**The Beacon OpenFlow Controller**

**Beacon** – מבוסס JAVA, נוצר ב-2010, מתפקד כמנהל OpenFlow בגישת open source.

נעשה בו שימוש רחב למטרת לימוד, מחקר ובסיס למוצרי תאורה Floodlight.

**המטרות של Beacon:**

1. **שיפור היצרנות (הפריון) של המפתח.**
2. **לספק את היכולת לפתוח אפליקציות חדשות ולעצור אפליקציות רצות, בזמן ריצה.**
3. **להיות בעל ביצועים טובים.**

**מנהל התקשורת הראשון בגישת open source שקדם ל-Beacon הוא NOX.  
NOX מאפשרת פיתוח בשפת Python שהיא נוחה וידידותית יותר עבור פיתוח אפליקציות רשת או בשפת C++ שהיא בעלת ביצועים טובים יותר.**

**NOX מציבה עבור המפתחים בה את ההפרדה בין נוחות לאיכות.**

**שאלות שנתעסק בהן:**

1. במנהל ה- OpenFlow NOX ישנה הפרדה בין כתיבת אפליקציות רשת בשפה ידידותית, נוחה וקלה כמו Python לבין כתיבה בשפה יעילה יותר מבחינת ביצועים (C, C++) – כלומר אפליקציות באיכות גבוהה.

**האם מנהל OpenFlow יכול לקבל את שניהם?**

**להיכתב בשפה ידידותית, נוחה וקלה מחד ולהיות בעל ביצועים טוב מאידך?**

1. **אם מנהל OpenFlow דומה במהותו למערכת הפעלת תקשורת (Network Operation System) האם הוא צריך להיות בעל יכולת לפתיחת וסגירת אפליקציות בזמן ריצה?**

שפת הפיתוח וסביבת הפיתוח חשובות מאוד ומהוות **חלק ניכר** מרמת היצרנות של המפתח ובאיכות המוצר שייצר.

ישנן המון שפות פיתוח ידידותיות אך טיב הביצוע שלהן בכל מה קשור לניהול OpenFlow לא היה ידוע בתקופת NOX.

**NOX – קצת רקע**

פלטפורמת מנהל ה-OpenFlow הראשון של NOX יצא ב-2008 עבד בשיטת ביצוע שיתופי בין תהליכונים.

**חסרונות של C ו- C++**

* זמן קומפילציה **מלאה** ארוך מאוד (יותר מ-10 דקות)
* שגיאות קומפילציה לא מובהקות, לא קל להבין מה יוצר את הבעיה.
* הקצאות ושחרור ידני של זכרון, יוצר קריסות עקב עומס על זכרון שמנוהל לא טוב או דליפות זכרון (זכרון שלא שוחרר כראוי ותופס ללא שימוש).

**שפות בעלות ניהול זכרון אוטומטי (Garbage Collector)**

יכולת ניהול הזכרון האוטומטי תורם לעבודה עם שגיאות זכרון מעטות. שפות שמנהלות את הזכרון שלהן באפן אוטמטי לרוב גם בעלות קומפילציה מהירה יותר ויכולת הכוונה מדויקת לטעויות עד רמת השורה שבה נעשתה הטעות.  
מאוד ידידותי למפתח.

בחירת השפות התמקדה ב-3 פרמטרים: ניהול זכרון אוטומטי (תרומה אדירה לידידותיות), גמישות שימוש במערכת הפעלה, דרגת ביצוע גבוהה.  
יודע על קיומן של המון שפות מתאימות שהמנהלות זכרון אוטמטי אבל המוכרות ביותר לדיוויד אריקסון (כותב המאמר) היו Python, C#, ו-JAVA.

**התמקדות בשפה אחת:**

* גמישות שימוש במערכת הפעלה היא אלמנט חשוב בעת בחירת שפה שצריכה לשרת Controller, רצוי שירוץ על 3 מערכות ההפעלה הגדולות Linux, Windows, Mac OSX.  
  3 השפות הנ"ל אכן עושות זאת אך C# מתפקדת לא טוב כשזה מגיע למערכות הפעלה שהן לא Windows שם היא מתפקדת ללא דופי, לכן C# נפסלה.
* איכות ביצוע זה עניין של מקרה ספציפי, בהתאם למה שמתעסקים בו. בכל מה שקשור לפיתוח מנהלי תקשורת ב-OpenFlow מאוד חשוב לשים דגש על ביצוע עיבוד, על תהליכונים לרוץ בצורה יעילה ככל האפשר כדי לתרום להליך יעיל והריצה היעילה ביותר היא ב-Multithreading.  
  מראשית המחקר טיב איכות הביצוע של JAVA ו-C# בכל מה שקשור לפיתוח ב-OpenFlow לא היה ידוע או ודאי אך בכל מה שנוגע ל-Python כבר מתקופת השימוש ב-Controller של NOX ידוע כי טיב הביצוע שלו נמוך לעומת שפות אחרות כמו C ו-C++ שגם בהן נעשה שימוש ב-NOX.  
  הסיבה שאיכות הביצוע שלו ירודה היא בגלל שהתפקוד שלו בכל מה שקשור ל-Multithreading לוקה בחסר ולכן Python נפסלה.
* JAVA מתפקדת בשלושת מערכות ההפעלה המדוברות באופן חלק ולכן גמישה מבחינת מערכת ההפעלה.  
  בעלת ניהול זכרון אוטומטי וידידותית למשתמש.  
  עובדת בשיטת Multithreading ולכן צפויה לתת ביצועים טובים בכל מה שקשור לפיתוח OpenFlow.  
  JAVA מבין השלושה נבחרה להיות השפה המרכזית במאמר עלייה בוצעו הניסויים, התוצאות היו מרשימות יותר ממה שצופה.