציפיים להם אנו זקוקים ברגע הנוכחי, ולא לקנות חבילה תוכנות כבדה רק כי אנחנו צריכים שהוא אחד ממנו.
 - כמו גם כל אבטחת המידע מתבצעת בرمאה מאוד גבוהה ע"י הענן.
2. תשתיית משותפת
 - מאפשרת שיתוף של שירותי מידע בין משתמשים, כמו github
3. דינמיות של ה- Data Center
 - כח עיבוד וזיכרון מאוד חזק ומשתנה (אפשר להוסף ולהוריד), אלגוריתמים דינמיים היודעים להתמודד עם שינויים תכופים בדרישות העומס של הלקוחות. דינמיות של הרשת. ניתן לחבר תחנות ירטואליות ברשותות כולל הגנה בגיןה.

חוקרים בתחום מתמקדים בשאלת: מתי עדיף להביא אפליקציה לענן ומתי עדיף להביא מידע מהענן. מתי נעדייף להביא מידע מהענן ולחשב אצלו לעומת מתי עדיף לנו להתקין מראש את כל האפליקציה בענן ולחשב שם.



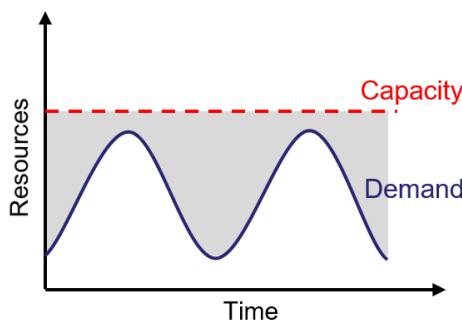
סוגי עננים

- פרטי – מוקם ומופעל באופן פרטי עבור ארגון אחד (מופעל מתוך הארגון או ע"י צד שלישי).
- קהילתי – ענן שיתופי עבור מספר ארגונים, עוזר להקטין עלויות הקמה והחזקה.
- ציבורי – ענן הפתוח לציבור הרחב, כמו google drive.
- ענן מעורב – שילוב של כמה מהסוגים לעליה. למשל חברת שומרת נתונים על לקוחות בענן פרטי אבל מספקת שירותים בענן ציבורי.

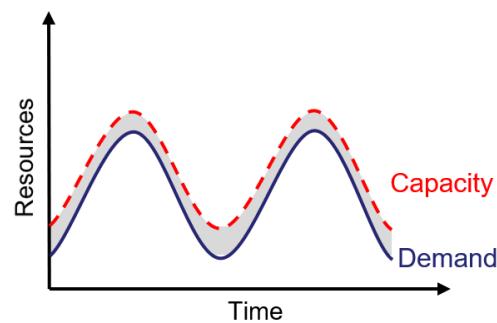
יתרונות הענן:

- חסכון עלויות – הגדלת ניצול המשאבים של הארגון.
- תחזוקה – מתבצעת ע"י הספקים ולא ע"י הלוקוחות.
- זמן – הגדלת התקופה של הארגון, המערכת זמינה תמיד.
- בטיחות ופרטיות – מסופקת ע"י הספקים.
- שיתוף פעולה – מקל על שיתוף מידע ומשאבים (כמו google docs).

בתמונה: עלות עבור לקוחות של שימוש בענן (ימין) לעומת פרטית של משאבי (שמאל). התמונה האפורה זה משאבי לא מנוצלים. כמובן זה רלוונטי לרוב המקרים, אבל יש חברות וגופים שיעדיפו להחזיק משאבי משל עצם מגוון סיבות.

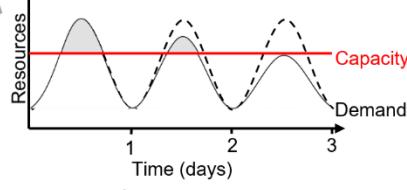
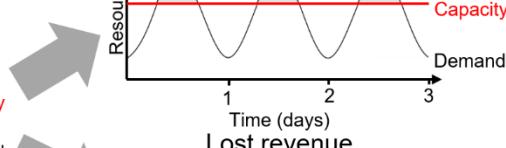
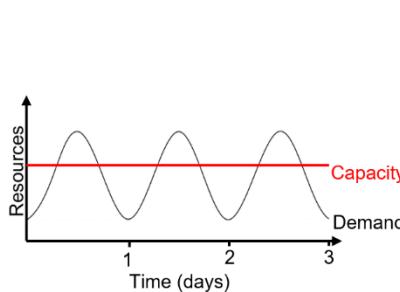


Static data center



Data center in the cloud

גרפים המתארים אובדן הכנסה ולקחות ללא שימוש בענן או שינוי משאבי.



לפי מידע מ-2008, לספק ענן משתמשים גדול. העלות שלהם יורדות ככל שהם גדלים ולקן הרווחים עולים.

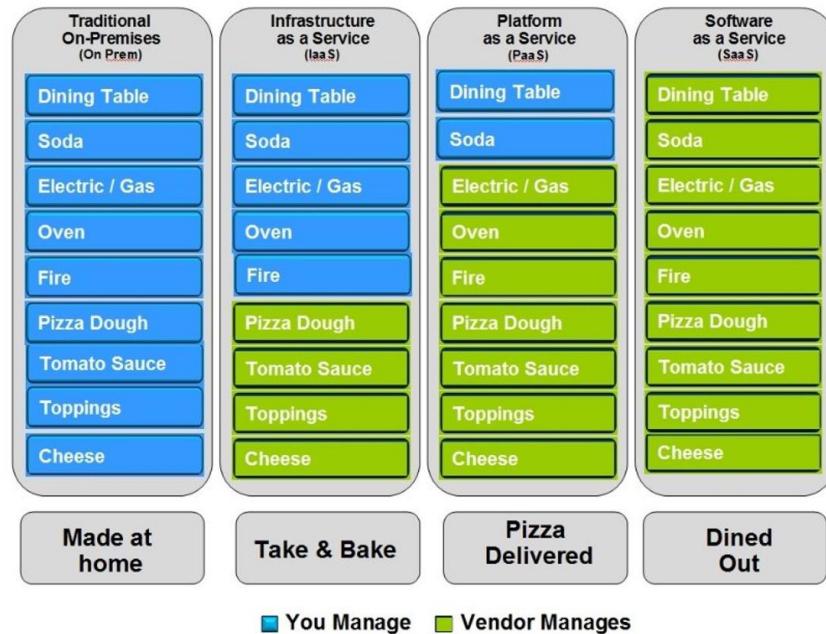
Resource	Cost in Medium DC	Cost in Very Large DC	Ratio
Network	\$95 / Mbps / month	\$13 / Mbps / month	7.1x
Storage	\$2.20 / GB / month	\$0.40 / GB / month	5.7x
Administration	≈140 servers/admin	>1000 servers/admin	7.1x

[Hamilton 2008]

באופן מסורתי, שירותי הענן מתחלקים ל-3 קטגוריות:
 - תשתיות כשירות (IaaS) Infrastructure as a Service
 - פלטפורמה כשירות (PaaS) Platform as a Service
 - תוכנה כשירות (SaaS) Software as a Service

בהתואיה לפיצה:

Pizza as a Service



בסיומו של דבר זה מסתכם איזה דברים ובאיזה רמה אתה רוצה לבנות בעצמך.

SaaS – תשתיית כשירות

בעצם הדגם הבסיסי ביותר של שירות ענן. כמו סטארטאפ שרצה להקים בעצמו מערכת מ-0 אבל לא רוצה לרכוש מחשבים, להקים צוות לתמיכה טכנית, להקים צוות סייר להגנה וכו'. סטארטאפ זה ישתמש בענן כתשתיית בסיסית שתספק לו לדוגמה: מחשבים לאחסון מידע, כתובות IP, חומרה אש ועוד מהוועה שירותים כמו: פתרונות ייעלים למידע (D), פתרונות לחברות שבמצוקת משאבים או מחפשות להתרחב, יכולת להקים רשות פנימית, שירותים וירטואליים להקמת אתרים ועוד.

יתרונות:

1. זמינות מיידית. (המידע זמין בענן בכל זמן שהליך צריך).
2. אין הגבלת מקום או מקום לא מנוצל. (הלקוח משתמש בבדיקה בקבילות הרציה).
3. חסוך בהוצאות. (הלקוח משלם רק על השימוש).
4. גיבוי המידע. (המידע מגובה בכמה מקומות).

SaaS – פלטפורמה כשירות

מציעה סביבת פיתוח ומערכות שונות. מספקת משאבי ותשתיות בשבייל הפיתוח. לדוגמה: Microsoft Azure. ספק השירות מציעים פלטפורמה נוחה לפיתוח מערכות שונות, כאשר התשתיית כולה מנולת על ידי הספק ומתעדכנת (כדי לעמוד בתחרות, ומספקת ללקוח כלים חדשניים שיעזרו לו להציג את המטרה שלו). לפעם גםנותנת הדרישה ורעיון לפיתוח ומספקת שירות בדיקה והרצה.

יתרונות:

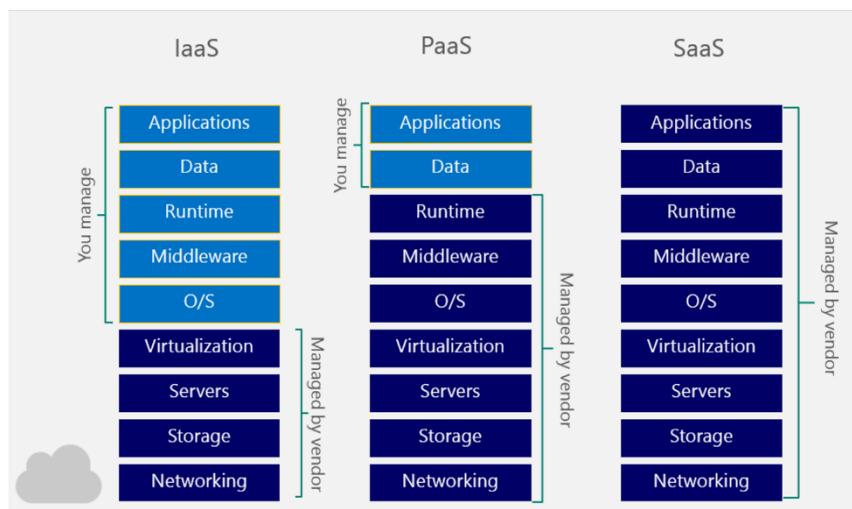
1. אין צורך להשקיע בתשתיות פיסית – שכן התשתיית כבר קיימת.
2. אין צורך להיות מומחה לפיתוח – שירותי הענן יעזרו לצוטרים להתעסק בהקמה עצמה כמו שיותר.
3. המפתחים יוצרים לעצם את סביבה העבודה שלהם – ע"י התקנת שירותי מותאמת אישית.
4. עובודת צוות פשוטה – כיוון שהפלטפורמה עצמה בענן, אפשר לעבוד ביחד ולהתחבר במספר מקומות.

SaaS – תוכנה כשירות

מציע תשתיית להתקנה והרצת אפליקציות דרך האינטרנט, כאשר האפליקציות עצמן מותקנות על הענן. למשל Google docs, facebook ועוד.

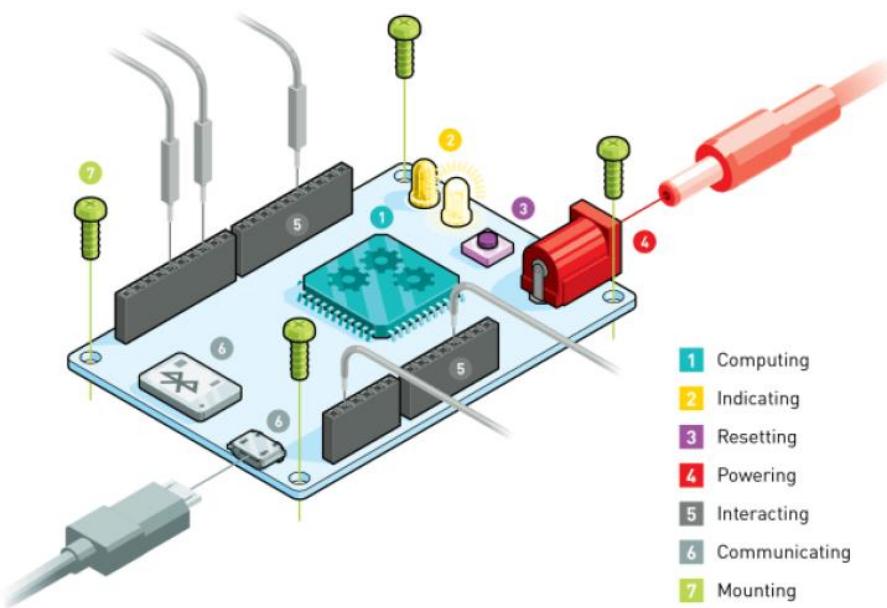
יתרונות:

1. אין צורך להתקין את התוכנה ולשמור את מאגרי המידע שלה במחשב הפרטי.
2. התחזקה והטמינה הטכנית מנווהל ע"י הספק.
3. ניתן לגשת לתוכנה דרך כמעט כל מכשיר המחבר לאיינטראנט (לדוגמה google sheets, google docs, שבאופן אישי אני אוהב אותו המונח גם דרך המחשב וגם דרך הסמארטפון, על אותן קבצים).



תרגול 1 Arduino Intro – 1

מאפיינים נפוצים של לוחות פיתוח:



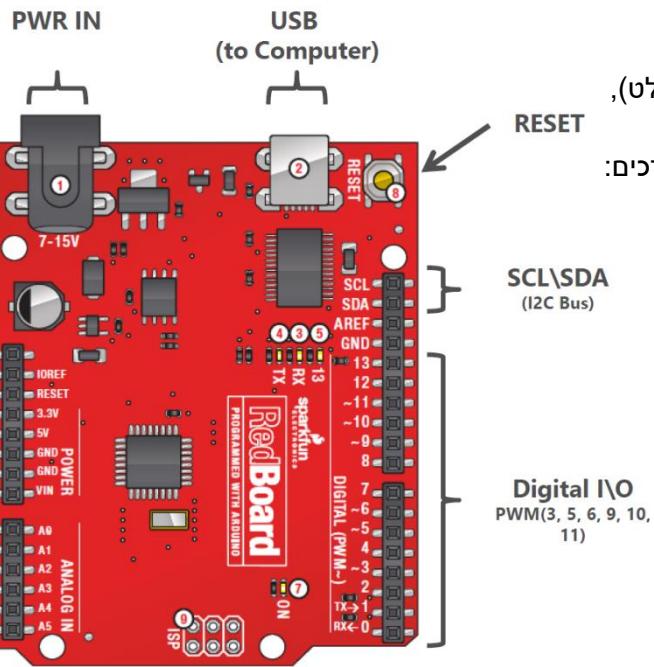
- 1 Computing
- 2 Indicating
- 3 Resetting
- 4 Powering
- 5 Interacting
- 6 Communicating
- 7 Mounting

1. יכולת חישוב – המעבד של הרכיב, זה שמבצע את הלוגיקה
2. אינדיקציה – LED שמהווים אינדיקטורים למצבים מסוימים. לרוב הלוחות, לכל הפחות יש LED שמראה אם הם מודלקים או לא.
3. ריסוט – Reset, אתחול
4. ספק כוח
5. אינטראקציה – הסבר בהמשך wifi, bluetooth, etc...
6. תקשורת – תקשורת פיזית להרכבה

אינטרקציה – לרוב הלוחות יש לפחות קלט ופלט בסיסיים. כמעט כל הלוחות יכולים להתמודד עם קלט דיגיטלי בסיסי, ויש גם לווח שמסוגלים להתמודד עם קלט אנלוגי.
לוחות מתקדמים יכולים להכיל גם אפליקציות שלמות, חיבור HD, חיבור SME, ועוד אחסון ועוד.

Scale of Microcontrollers/processors						
	CONTROLLERS			PROCESSORS		
Processor family	tinyAVR	megaAVR	ARM Cortex-M (size/low-power)	ARM Cortex-R (RT response)	ARM Cortex-A (performance)	Intel Atom
Architecture	8-bit RISC	8-bit RISC	32-bit RISC	32-bit RISC, Multicore	32-bit, 64-bit RISC, Multicore	32-bit x86, Multicore
Clock	4-20 MHz	Up to 20MHz	Up to 200MHz Often underclocked	Up to 1GHz	Can exceed 1.4GHz	500MHz
RAM	32-1024 Bytes	Up to 8KB	Up to 192KB	Implementation dependent	Implementation dependent	External
Storage	0.5-16KiB	4-256KB	Up to 4MB	Implementation dependent	Implementation dependent	External
Development Boards	Digispark	Arduino & clones	ESP-8266, NodeMCU/WeMos WiFiMCU	C.H.I.P	Raspberry Pi, Dragonboard, many others	Intel Edison
More info	http://www.atmel.com/products/microcontrollers/avr/tinyavr.aspx	http://www.atmel.com/products/microcontrollers/avr/megaavr.aspx		http://www.arm.com/products/processors		http://www.intel.com/content/www/us/en/processors/atom/atom-processor.html

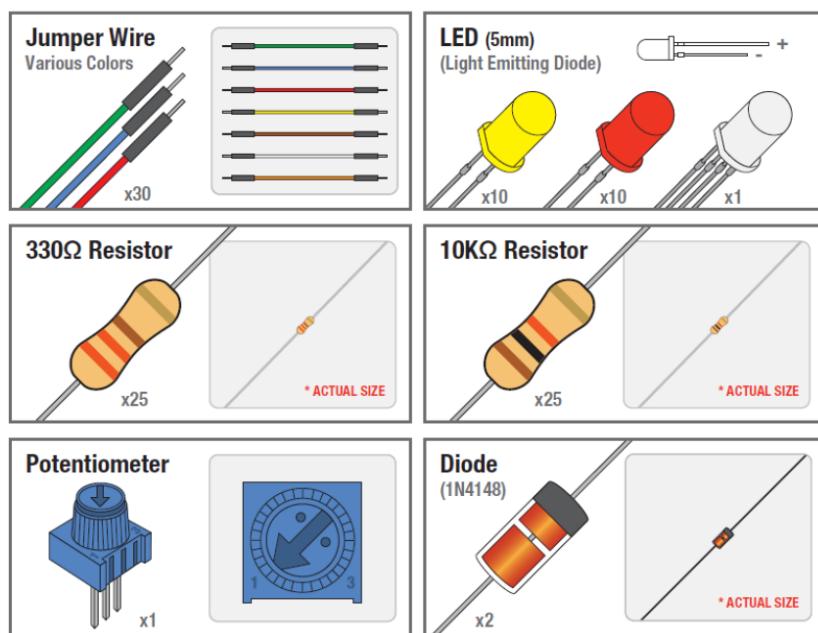
ארדואינו
נקרא על שם של פאב שקרי על שם של מלך איטליה.



[https://www.sparkfun.com/products/10000](#)

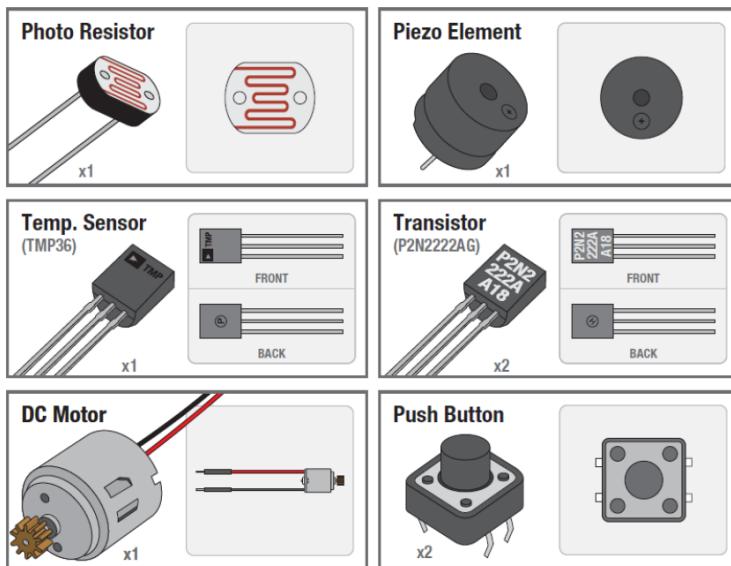
כמו שניתן לראות, יש פה כפטור אתחול באסים להעברת מידע, 0-13 זה קלט פלט דיגיטלי, A0-A5 זה קלט אנלוגי בלבד (לא פלט), ומוגון יציאות מתחה (לפי V וגם האורך). אפשר לחבר את הלווח למקור אנרגיה ב-3 דרכים:

1. ע"י IN PWR
2. ע"י USB
3. ע"י חיבור למתח V_{in} ובתנאי שנאריך מיציאת GND.



חומרים נפוצים

חוטים – להעביר זרם, עדיף לשמר על צבעים מסודרים כדי לדעת מי נכנס מי יוצא ולאן LED – נגד צבעו
נגדים – מוריד את הזרם החשמלי.
פוטנציאומטר – נגד משתנה
דיודה – מאפשר העברת זרם חשמלי בכיוון אחד

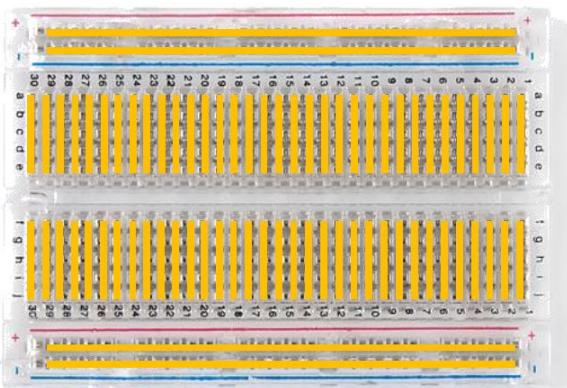


פוטודיסטור – מוליך למחצה שכתוכזה מגיעה זרם דרך התא.

פיאזו – (לא יודע) חישון טמפרטורה.

טרנזיסטור – מוליך למחצה שמשמה מנגנון אלקטרוני. שלוטים בזרם האלקטרוניים דרך התחkon. מנוע זרם עקיף.

כפטור לחיצה – מעביר זרם רק בעת סגירת מעגל, שנעשה באופן ידני ע"י לחיצה.



מטריצה:

כל שורה (5 חורים) מחוברים ומעבירים מתח אחד בין השניהם.
כל שורה שכזאת מחוברת לבאוס מתח שהוא טו.

קצת הערות לגבי הקוד:

`pinMode`, מקבלת מספר חיבור דיגיטלי ומצב (Input/Output). אתחול של החיבור.
`digitalWrite`, מקבלת מספר חיבור, ואת האות (HIGH/LOW) שציריך לכתוב אליו.

`digitalRead`, מקבלת רק מספר חיבור ומחזירה את הערך הנקרה ממנו. זה דילוי, `delay`

-
-
-
-
-

שמייצג מתח בין 0 ל 5 volts

`analogWrite`, מקבלת כניסה וערך בין 0 ל 255, כאשר 255 זה מתח של 100%, 0 זה מתח של 0% ו 127 זה מתח של 50%.

נשים לב, שמספק החום מוגבל ל 3,5,6,9,10,11, עם הסימן (~). Serial – מגננון שמאפשר תקשורת בין הלווח למכשיר אחר.

-
-

(x), Begin את הקצב x בו יעברו ביטים. בשבייל

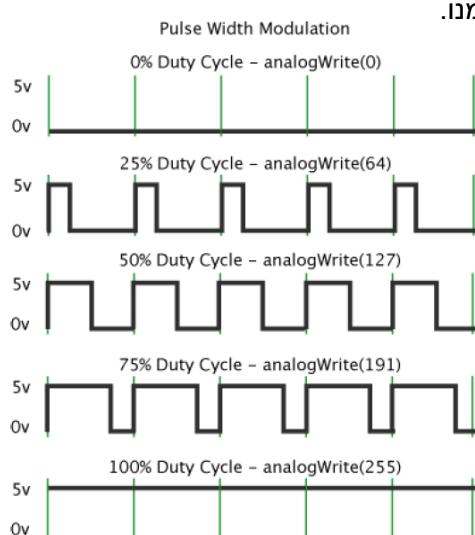
תקשרות עם מחשב משתמשים ב 300,600,1200,28800,38400

2400,4800,9600,14400,19200,57600,115200

Print/println, מקבל ערך ופורמט ומדפסה אותו.

-

פורמט יכול להיות HEX,DEC,OCT,BIN.

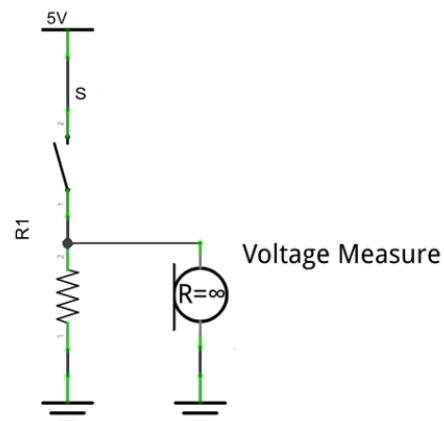
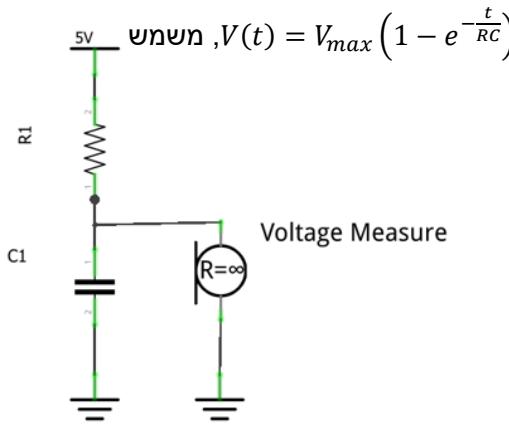


קצת חשמל בתיאוריה:

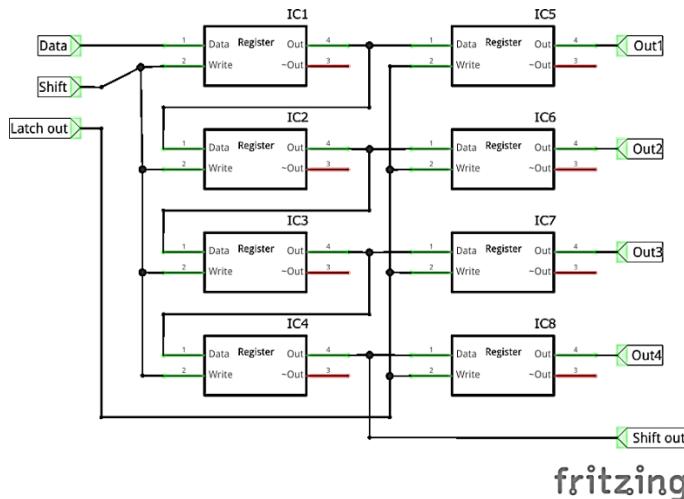
- חוק א Ohm, $V=IR$
- חיבור נגדים בטור, $R_{total} = R_1 + R_2 + \dots$
- חיבור נגדים במקביל, $\frac{1}{R_{total}} = \frac{1}{R_1} + \frac{1}{R_2} + \dots$
- חיבור קבילים בטור, $\frac{1}{C_{total}} = \frac{1}{C_1} + \frac{1}{C_2} + \dots$
- חיבור קבילים במקביל, $C_{total} = C_1 + C_2 + \dots$
- הזרם קבוע במעגל, אבל מתח על כל נגד או חוט יכול להיות שונה
- בחיבור מקביל, הזרם יתפצל אבל המתח יהיה זהה

מעגליים שימושיים
מפצל מפתח

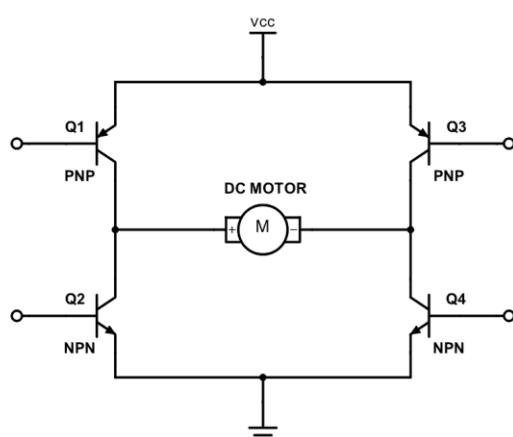
smoothed voltage (טעינה של קובלן נגד), משמש ל



מעביר מידעシリלי למקביל – Shift register



גשר H – מאפשר הנעה של גלגל אחורה
או קידימה



מחלק מתח פשוט מורכב ממשי נגדים בחיבור טורי בין מקורות מתח בין הארקה. הה Tangendot השוקה של שני הנגדים היא R_1+R_2 . על פי חוק א Ohm, הזרם במעגל הוא:

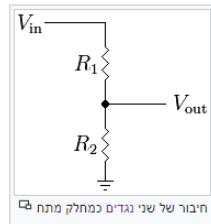
$$I = \frac{V_{in}}{R_1 + R_2}$$

לכן המתח על הנגד R_2 , שהוא מתח המוצא, יהיה:

$$V_{out} = V_2 = I_2 \cdot R_2 = \left(\frac{V_{in}}{R_1 + R_2}\right) \cdot R_2 = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$

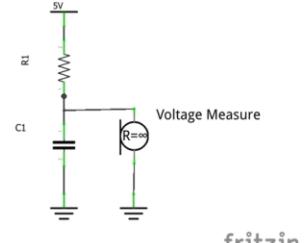
לפ"כ, תוצאה של מחלק המתח היא $\frac{R_2}{R_1 + R_2}$ מהותה במובן

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in} = \frac{R_1}{R_1 + R_1} \cdot V_{in} = \frac{R_1}{2 \cdot R_1} \cdot V_{in} = \frac{1}{2} \cdot V_{in}$$



Low-Pass Filter

- Charging a capacitor through a resistor
- RC Law
- RC Constant depends on resistor and capacitor
- $V(t) = V_{max} \cdot (1 - e^{-\frac{t}{R \cdot C}})$
- Greater resistance and/or capacitance means slower charge



RC ביחידות של 10^{-3}

Color	Digit	Multiplier	Tolerance (%)
Black	0	10^0 (1)	
Brown	1	10^1	1
Red	2	10^2	2
Orange	3	10^3	
Yellow	4	10^4	
Green	5	10^5	0.5
Blue	6	10^6	0.25
Violet	7	10^7	0.1
Grey	8	10^8	
White	9	10^9	
Gold		10^{-1}	5
Silver		10^{-2}	10
(none)			20

איך נחשב? לפי הtablלה. קודם הספרה כי שמאלית. זה אדום, אז 2. אחריו זה שחור. אז זה 20. נכפיל במספר האפסים של הצבע השלישי, כתום זה 10^3 . הצבע האחרון זה אוחז. למשל כאן, $20 * 10^3 Ohms$ עם 1% אפיקות.

אנדרואיד

באופן כללי, המושג Activity מייצג מסך. לחיצה על כפתור או פעולה מסוימת יכולה להעביר אותו למסך אחר, משמע לפעולות אחרות.

סוגי קבצים חשובים:

1. AndroidManifest.xml – מכיל מידע חיוני על האפליקציה עבור הבילד של אנדרואיד, מערכת הפעלה של אנדרואיד והעילות Google Play.
2. MainActivity.java – מכיל את כל הקוד שמנהל את האפליקציה. זה בעצם המסך הראשי שלנו וממנו מתחילה. זה קובץ java ולכן מהוות בעצם את ה backend.
3. Activity_main.xml – קובץ שמנגדיר את ה UI. מכיל טקסט, כפתורים, מפות, תמונות ועוד. ניתן לערוך אותו הן מבחינת קוד XML והן עם המשחק של Android Studio.
4. Values – מחזיק מידע סטטי לגבי האפליקציה, כמו צבעים, מחרוזות קבועות וכו'.
5. Build.gradle – מגדר את הבילד עבור מודל מסוים ועבור כל הפרויקט.

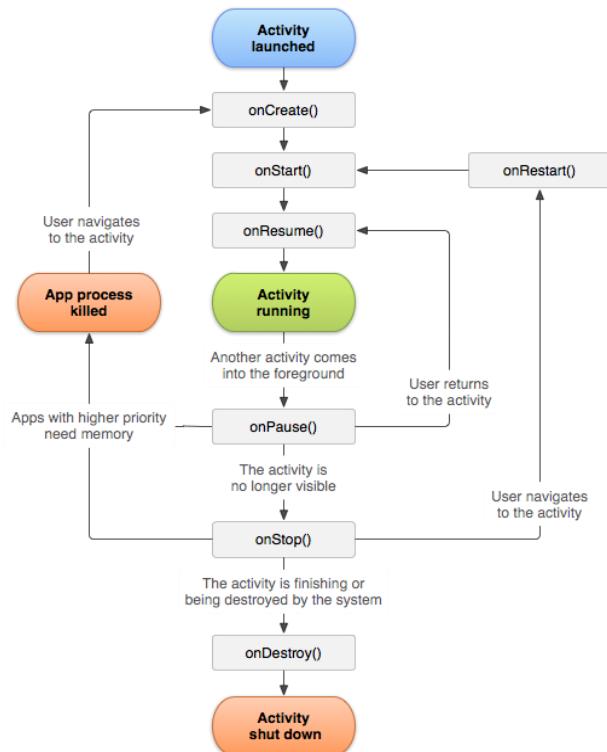
איך נראה קובץ xml ? פותחים עם <> אובייקט וסוגרים אותו עם </>. לפ' scoping, ניתן לגשת למשתנים פנימיים ולערוך אותם.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.marwantemp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MarwanTemp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```



הFLOW הבסיסי:
הבדל בין create ל start:
הבדל בין ליצור מסך, לבין להתחיל את הפעולה שלך.

Serverless computing

הרעיון, שבמוקם לשכור שרת, אפשר לשכור זמןuibוד על הענן. משמעו, כתובים קוד, מעלים אותו לענן, והתשלומים מתבצע לפ' כמות הלקוחות לקוד. הקוד לא בהכרח ירוץ תמיד על אותו סוכן או שרת. זה אפקטיבי למשימות שלא צרכות לשמור state ולהסתמך עליו.

היתרון שזה מעניק למפתח, הוא התעסקות מקסימלית בקוד האפליקציה וה מוצר עצמו. זמינות גבוהה, אמינות ואין תשלום על מהهو שלא מסופק. משמעו, אם התוכנית שלך צריכה 10gb, אבל השירות המינימלי שתוכל לשכור הוא 16gb תצטרך לשלם על 9gb שלא בשימוש. בשיטה הזאת, אתהשלם רק על 10gb שכן אתהשלם לפ' הפעלה ולא לפ' אחסון.

פונקציות ענן

מהוות הפעלה שהיא serverless שנועדה ליצור ולחבר שירות ענן. ב-[Firebase](#), מטאפר לשורץ קוד backend בתורה לאירועים בטור firebase ובקשוט HTTPS.

באופן יותר פשוט, כתובים פונקציות פשוטות עם מטרה אחת שמוצאות לאירועים. אירועים אלה מקבלים טרייגר מתחנית הענן ושירותים אחרים. הקוד מאוחנן ורץ בענן של גול מבוקר, כך שאין צורך להתעסק בה.

הודעת HTTP

דרך ייעלה של תקשורת בין המשתמש לשרת. נשלחת ע"י המשתמש אל השירות.

לדוגמא, אפליקציה שעושה הרבה דברים. לא צריך לשים את כל הקוד על האפליקציה. זה יכבד עליה, זה יקשה על עדכנים, ויתכן שהאפליקציה תיפרץ וחלקי קוד חשובים יגנבו.

במקרה זה, מישות פשוטות שלא באמצעות מיצירות את הימצאותן בקוד המקור של האפליקציה יכולות לעבור לשרת firebase. זה יפתר את כל הבעיה שרשמנו קודם.

עם firebase ופונקציות ענן אפשר בנוסף לחבר שירותים ופיצרים מובנים של גול, כדי לחזק את יכולת הטכנית יכולות האבטחה של האפליקציה, תוך כדי התעסקות מקסימלית באפליקציה עצמה.

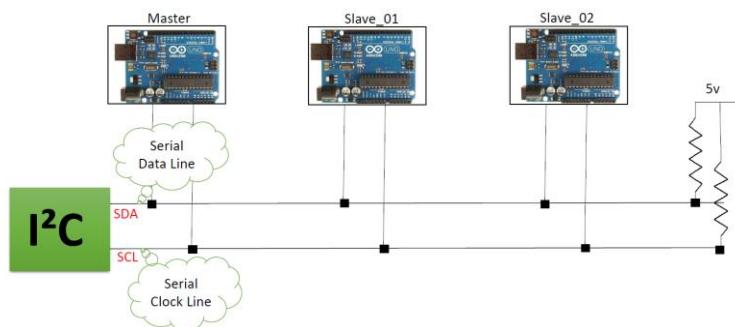
טכנולוגיות תקשורת ופרוטוקולים

WiFi, Bluetooth, NFC

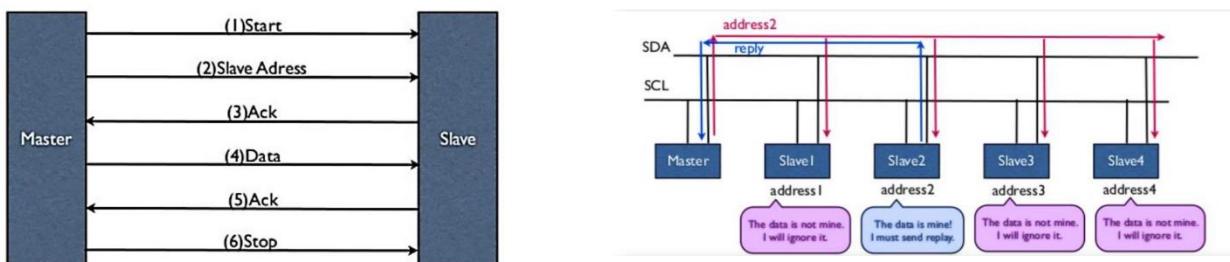
פרוטוקולים סדרתיים נפוצים: I2C, SPI, USB

I2C

פרוטוקול סינכרוני שמאפשר למכשיר מאסטר ליזום תקשורת עם מכשיר שכפוף לו. מידע מועברים בין שני המכנים. המכשיר מאסטר שלט בשעון (SCL), וכל הkopופים נשלטים ע"י שעון המאסטר. מכשיר כפוף מקבל את השעון והוא מאשר המאסטר פונה אליו. Serial Data ,SDA שנקרא בReLU זיהה את המאסטר.



על מנת ליזום העברת מידע, המאסטר שלוחה בReLU אדדס SDA את ההודעה address2, הכוונה ל2 slaves. כל הkopופים קיבלו את ההודעה הזאת, אבל יזהו שההודעה לא מיועדת אליהם ולכך ידעו להתעלם מההודעה הבאה, חוץ מ-2 slaves. שיזהה שההודעה בשבלו, ויגיב בחזרה. לאחר מכן, המאסטר ישלח את המידע שוב עלReLU SDA, ורק 2 slaves יקבלו את המידע ויחזירו תגובה.



אם נרצה להעביר מידע של יותר מ-1byte, נצטרח לחזור על שלבים 4 ו-5.

