

# פרויקט בקר רשת

שם: דן מדר

ת.ז.: 207270851

שם המנחה: גולן מור

תאריך: 08/06/2017

תוכן עניינים

3	תקציר ורציונל הפרויקט
4	מבוא ורקע כללי
5	מטרת הפרויקט
6	שפת התכנות וסביבת העבודה
7	ניסוח וניתוח הבעיה האלגוריתמית
9	תיאור אלגוריתמים קיימים
10	הפתרון הנבחר
13	פיתוח הפתרון בשכלול הקוד עם שפת התכנות
41	השוואת העבודה עם פתרונות ויישומים קיימים
42	הערכת הפתרון לעומת התכנון והמלצות לשיפור
43	תיאור של הממשק למשתמש – הוראות הפעלה
45	מבט אישי על העבודה ותהליך הפיתוח
45	ביבליוגרפיה

### **תקציר ורציונל הפרויקט**

הפרויקט מזהה את כל המחשבים ברשת הפנימית ומשיג מידע על כל אחד מהם. לאחר מכן הוא מציג אותם בצורת עץ כאשר המחשב שלי הוא השורש של העץ, הבנים שלו הם הראוטר, מחשבים שמשתמשים בפרוטוקול SNMP, ומחשבים שלא משתמשים

בפרוטוקול הזה.  
כמו כן, הפרויקט יזהה מחשבים שנכבו (או שהתנתקו מהאינטרנט)  
וכך אפשר לזהות תקלות באינטרנט, ואפילו אפשר להדליק מחשב  
שנכבה.

- בחרתי לעשות את הפרויקט הזה מכמה סיבות:
- אני מתעניין מאוד בתקשורת, אז רציתי לעשות פרויקט שעוסק בעיקר בנושא הזה.
  - רציתי לחקור את העולם של "ניהול רשת". רציתי להבין איך רשת פנימית עובדת ולדעת עליה דברים חדשים.

### **מבוא ורקע כללי**

פעם, מערכות מקושרות היו מוחזקות בדרך כלל בבניין או אפילו בחדר בודד. אבל כיום, ארגוני תקשורת מקיפים ערים, מדינות ואת העולם כולו. המסובכות הזאת גרמה לניהול התקשורת להיות משימה קשה.  
תאור מדויק של הרשת הוא תנאי בסיסי בשביל לנהל מערכת,

לפתור תקלות ולאפשר גדילה בהצלחה(למשל, כמות השרתים שמשתמשים בהם).  
בעזרת כלים לניהול רשת, התהליך הזה נהיה יותר פשוט.

### **מטרת הפרויקט**

#### **מה המוצר המוגמר אמור לבצע:**

לזהות את המחשבים (שדלוקים) ברשת הפנימית ולהציג אותם בצורת עץ.  
להראות מידע שונה עליהם, ולהראות אם אחד מהמחשבים נכבה (או מתנתק מהאינטרנט).

### **דרישות מרכזיות:**

- לזהות את המחשבים במהירות הגבוהה ביותר שאפשר.
- להציג בצורה ברורה ונוחה את המחשבים.

### **תרחישים במערכת:**

- העץ יוצג ויהיה ניתן לראות את המחשב שמריץ את התוכנה ואת הראוטר.
- למשתמש תהיה אופציה האם לסרוק את המחשבים ברשת הפנימית.
- לאחר שהוא יגיד שכן, המחשבים שיזוהו יתווספו לעץ.
- נכבה את אחד המחשבים שמוצג בעץ, והצומת המציג אותו בעץ יהיה בצבע אדום.
- המשתמש יוכל להדליק את המחשב דרך הממשק משתמש גרפי.
- לאחר שהמחשב יודלק, הצומת המציג אותו בעץ יהיה בצבע ירוק.

### **שפת התכנות וסביבת העבודה**

בכתיבת הפרויקט עשיתי שימוש בשפות התכנות Python ו-C#. השתמשתי ב- python2.7 וכתבתי סקריפטים של Python ב- notepad++ והשתמשתי ב-windows forms של visual studio מאת Microsoft. מערכת ההפעלה שעליה עבדתי היא windows. השתמשתי ב-Python בעיקר בשביל שימוש בשליחת פקטות למען זיהוי מחשבים ועוד, וב-C# השתמשתי בשביל להציג את המידע שהשגתי מהסקריפטים שכתבתי-Python.

בשביל לקשר בין שפות התכנות הנ"ל, השתמשתי ב-json file שהוא קובץ שאפשר לשים בו מידע, ושפות תכנות שונות יודעות לקרוא אותו (Python ו-C#). כמו כן, השתמשתי ב-C# כדי להריץ סקריפטים של Python. בחרתי בשפות אלו כי הן שפות שאני מכיר ומתמצא בהן, וגם C# היא שפה נוחה לכתיבת GUI

### **ניסוח וניתוח הבעיה האלגוריתמית**

הבעיה אותה הפרויקט שלי מנסה לפתור היא כיצד ניתן לזהות את המחשבים ברשת הפנימית ולהציג אותם ב-GUI במהירות המרבית ביותר. ישנן שתי בעיות עיקריות:

- יש לדעת שכדי לזהות מחשבים ברשת הפנימית שלך, צריך לשלוח Ping request לכל כתובות ה-IP שעלולות להיות ברשת הפנימית. בדרך כלל, בבתי

מגורים כתובות ה-IP שיכולות להיות הם כ-254 כתובות IP, וברשתות גדולות יותר כמו בית ספר, יש הרבה יותר כתובות IP שיכולות להיות במחשבים. כלומר, צריך לדעת לשלוח המון ping request לכל כתובות ה-IP, ומתי שמקבלים תשובה מ-IP מסוים, צריך לשמור אותו ולהשיג עליו עוד פרטים (למשל mac address).

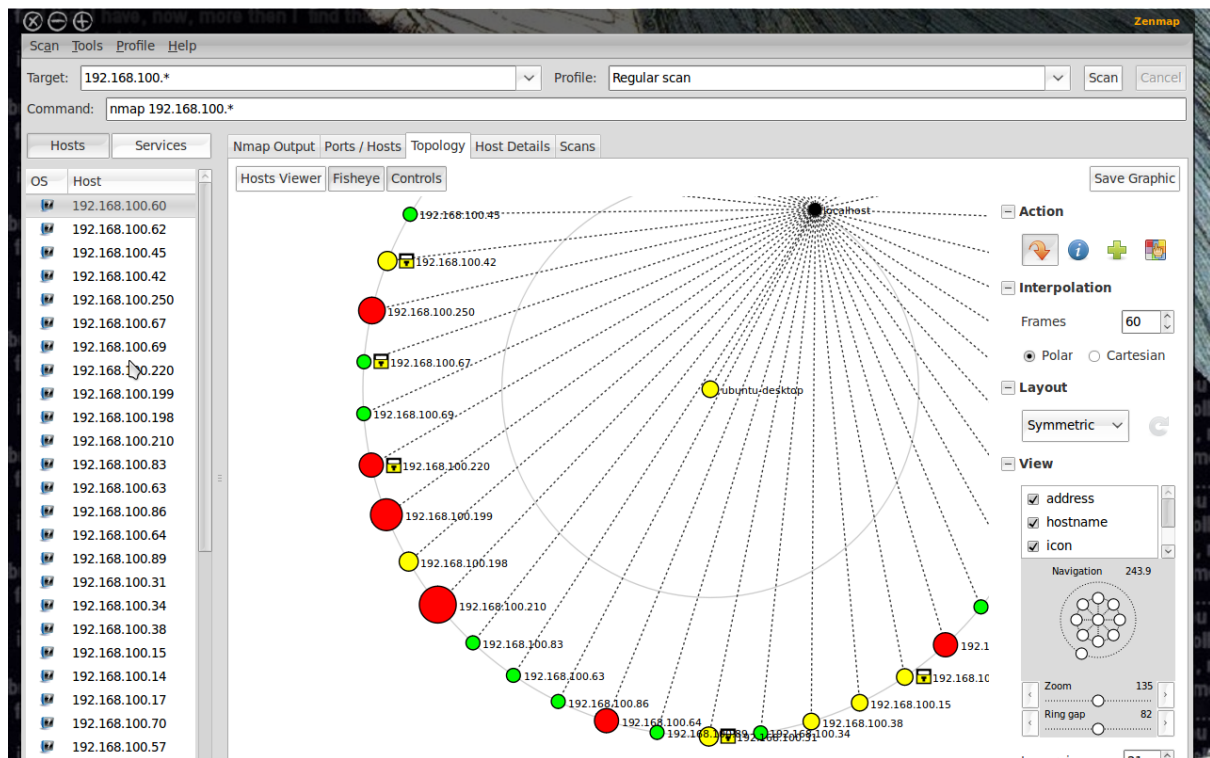
צריך לבצע זאת ביעילות הגבוהה ביותר ובמהירות המרבית ביותר.

- צריך להציג את המידע שהושג בצורה ברורה, נוחה ואינטראקטיבית שתיתן למשתמש לבצע פעולות שונות עם העכבר. צריך לדעת באיזו צורה להציג את המידע (למשל, גרף או עץ), ולדעת איך להשיג את המידע בצורה יעילה.

### תיאור אלגוריתמים קיימים

ישנם אלגוריתמים קיימים שפותרים בעיה זאת.

- Nmap (ממפה רשת - Network Mapper) - זוהי תוכנית סריקת רשת, המשמשת לגילוי מתחמים ושירותים, ברשת מחשבים. פעולת Nmap היא שליחת חבילות בעלות מבנה מסוים אל המתחם הרצוי, וניתוח התגובה המתקבלת ממנו. יש לציין ש-Nmap סורק את הרשת בצורה מהירה ביותר ויכול גם להציג אותה בצורה גרפית (באמצעות zenmap - תוכנה שמציגה את המידע שהתוכנה Nmap משיגה, בצורה גרפית):



## הפתרון הנבחר

## האלגוריתם הנבחר

האלגוריתם שבחרתי הוא לסרוק את הרשת הפנימית ב-Python, וכדי לעשות זאת במהירות המרבית אני משמש במודול multiprocessing על מנת לשלוח ping request למחשבים שונים במקביל. כמו, כן אני מגדיר timeout קצר יותר לבקשות כדי שהסריקה תהיה מהירה יותר, וגם מגדיר שהמחשב שלי ישלח Ping request אחד לכל מחשב (ברירת המחדל של Ping request היא 4 בקשות).

כל מחשב שאני מזהה שהוא קיים, אני אבקש ממנו מידע שונה (למשל Mac address ו-Hostname), ולבסוף אני אשמור את כל המידע שהשגתי ב-Json file.

במקביל, אני אצור עץ ב-C# שיציג את כל המידע שיש ב-json file. הסקריפט ב-Python והתוכנית ב-C# ירוצו בנפרד, כך שכל פעם שהסקריפט ב-Python יזהה מחשב



חדש, הוא ישמור אותו ב-Json file, ואז התוכנית ב-C# תיקח את המידע מהקובץ ותציג אותו מיד בעץ. כמו כן, התוכנית ב-C# תריץ סקריפט ב-Python, שתעביר אליו את המחשבים בעץ, והסקריפט יזהה אילו מהמחשבים נכבו (או שהתנתקו מהאינטרנט), ישמור אותם ב-Json file. התוכנית ב-C# תזהה אילו מחשבים נכבו ותדגיש אותם בצבע אדום. כמו כן, אם לאחר מכן הסקריפט יזהה שהם נדלקו, אז צבעם ישתנה לצבע ירוק.

### **מבנה המערכת**

#### **C# - TreeDisplay Project**

הפרויקט ב-C# שבונה את ה-GUI של המחשבים ברשת הפנימית.

יש בו כמה קבצים:

#### **Form1.cs**

התוכנית המרכזית, היא מריצה את הסקריפטים ב-Python, משיגה את המידע מהם, ומציגה אותו בעץ. יש לה כמה פונקציות שונות. למשל, להציג רק כתובות IP של המחשבים.

יש כמה מחלקות בפרויקט:

#### **Machine.cs**

מחלקה המייצגת "מכשיר". יש לה כמה תכונות כמו כתובת IP וגם Hostname. כל צומת בעץ הוא מייצג "מכשיר"

#### **SNMP Machine.cs**

מחלקה היורשת את המחלקה Machine, ומייצגת מכשיר שמשתמש בפרוטוקול SNMP. בעזרת פרוטוקול זה, אפשר

לדעת פרטים שונים על המכשיר (למשל, מערכת ההפעלה של המכשיר).

### **AllMachines.cs**

מחלקה שיש בה רשימה של "מכשירים". יש לה כמה פונקציות, למשל, פונקציה המחזירה את המכשיר שמייצג את המחשב שלי.

ב-Python ישנם כמה סקריפטים:

### **cmdInteracting.py**

סקריפט חשוב שמשיג מידע על המחשב שלי, ועל הראוטר, דרך ה-CMD של Windows. נעשה בו שימוש בסקריפטים השונים שכתבתי ב-Python.

### **scan\_all.py**

סקריפט הסורק את כל הרשת הפנימית ושומר את המחשבים שזיהה ב-Json file, כדי שהתוכנית ב-C# תיקח את המידע שהושג.

### **getMyComputer\_router.py**

סקריפט הבונה עצמים של המחשב שלי ושל הראוטר ושומר אותם ב-Json file כדי שהתוכנית ב-C# תיקח את המידע ותציג אותם בעץ.

### **getShutdownComputers.py**

סקריפט המקבל את כתובות ב-IP של המחשבים בעץ, בודק איזה מהם כבוי (או מנותק מהאינטרנט) ואז שומר את המידע ב-Json file. התוכנית ב-C# מקבלת את המידע, ומדגישה את המחשב הכבוי בצבע אדום.

### **file\_wakeonlan.py**

סקריפט המקבל mac address מהתוכנית ב-C# ומדליק את המחשב מרחוק.

## **פרוטוקול תקשורת-SNMP (Simple Network Management Protocol)**

ישנם מחשבים שמשתמשים בפרוטוקול SNMP. ממחשבים אלו אני מבקש מידע מסוים (למשל, פרטים על מערכת ההפעלה של המחשב), ומציג אותו ב-GUI.

## **פיתוח הפתרון בשכלול הקוד עם שפת התכנות Machine.cs**

המשתנים של מחלקה "Machine" והבנאי שלה:

```

public class Machine
{
    private string ip; // ip address of machine
    private string mac; // mac address of machine
    private string hostname; // hostname of machine
    private bool isMyComputer; //true or false if the machine is my computer
    private bool isRouter; // true or false if the machine is the router

    public Machine(string ip, string mac, string hostname, bool isMyComputer, bool isRouter)
    {
        this.ip = ip;
        this.mac = mac;
        this.hostname = hostname;
        this.isMyComputer = isMyComputer;
        this.isRouter = isRouter;
    }
}

```

פעולות Get ו-Set:

```
// Get & Set

public string IP
{
    get { return ip; }
    set { ip = value; }
}

public string Mac
{
    get { return mac; }
    set { mac = value; }
}

public string Hostname
{
    get { return hostname; }
    set { hostname = value; }
}

public bool IsMyComputer
{
    get { return isMyComputer; }
    set { isMyComputer = value; }
}

public bool IsRouter
{
    get { return isRouter; }
    set { isRouter = value; }
}
```

---

## פעולה המחזירה את הטקסט שצריך להיות למכשיר בעץ:

```
public string GetNodeOutput(string status) //returns the output of the machine in the treeView
{
    if (status == "ip")
    {
        return ip;
    }
    else if (status == "mac")
    {
        return mac;
    }
    return hostname;
}
```

## פעולת ToString של המחלקה:

```
public override string ToString()
{
    if (isMyComputer) // if the machine is my computer
    {
        return string.Format("My Computer: ip: {0} , mac: {1} , hostname: {2}", ip, mac, hostname);
    }
    if (isRouter) // if the machine is the router
    {
        return string.Format("Router: ip: {0} , mac: {1} , hostname: {2}", ip, mac, hostname);
    }
    // the machine is not my computer or the router
    return string.Format("ip: {0} , mac: {1} , hostname: {2}", ip, mac, hostname);
}
```

## SNMP\_Machine.cs

מחלקה היורשת ממחלקה Machine. המשתנים שלה:

```
public class SNMP_Machine : Machine
{
    private string sysName; // machine name
    private string sysDescription; // machine operating system description
    private string sysContact; // machine contact name
    private string sysLocation; // machine location
```

בנאי של המחלקה:

```

public SNMP_Machine(string ip, string mac, string hostname, bool isMyComputer,
    bool isRouter, string sysName, string sysDescription, string sysContact, string sysLocation)
    : base(ip, mac, hostname, isMyComputer, isRouter)
{
    if(sysName==" " || sysName==null)
    {
        this.sysName = "Unknown";
    }
    else
    {
        this.sysName = sysName;
    }

    if (sysDescription == " " || sysDescription==null)
    {
        this.sysDescription = "Unknown";
    }
    else
    {
        this.sysDescription = sysDescription;
    }

    if(sysContact==" " || sysContact==null)
    {
        this.sysContact= "Unknown";
    }
    else
    {
        this.sysContact = sysContact;
    }

    if(sysLocation==" " || sysLocation==null)
    {
        this.sysLocation = "Unknown";
    }
    else
    {
        this.sysLocation = sysLocation;
    }
}

```

פעולות Get ו-Set של המחלקה:

```

public string SysName
{
    get { return sysName; }
}

public string SysDescription
{
    get { return sysDescription; }
}

public string SysContact
{
    get { return sysContact; }
}

public string SysLocation
{
    get { return sysLocation; }
}

```

פונקציה המחזירה כמה זמן עבר (בדקות) מאז הפעם האחרונה ש-SNMP Services של המכשיר אותחל. הפונקציה מריצה קובץ ב-Python המחזיר את הזמן הזה.

```

public int GetSysUpTime() // get the time passed since the network management portion of the systems
                          // was last re-initialized
{
    //running python script that get the time passed since the machine started using snmp services
    Process process = new Process();
    process.StartInfo.FileName = @"D:\Program Files\Python27\python.exe";
    process.StartInfo.Arguments = @"D:\MyFolder\Python_Projects\getUpTime.py " + this.IP;
    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.Start();
    process.WaitForExit();
    return process.ExitCode; //return value from python scripts (the value is the uptime)
}

```

פעולת ToString של המחלקה:



```

public override string ToString()
{
    string s = base.ToString() + "\n";
    int upTime = this.GetSysUpTime();
    if (upTime > 60) //if uptime is greater than 60 minutes
    {
        upTime /= 60; //convert the uptime to hours
        s += string.Format("Machine Up time: {0} hours\n", upTime);
    }
    else
    {
        s += string.Format("Machine Up time: {0} minutes\n", upTime);
    }
    s += string.Format("Machine name: {0}\n", sysName);
    s += string.Format("Machine description: {0}\n", sysDescription);
    s += string.Format("Machine contact name: {0}\n", sysContact);
    s += string.Format("Machine location: {0}", sysLocation);
    return s;
}

```

## AllMachines.cs

מחלקה המכילה רשימה של "מכשירים".  
יש לה משתנה "number\_hostname", כדי למנוע מצבים  
שיש מכשירים שה-hostname שלהם לא ידוע, ואז יהיה  
להם אותו-hostname ("unknown hostname").  
בפונקציה Add ניתן לראות שימוש במשתנה  
:number\_hostname

```

public class AllMachines
{
    private List<Machine> machines;
    private int number_hostname; //being used in Add method

    public AllMachines()
    {
        machines = new List<Machine>();
        number_hostname = 1;
    }

    public void Add(Machine m) // adds a machine to the machines list
    {
        if (m.Hostname == "Unknown hostname")
        {
            m.Hostname = "Unknown hostname " + number_hostname; //prevent situation that different machines
            number_hostname++; //have the same hostname(machine1.hostname="Unknown hostname1"
            //,machine2.hostname="Unknown hostname 2")
        }
        machines.Add(m);
    }
}

```

פעולות Get ו-Set:

```

public int Number_Hostname
{
    get { return number_hostname; }
    set { number_hostname = value; }
}

public List<Machine> GetAllMachines()
{
    return machines;
}

```

פונקציה המחזירה True אם הרשימה של המכשירים ריקה.  
אחרת, מחזירה False:

```

public bool IsEmpty() //returns true or false if the machines list is empty
{
    return machines.Count == 0;
}

```

---

פונקציה המקבלת "מכשיר", ומחזירה True אם המכשיר  
נמצא ברשימה של "מכשירים". אחרת, מחזירה False.

```

public bool IsMachineExists(Machine m) //returns true or false if the machines exists
{
    foreach (Machine m2 in machines)
    {
        if (m2.IP == m.IP)
        {
            return true;
        }
    }
    return false;
}

```

---

פונקציה MyComputer() המחזירה את המחשב שלי

מהרשימה, ופונקציה Router() המחזירה את הראוטר:

```
public Machine MyComputer() //returns my computer from the machines list
{
    foreach (Machine m in machines)
    {
        if (m.IsMyComputer)
        {
            return m;
        }
    }
    return null;
}

public Machine Router() // returns the router from the machines list
{
    foreach (Machine m in machines)
    {
        if (m.IsRouter)
        {
            return m;
        }
    }
    return null;
}
```

פונקציה המחזירה רשימה של מכשירים מסוג  
SNMP\_Machine ושהם לא המחשב שלי או הראוטר:

```
public List<SNMP_Machine> SNMP_Machines() //returns list of snmp machines that not my computer or the router
{
    List<SNMP_Machine> lst = new List<SNMP_Machine>();
    foreach (Machine m in machines)
    {
        if (!(m.IsMyComputer) && !(m.IsRouter) && (m is SNMP_Machine))
        {
            lst.Add(((SNMP_Machine)m));
        }
    }
    return lst;
}
```

פונקציה המחזירה רשימה של מכשירים שהם לא המחשב  
שלי או הראוטר, ושהם לא מסוג SNMP\_Machine:

```

public List<Machine> OtherMachines() //returns list of machines that not snmp machines and not my computer or the router
{
    List<Machine> lst = new List<Machine>();
    foreach (Machine m in machines)
    {
        if (!(m.IsMyComputer) && !(m.IsRouter) && !(m is SNMP_Machine))
        {
            lst.Add(m);
        }
    }
    return lst;
}

```

## פעולת ToString של המחלקה:

```

public override string ToString()
{
    string s = "";
    for (int i = 0; i < machines.Count; i++)
    {
        s += machines[i].ToString();
        if (i != (machines.Count - 1)) // if not the last element in the list
        {
            s += "\n";
        }
    }
    return s;
}

```

## Form1.cs

משתנים במחלקה הזאת:

**machines** - מייצג את ה-Database של המחשבים ברשת הפנימית.

**myComputer** - מייצג את המחשב שלי.

**router** - מייצג את הראוטר שלי.

**status** - מייצג את הפלט שיש לכל צומת בעץ.

```

public partial class Form1 : Form
{
    private AllMachines machines; //instance of AllMachine that represent the all machines in the local network
    private Machine myComputer; //instance of Machine that represent the router
    private Machine router; //instance of Machine that represent the router
    private string status = "hostname"; //status of the nodes output in the tree
}

```

בנאי:

```

public Form1()
{
    InitializeComponent();
    this.StartPosition = FormStartPosition.CenterScreen; // the form will be in the center of the screen
}

```

## כשה-Form1 נטען:

```

private void Form1_Load(object sender, EventArgs e) // this function start when loading the form
{
    Thread thread = new Thread(ScanAll); // running python script "scan_all.py"
    thread.Start();

    //get my computer and the router from json file
    AllMachines myComputer_router = LoadJsonMyComputer_Router();
    myComputer = myComputer_router.MyComputer();
    router = myComputer_router.Router();
    machines = new AllMachines(); //define the database
    //adding my computer and the router to the database
    machines.Add(myComputer);
    machines.Add(router);
    // creating nodes that represent my computer and the router
    TreeNode myComputer_node = new TreeNode();
    myComputer_node.Tag = myComputer;
    myComputer_node.Name = myComputer.IP;
    myComputer_node.Text = myComputer.Hostname;
    treeView1.Nodes.Add(myComputer_node); // adding my computer's node to the root of the tree
    TreeNode router_node = new TreeNode();
    router_node.Tag = router;
    router_node.Name = router.IP;
    router_node.Text = router.Hostname;
    treeView1.Nodes[0].Nodes.Add(router_node); // adding router's node to be the child of my computer's node
    // adding two categories of machines
    treeView1.Nodes[0].Nodes.Add("SNMP Computers");
    treeView1.Nodes[0].Nodes.Add("Other");
    // set colors of the treeview, my computer's node and the router's node
    treeView1.BackColor = Color.Yellow;
    treeView1.Nodes[0].BackColor = Color.LightSkyBlue;
    treeView1.Nodes[0].Nodes[0].BackColor = Color.LightSteelBlue;
}

```

## איך ה-Form1 נראה:



כפתור המשנה את הטקסט של הצמתים בעץ לכתובות IP

## של המכשירים שהם מייצגים:

```
private void button1_Click(object sender, EventArgs e) //show ip button
{
    status = "ip";
    ChangeNodesTextToIP(treeView1.Nodes[0], machines, status);
}

public static void ChangeNodesTextToIP(TreeNode tn, AllMachines machines, string status)
    //changing the nodes text to ip addresses
{
    if (tn.Name!="") // if the node represent machine
    {
        tn.Text = tn.Name; // the text will be the node's name (node's name is the ip of the machine)
    }
    foreach (TreeNode tn2 in tn.Nodes) // going through all nodes of the tree
    {
        ChangeNodesTextToIP(tn2, machines, status);
    }
}
```

## כפתור המשנה את הטקסט של הצמתים בעץ לכתובות הפיזיות של המכשירים שהם מייצגים:

```
private void button2_Click(object sender, EventArgs e) //show mac button
{
    status = "mac";
    ChangeNodesTextToMac(treeView1.Nodes[0], machines, status);
}

public static void ChangeNodesTextToMac(TreeNode tn, AllMachines machines, string status)
    //changing the nodes text to mac addresses
{
    if (tn.Name!="") // if the node represent machine
    {
        tn.Text = NodeToMachine(tn).Mac; // the text will be the node's mac address
    }
    foreach (TreeNode tn2 in tn.Nodes) // going through all nodes of the tree
    {
        ChangeNodesTextToMac(tn2, machines, status);
    }
}
```

## כפתור המשנה את הטקסט של הצמתים בעץ ל-hostnames של המכשירים שהם מייצגים:

```

private void button3_Click(object sender, EventArgs e) //show hostnames button
{
    status = "hostname";
    ChangeNodesTextToHostname(treeView1.Nodes[0], machines, status);
}

public static void ChangeNodesTextToHostname(TreeNode tn, AllMachines machines, string status)
//changing the nodes text to hostnames
{
    if (tn.Name!="") // if the node represent machine
    {
        tn.Text = NodeToMachine(tn).Hostname; //the text will be the node's hostname
    }
    foreach (TreeNode tn2 in tn.Nodes) // going through all nodes of the tree
    {
        ChangeNodesTextToHostname(tn2, machines, status);
    }
}

```

אירוע שמתרחש כאשר מקליקים פעמיים על צומת בעץ המייצגת מכשיר (המידע עליו יוצג על המסך):

```

private void treeView1_MouseDoubleClick(object sender, MouseEventArgs e)
//showing the selected node information
{
    TreeNode node = treeView1.SelectedNode; // get the selected node
    if (node.Name!="") // if the node represent machine
    {
        MessageBox.Show(node.Tag.ToString()); // showing node information
    }
}

```

כפתור שמדליק את הצומת הנבחר:



```

private void button4_Click(object sender, EventArgs e) // turn on computer button
{
    TreeNode node = treeView1.SelectedNode; // get the selected node
    if(node.Name!="") // if the node represent machine
    {
        string mac = NodeToMachine(node).Mac; // get the node's mac address
        TurnOnComputer(mac); // turning on the computer with this mac address
    }
}

public static void TurnOnComputer(string mac)
// run python script that turn on computer with the given mac address
{
    RunPythonScript(@"D:\MyFolder\Python_Scripts\file_wakeonlan.py " + mac,true);
}

```

## כפתור המשנה את הטקסט של הצמתים שמייצגים SNMP\_Machines ל-Uptime שלהם:

```

private void button5_Click(object sender, EventArgs e)
// changing snmp machines nodes text to the time in minutes since they started using snmp services
{
    TreeNode myComputer_node = treeView1.Nodes[0]; // get the node of my computer
    if(myComputer_node.Tag is SNMP_Machine) // if my computer is snmp_machine
    {
        myComputer_node.Text = ((SNMP_Machine)myComputer_node.Tag).GetSysUpTime().ToString();
    }
    TreeNode router_node = treeView1.Nodes[0].Nodes[0]; // get the node of my router
    if(router_node.Tag is SNMP_Machine) // if my router is snmp_machine
    {
        router_node.Text = ((SNMP_Machine)router_node.Tag).GetSysUpTime().ToString();
    }

    foreach(TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes) // child nodes of snmp machines
    {
        node.Text = ((SNMP_Machine)node.Tag).GetSysUpTime().ToString();
    }
}

```

## כפתור המשנה את הטקסט של הצמתים שמייצגים SNMP\_Machines לשם שלהם:

```

private void button6_Click(object sender, EventArgs e)
    // changing snmp machines nodes text to their system name
{
    TreeNode myComputer_node = treeView1.Nodes[0];
    if (myComputer_node.Tag is SNMP_Machine)
    {
        myComputer_node.Text = ((SNMP_Machine)myComputer_node.Tag).SysName.ToString();
    }
    TreeNode router_node = treeView1.Nodes[0].Nodes[0];
    if (router_node.Tag is SNMP_Machine)
    {
        router_node.Text = ((SNMP_Machine)router_node.Tag).SysName.ToString();
    }

    foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes)
    {
        node.Text = ((SNMP_Machine)node.Tag).SysName.ToString();
    }
}

```

## כפתור המשנה את הטקסט של הצמתים שמייצגים SNMP\_Machines לתאור של שלהם:

```

private void button7_Click(object sender, EventArgs e)
    // changing snmp machines nodes text to their system description
{
    TreeNode myComputer_node = treeView1.Nodes[0];
    if (myComputer_node.Tag is SNMP_Machine)
    {
        myComputer_node.Text = ((SNMP_Machine)myComputer_node.Tag).SysDescription.ToString();
    }

    TreeNode router_node = treeView1.Nodes[0].Nodes[0];
    if (router_node.Tag is SNMP_Machine)
    {
        router_node.Text = ((SNMP_Machine)router_node.Tag).SysDescription.ToString();
    }

    foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes)
    {
        node.Text = ((SNMP_Machine)node.Tag).SysDescription.ToString();
    }
}

```

## כפתור המשנה את הטקסט של הצמתים שמייצגים SNMP\_Machines ל-Contact name שלהם:

```

private void button8_Click(object sender, EventArgs e) //Machines contact name button
    // changing snmp machines nodes text to their contact name
{
    TreeNode myComputer_node = treeView1.Nodes[0];
    if (myComputer_node.Tag is SNMP_Machine)
    {
        myComputer_node.Text = ((SNMP_Machine)myComputer_node.Tag).SysContact.ToString();
    }

    TreeNode router_node = treeView1.Nodes[0].Nodes[0];
    if (router_node.Tag is SNMP_Machine)
    {
        router_node.Text = ((SNMP_Machine)router_node.Tag).SysContact.ToString();
    }

    foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes)
    {
        node.Text = ((SNMP_Machine)node.Tag).SysContact.ToString();
    }
}

```

## כפתור המשנה את הטקסט של הצמתים שמייצגים SNMP\_Machines למיקום שלהם:

```

private void button9_Click(object sender, EventArgs e)
    // changing snmp machines nodes text to their location
{
    TreeNode myComputer_node = treeView1.Nodes[0];
    if (myComputer_node.Tag is SNMP_Machine)
    {
        myComputer_node.Text = ((SNMP_Machine)myComputer_node.Tag).SysLocation.ToString();
    }

    TreeNode router_node = treeView1.Nodes[0].Nodes[0];
    if (router_node.Tag is SNMP_Machine)
    {
        router_node.Text = ((SNMP_Machine)router_node.Tag).SysLocation.ToString();
    }

    foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes)
    {
        node.Text = ((SNMP_Machine)node.Tag).SysLocation.ToString();
    }
}

```

## כפתור אחד שמתחיל לעדכן את העץ וכפתור שני שעוצר את העדכון שלו:

```
private void button10_Click(object sender, EventArgs e) //start updating button
{
    timer1.Enabled=true; // enable timer1
    textBox1.Text = "Updating";
}
```

```
private void button11_Click(object sender, EventArgs e) //stop updating button
{
    timer1.Enabled=false; // disable timer1
    textBox1.Text = "Not Updating";
}
```

ברגע שה-timer מופעל, הוא יפעיל את קטע הקוד הבא כל 5 שניות (זה יעדכן את העץ):

```
private void timer1_Tick(object sender, EventArgs e) //timer for checking new computers
{
    timer1.Stop();
    UpdateTree(treeView1, ref machines, status);
    timer1.Start();
}
```

timer שכל 10 שניות בודק אם מחשבים נכבו (או נותקו מהאינטרנט):

```
private void timer2_Tick(object sender, EventArgs e) //timer for checking shutdown computers
{
    timer2.Stop();
    CheckForShutDownComputers(treeView1);
    timer2.Start();
}
```

פונקציה המקבלת נתיב לקובץ מסוג Json ומחזירה את הרשימה של dictionaries שיש בו.

```

public static List<Dictionary<string, string>> LoadJson(string path) //return dictionary from json file
{
    try
    {
        using (StreamReader r = new StreamReader(path))
        {
            string json = r.ReadToEnd();
            List<Dictionary<string, string>> items =
                JsonConvert.DeserializeObject<List<Dictionary<string, string>>>(json);
            return items;
        }
    }
    catch(FileNotFoundException)
    {
        return null;
    }
}

```

פונקציה המקבלת dictionary ומחזירה Machine או  
SNMP\_Machine עם המידע שיש ב-dictionary:

```

public static Machine DictionaryToMachine(Dictionary<string, string> dictionary)
//convert dictionary to mahchine instance
{
    string ip = dictionary["ip"];
    string mac = dictionary["mac"];
    string hostname = dictionary["hostname"];
    bool isMyComputer = (dictionary["myComputer"] == "true");
    bool isRouter = (dictionary["router"] == "true");
    bool isUsingSNMP = (dictionary["isUsingSNMP"] == "true");
    if (isUsingSNMP)
    {
        string sysName = dictionary["sysName"];
        string sysDescription = dictionary["sysDescription"];
        string sysContact = dictionary["sysContact"];
        string sysLocation = dictionary["sysLocation"];
        return new SNMP_Machine(ip, mac, hostname, isMyComputer, isRouter, sysName,
            sysDescription, sysContact, sysLocation);
    }
    return new Machine(ip, mac, hostname, isMyComputer, isRouter);
}

```

פונקציה המקבלת נתיב לקובץ מסוג json שיש בו רשימה  
של dictionaries המייצגים "מכשירים", ומחזירה מופע

של AllMachines, שכל מכשיר בו הוא המכשיר שהיה בקובץ:

```
public static AllMachines Get_Machines(string path) //returns all machines
{
    List<Dictionary<string,string>> lst = LoadJson(path);
    if (lst != null && lst.Count!=0)
    {
        AllMachines machines = new AllMachines();
        foreach (Dictionary<string,string> dictionary in lst)
        {
            machines.Add(DictionaryToMachine(dictionary));
        }
        return machines;
    }
    return null;
}
```

פונקציות המחזירות מידע מקבצי Json:

```
public static AllMachines LoadJsonMyComputer_Router() //return my computer and the router
{
    return Get_Machines(@"D:\MyFolder\Python_Scripts\myComputer_router.json");
}

public static AllMachines LoadJsonNewComputers() // return new computers that scanned
{
    return Get_Machines(@"D:\MyFolder\Python_Scripts\new_computers.json");
}

public static List<Dictionary<string, string>> LoadJsonShutdownComputers() //return info about known computers
{
    return LoadJson(@"D:\MyFolder\Python_Scripts\shutdown_computers.json"); ;
}
```

פונקציה המקבלת צומת בעץ, ומחזירה מופע של Machine המייצג אותו:

```
public static Machine NodeToMachine(TreeNode node) // returns machine instance that represent the node
{
    return ((Machine)node.Tag);
}
```

פונקציה המקבלת עץ וכתובת IP ומחזירה את הצומת עם כתובת ה-IP הזאת.

```

public static TreeNode GetNodeByIP(TreeView treeView1, string ip) //returns the node with the given ip address
{
    TreeNode myComputer_node = treeView1.Nodes[0];
    if(myComputer_node.Name==ip)
    {
        return myComputer_node;
    }
    TreeNode router_node = treeView1.Nodes[0].Nodes[0];
    if(router_node.Name==ip)
    {
        return router_node;
    }

    foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes) //nodes of snmp machines
    {
        if (node.Name == ip)
        {
            return node;
        }
    }

    foreach (TreeNode node in treeView1.Nodes[0].Nodes[2].Nodes) //nodes of other
    {
        if(node.Name==ip)
        {
            return node;
        }
    }
    return null;
}

```

פונקציה המקבלת צומת ואת המשתנה "status", ומחזירה את הטקסט שצריך להיות לצומת בעץ.

```

public static string GetNodeOutput(TreeNode node, string status) //get the node output that need to be in the tree
{
    if(status=="ip")
    {
        return NodeToMachine(node).IP;
    }
    if(status=="mac")
    {
        return NodeToMachine(node).Mac;
    }
    return NodeToMachine(node).Hostname;
}

```

פונקציה המקבלת עץ וצומת, ומחזירה True אם הצומת קיים בעץ, אחרת- מחזירה False.

```

public static bool NodeExists(TreeView treeView1,TreeNode node) //return true or false if the node exists
{
    TreeNode myComputer_node = treeView1.Nodes[0];
    if(myComputer_node.Name==node.Name)
    {
        return true;
    }

    TreeNode router_node = treeView1.Nodes[0].Nodes[0];
    if(router_node.Name==node.Name)
    {
        return true;
    }

    foreach (TreeNode tn in treeView1.Nodes[0].Nodes[1].Nodes) //nodes of snmp machines
    {
        if (tn.Name == node.Name)
        {
            return true;
        }
    }

    foreach (TreeNode tn in treeView1.Nodes[0].Nodes[2].Nodes) //nodes of other
    {
        if(tn.Name==node.Name)
        {
            return true;
        }
    }
    return false;
}

```

פונקציה המקבלת נתיב לסקריפט ב-Python ו-arguments בשביל (למשל C:\file.py 10.0.0.3) ומקבלת גם true האם להחביא את החלון של ההרצה של הסקריפט, או false - לא להחביא את החלון:

```

public static void RunPythonScript(string arguments, bool window_hidden) //run python script
                                                                    //arguments- path+args
{
    Process process = new Process();
    process.StartInfo.FileName = @"D:\Program Files\Python27\python.exe";
    process.StartInfo.Arguments = arguments;
    if (window_hidden)
    {
        //the window of the process will be hidden
        process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    }
    process.Start();
    process.WaitForExit();
}

```

פונקציה המריצה קובץ ב-Python שסורק את הרשת הפנימית.



```
public static void ScanAll() //running python scan_all.py script (scanning local network)
{
    RunPythonScript(@"D:\MyFolder\Python_Scripts\scan_all.py", false);
}
```

פונקציה המקבלת string המכיל כתובות ip ומריצה קובץ  
ב-Python שבודק אילו מחשבים בעלי כתובות ה-ip האלו,  
כבויים ואילו מחשבים דלוקים.

```
public static void RunGetShutdownComputers(string all_ip) // running python getShutdownComputers.py script
{
    RunPythonScript(@"D:\MyFolder\Python_Scripts\getShutdownComputers.py " + all_ip, true);
}
```

פונקציה המוסיפה מכשירים חדשים שגולו לעץ:

```

public static void UpdateTree(TreeView treeView1, ref AllMachines machines, string status) // update the treeview
{
    foreach(TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes) //nodes of snmp machines
    {
        if(node.BackColor==Color.Pink) //if the color of the node is pink
        {
            node.BackColor = Color.Empty; //set the color to default
        }
    }
    foreach (TreeNode node in treeView1.Nodes[0].Nodes[2].Nodes) //nodes of other
    {
        if (node.BackColor == Color.Pink) //if the color of the node is pink
        {
            node.BackColor = Color.Empty; //set the color to default
        }
    }

    // get machines from json file
    AllMachines new_machines = LoadJsonNewComputers();
    if (new_machines != null)
    {
        List<SNMP_Machine> snmp_machines = new_machines.SNMP_Machines(); //list of snmp machines
        List<Machine> other = new_machines.OtherMachines(); //list of other machines
        foreach(SNMP_Machine m in snmp_machines)
        {
            if(!machines.IsMachineExists(m)) // if machine not in the database
            {
                machines.Add(m); //add machine to the database
                //creating node, his tag is the machine, his name is the machine's ip
                //the node's color will be pink
                TreeNode node = new TreeNode();
                //machines define m hostname
                node.Tag = machines.GetAllMachines()[machines.GetAllMachines().Count - 1];
                node.Name = m.IP;
                node.Text = GetNodeOutput(node, status); //get the node output that need to be in the tree
                node.BackColor = Color.Pink;
                treeView1.Nodes[0].Nodes[1].Nodes.Add(node); //add node to snmp machines
            }
        }

        foreach (Machine m in other)
        {
            if (!machines.IsMachineExists(m)) // if machine not in the database
            {
                machines.Add(m); //add machine to the database
                //creating node, his tag is the machine, his name is the machine's ip
                //the node's color will be pink
                TreeNode node = new TreeNode();
                //machines define m hostname
                node.Tag = machines.GetAllMachines()[machines.GetAllMachines().Count - 1];
                node.Name = m.IP;
                node.Text = GetNodeOutput(node, status); //get the node output that need to be in the tree
                node.BackColor = Color.Pink;
                treeView1.Nodes[0].Nodes[2].Nodes.Add(node); //add node to other machines
            }
        }
    }
}

```

פונקציה המדגישה אילו מחשבים כבויים (או שנותקו מהאינטרנט) ומדגישה עם מחשב נדלק לאחר שנכבה (או

## שהתחבר לאינטרנט לאחר שנותק ממנו).

```
public static void CheckForShutDownComputers(TreeView treeView1) //colors shutdown computers
{
    string all_ip = "";
    foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes) //nodes of snmp machines
    {
        all_ip += node.Name + " ";
    }
    foreach (TreeNode node in treeView1.Nodes[0].Nodes[2].Nodes) //nodes of other
    {
        all_ip += node.Name + " ";
    }

    //getting status (up or down) for each node
    RunGetShutdownComputers(all_ip); // running python script getShutdownComputers.py
    List<Dictionary<string, string>> lst = LoadJsonShutdownComputers();
    if (lst != null)
    {
        foreach (Dictionary<string, string> dictionary in lst) //lst=[{"ip":"0.0.0.0","valid":true/false}]
        {
            string ip = dictionary["ip"];
            bool valid = (dictionary["valid"] == "true");
            TreeNode node = GetNodeByIP(treeView1, ip); //getting the node from the tree with the given ip
            if (node != null)
            {
                if (valid) // if node is up
                {
                    if (node.BackColor == Color.Red) //if it was in red color
                    {
                        node.BackColor = Color.Green; //turn its color to greeb
                    }
                    else if (node.BackColor == Color.Green) //if it in green color
                    {
                        node.BackColor = Color.Empty; //set the color to the default
                    }
                }
                else //if node is down
                {
                    if (node.BackColor != Color.Red) //if node color is not red
                    {
                        node.BackColor = Color.Red; //set the node color to red
                    }
                }
            }
        }
    }
}
```

## cmdinteracting.py

סקריפט השולף מידע מ-cmd של windows כמו כתובת IP של המחשב שלי. התפקיד העיקרי שלו, הוא לחשב את

כתובות ה-IP שיכולות להיות ב-subnet mask שלי.

פונקציות עיקריות:

**def myIP() -**

פונקציה המחזירה את כתובת ה-IP שלי.

**def myMacAddress()-**

פונקציה המחזירה את הכתובת הפיזית שלי.

**def routerIP()-**

פונקציה המחזירה את הכתובת IP של הראוטר.

**def routerMacAddress()-**

פונקציה המחזירה את הכתובת הפיזית שלי.

**def subnetMask()-**

פונקציה המחזירה את ה-subnet mask.

**def getAll\_IP()-**

פונקציה המחזירה רשימה שיש לה כמה רשימות בעלי אורך שווה, שבכל אחת מהן יש כתובות IP המתאימות ל-subnet mask ולכתובת ה-IP של המחשב שלי.

**scan\_all.py**

סקריפט הסורק את כל הרשת הפנימית.  
מחלקה Machine המייצגת "מכשיר":

```

6 class Machine:
7     def __init__(self, ip, mac, hostname):
8         self.ip=ip
9         self.mac=mac
10        self.hostname=hostname
11        self.myComputer=(ip==cmdInteracting.myIP())
12        self.router=(ip==cmdInteracting.routerIP())
13        if self.router and self.hostname=="Unknown hostname":
14            self.hostname="Router"
15        self.isUsingSNMP=isSNMP(self.ip)
16        if self.isUsingSNMP:
17            self.sysName=getSysName(self.ip)
18            self.sysDescription=getSysDescription(self.ip)
19            self.sysContact=getSysContact(self.ip)
20            self.sysLocation=getSysLocation(self.ip)
21
22
23        def dict(self): #convert the machine do dictionary
24            d={}
25            d["ip"]=self.ip
26            d["mac"]=self.mac
27            d["hostname"]=self.hostname
28            d["myComputer"]=self.myComputer
29            d["router"]=self.router
30            d["isUsingSNMP"]=self.isUsingSNMP
31            if self.isUsingSNMP:
32                d["sysName"]=self.sysName
33                d["sysDescription"]=self.sysDescription
34                d["sysContact"]=self.sysContact
35                d["sysLocation"]=self.sysLocation
36            return d
37
38        def __str__(self):
39            return str(self.dict())

```

פונקציות עיקריות:

### def ping(ip,lst,lock)-

פונקציה המקבלת כתובת ip, רשימה של המכשירים שגולו, ומנעול (multiprocessing.Lock) היא בודקת אם יש מחשב קיים בעל כתובת ה-ip הזאת. אם קיים מחשב כזה, אז זה מוסיף מכשיר לרשימה של המכשירים ומחזיר true. אם אין מחשב כזה, היא מחזירה false.

### Write Json(manager list)-

פונקציה המקבלת רשימה של המכשירים וכותבת אותם לקובץ מסוג json.

## getMyComputer\_router.py

סקריפט הכותב מידע על המחשב שלי ועל הראוטר ב-Json file.

```
125 lst=[]
126 #getting my computer and the router
127 myComputer=get_myComputer().dict() #g
128 router=get_router().dict()
129 lst.append(myComputer)
130 lst.append(router)
131 Write_Json(lst) #write my copmputer and the router to json file
132
```

יש מחלקה Machine:

```
6 class Machine:
7     def __init__(self,ip,mac,hostname):
8         self.ip=ip
9         self.mac=mac
10        self.hostname=hostname
11        self.myComputer=(ip==cmdInteracting.myIP())
12        self.router=(ip==cmdInteracting.routerIP())
13        if self.router and self.hostname=="Unknown hostname":
14            self.hostname="Router"
15        self.isUsingSNMP=isSNMP(self.ip)
16        if self.isUsingSNMP:
17            self.sysName=getSysName(self.ip)
18            self.sysDescription=getSysDescription(self.ip)
19            self.sysContact=getSysContact(self.ip)
20            self.sysLocation=getSysLocation(self.ip)
21
22
23        def dict(self): #convert the machine do dictionary
24            d={}
25            d["ip"]=self.ip
26            d["mac"]=self.mac
27            d["hostname"]=self.hostname
28            d["myComputer"]=self.myComputer
29            d["router"]=self.router
30            d["isUsingSNMP"]=self.isUsingSNMP
31            if self.isUsingSNMP:
32                d["sysName"]=self.sysName
33                d["sysDescription"]=self.sysDescription
34                d["sysContact"]=self.sysContact
35                d["sysLocation"]=self.sysLocation
36            return d
37
38        def __str__(self):
39            return str(self.dict())
```

פונקציות עיקריות:

### def get\_myComputer()-

פונקציה המחזירה מופע של Machine המייצג את המחשב שלי.

### def get\_router()-

פונקציה המחזירה מופע של Machine המייצג את הראוטר.

### Write Json(lst):

פונקציה המקבלת רשימה שבה יש את המחשב שלי ואת הראוטר וכותבת אותה ל-Json file.

### getShutdownComputers.py

סקריפט המקבל כתובות IP מהעץ בתוכנית ב-C# ובודק אילו מחשבים דלוקים ואילו כבויים.  
פונקציות עיקריות:

### def ping(ip)-

פונקציה המקבלת כתובת IP ומחזירה true אם המחשב דלוק ו-false אם לא.

### def Write Json(x)-

פונקציה המקבלת רשימה של dictionaries שבהם כתובות ה-IP והאם המחשב דלוק או לא.

### file\_wakeonlan.py

סקריפט המקבל mac address של מחשב כבוי ושולח magic packet אליו על מנת שהמחשב יהיה דלוק.

```
1 from wakeonlan import wol
2 import sys
3
4
5 def turnOn(mac): #turn on the computer with the given mac address
6     wol.send_magic_packet(mac)
7
8 mac=sys.argv[1] #get mac address from the arguments given
9 mac=mac.replace("-",".") #mac="ff-ff-ff-ff-ff-ff" need to be "ff.ff.ff.ff.ff.ff"
10 mac=mac.lower() #change capital letters to small letters
11 turnOn(mac) #turning on the copmuter
12
```

## getUpTime.py

סקריפט המקבל כתובת IP של מחשב שמשתמש ב-SNMP Services ומחזיר את הזמן שעבר מאז הפעם האחרונה שה-SNMP Services אצלו אותחל.

```
1 from pysnmp.hlapi import *
2 import sys
3
4 def getSysUpTime(ip):
5     g = getCmd(SnmpEngine(), CommunityData('public'),
6               UdpTransportTarget((ip, 161)), ContextData(),
7               ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysUpTime', 0)))
8     try:
9         x=(next(g)[3][0][1])/6000
10        return x
11    except:
12        pass
13
14 ip=sys.argv[1]
15 upTime=getSysUpTime(ip)
16 sys.exit(upTime)
17
```

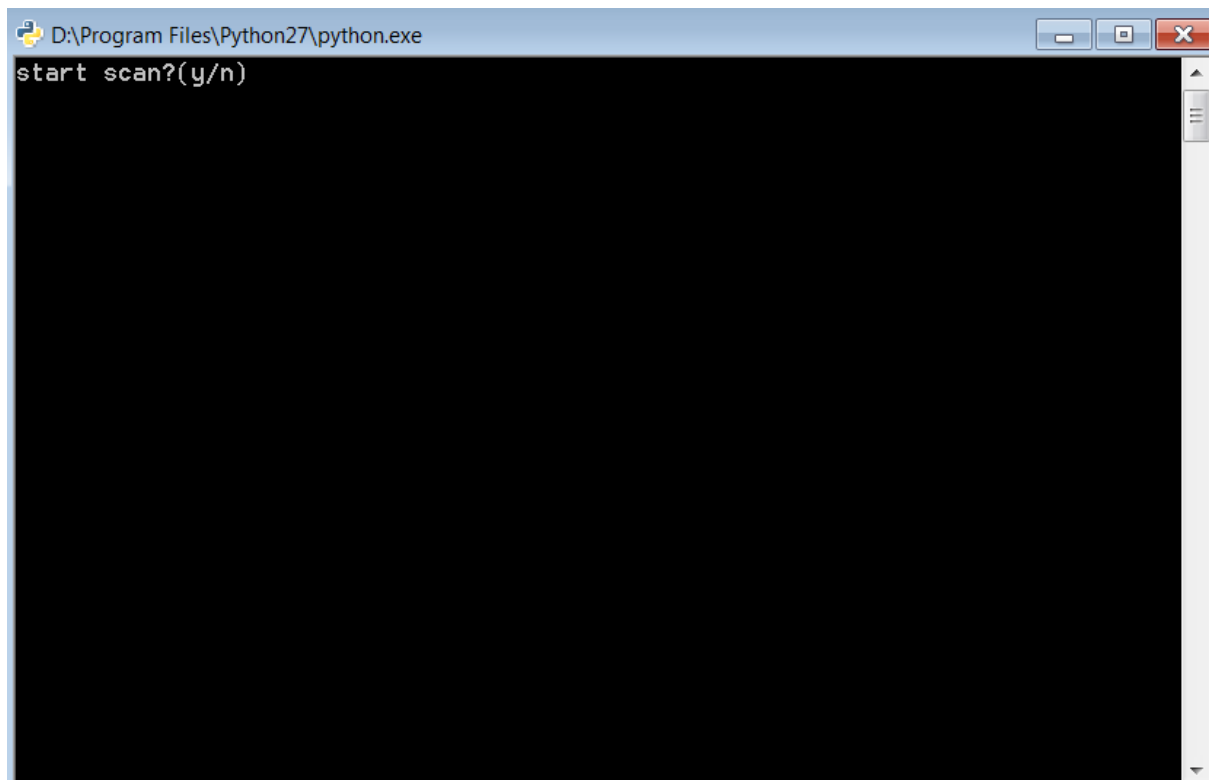
## השוואת העבודה עם פתרונות ויישומים קיימים

מטרת הפרויקט שלי היא להציג נתונים על הרשת הפנימית במהירות המרבית ביותר ולהציג בצורה נוחה ביותר. Nmap, דוגמה ליישום קיים שמקיים מטרה זאת, עושה זאת במהירות גבוהה יותר ממה שאני הצלחתי להשיג. יישום זה גם מצליח לראות את הנתיב שחבילת מידע עוברת ממחשב למחשב.



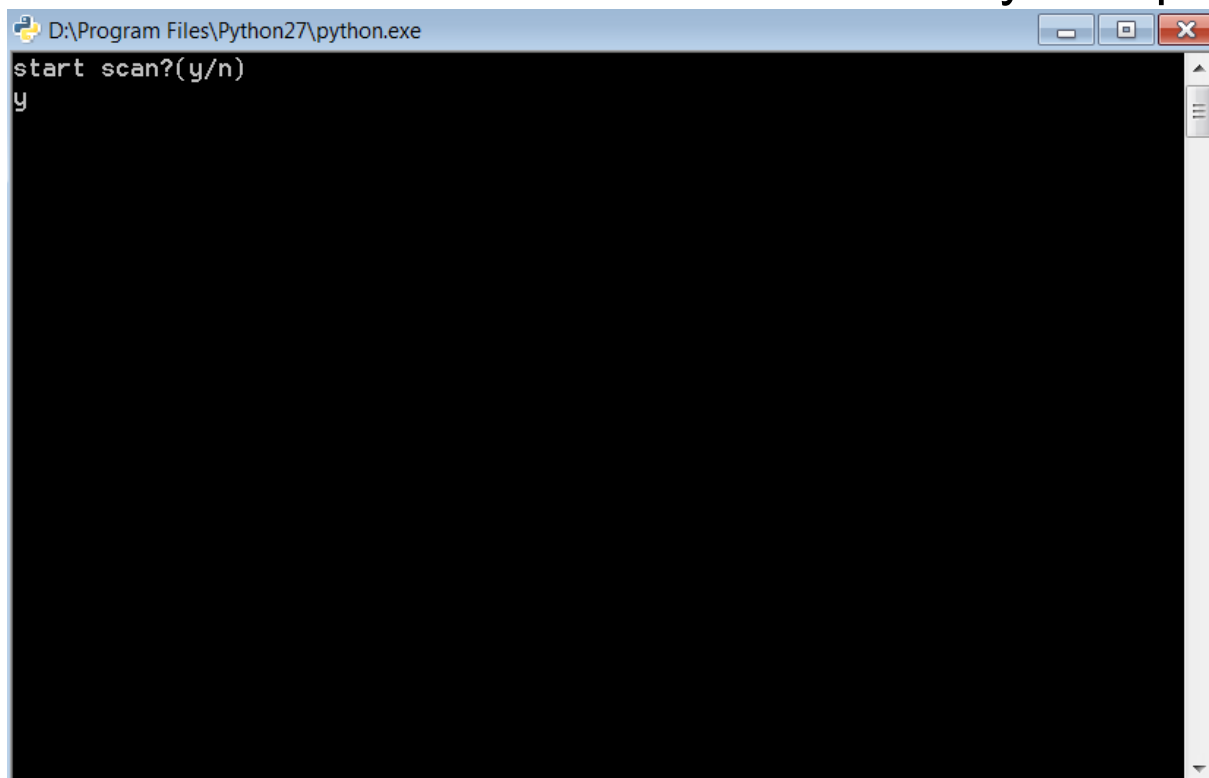
יישום זה גם מצליח לעקוף firewall למרות מה שאני עשיתי.  
אבל, מטרת הפרויקט שלי היא לעשות בקר רשת בסיסי,  
והצלחתי לעשות זאת.

**תאור של הממשק למשתמש - הוראות הפעלה**  
מריצים את התוכנית TreeDisplay



A screenshot of a Windows command prompt window. The title bar shows the file path "D:\Program Files\Python27\python.exe". The command prompt displays the text "start scan?(y/n)" and is waiting for input. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

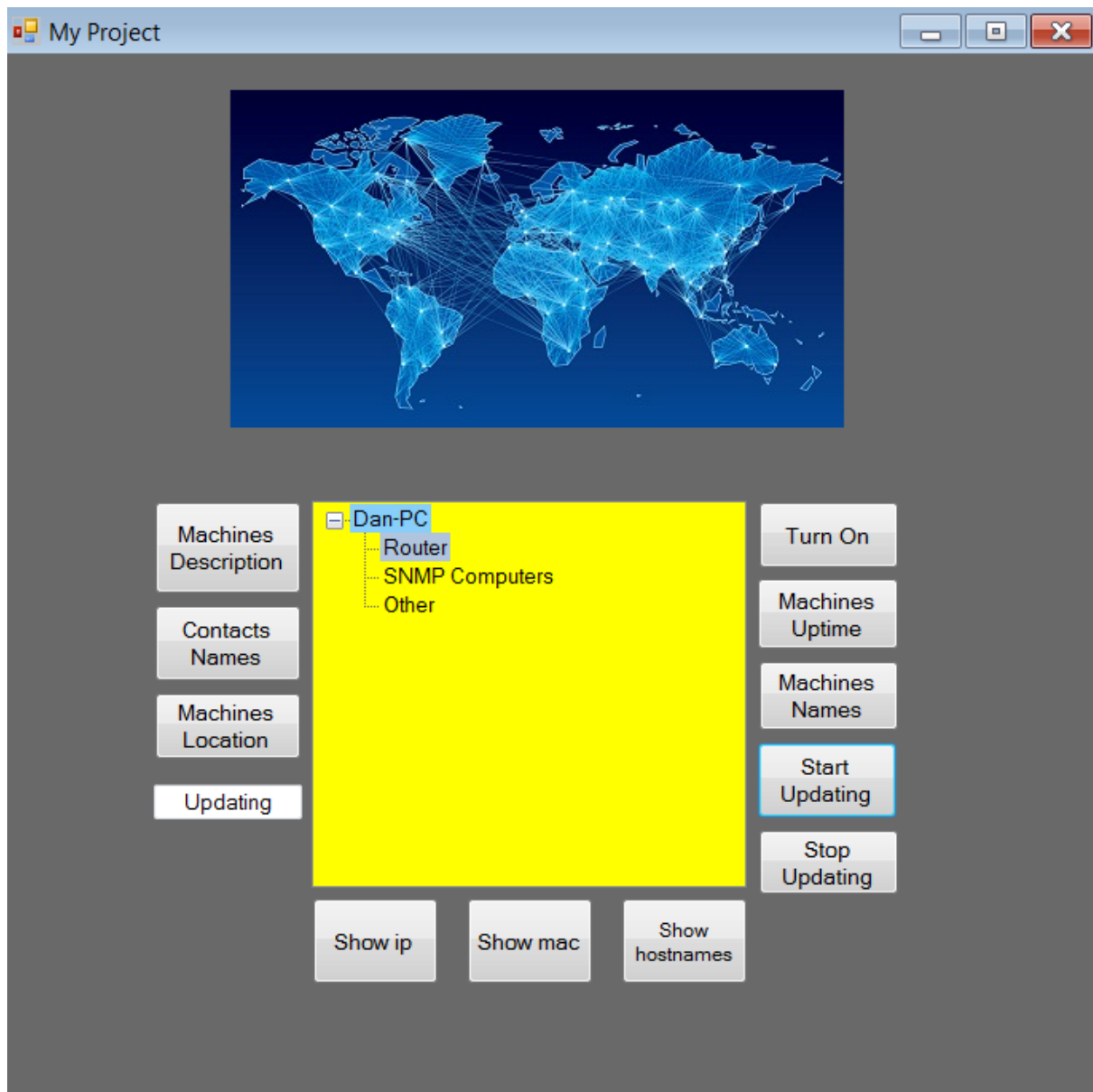
מקישים "y" ואז "Enter".



A screenshot of a Windows command prompt window, similar to the one above. The title bar shows the file path "D:\Program Files\Python27\python.exe". The command prompt displays the text "start scan?(y/n)". Below the prompt, the character "y" has been entered, indicating the user's response. The window has standard Windows window controls in the top right corner.



מקליקים על "Start Updating".



זהו, כעת לחכות שמחשבים נוספים יוספו לעץ.

### הוראות מיוחדות

**כדי להדליק מחשב-** להקליק על הצומת שלו פעם אחת ואז ללחוץ על הכפתור "Turn On".

**כדי לעצור את העדכון של העץ-** ללחוץ על הכפתור "Stop Updating".

**הכפתורים: show ip, show mac, show hostnames:**

משנים את הטקסט של כל הצמתים בעץ.

**הכפתורים: Machines Uptime, Machines Names,  
Machines Description, Contact Names,  
Machines Location: משנים את הטקסט רק של  
המכשירים שמשתמשים ב-SNMP Services.**

**מבט אישי על העבודה ותהליך הפיתוח**

העבודה על הפרויקט הייתה מעניינת ומאתגרת מאוד.  
נהייתי מהדרך שעשיתי כדי להתגבר על בעיות מסוימות.  
אחד הקשיים היה לסרוק את הרשת הפנימית במהירות  
גבוהה.

עשיתי זאת באמצעות המודול multiprocessing של  
Python והצלחתי לחלק את העבודה בסריקת כתובות ה-IP  
בין תהליכים שונים.

בסיום הפרויקט, אני יכול להגיד שלמדתי איך להיעזר  
באינטרנט בשביל תכנות בצורה יעילה ביותר, ולמדתי איך  
לקשר בין שפות תכנות שונות (C# ו-Python).

### **ביבליוגרפיה**

- אתר התמיכה של מייקרוסופט MSDN  
<https://msdn.microsoft.com>
- פורום עזרה למתכנתים Stack Overflow  
<https://stackoverflow.com>
- מדריך לשימוש בפרוטוקול SNMP ב-Python  
<http://pysnmp.sourceforge.net>