```
'''
Created on Apr 21, 2020

@author: klein

Gets all the csv files from the different directories and plots them
'''



import dropbox
import datetime
import numpy as np
from pathlib import Path # this is python 3
import matplotlib.pyplot as plt
import matplotlib.dates as md
from matplotlib.backends.backend_pdf import PdfPages
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
import ast
from os.path import expanduser



class PlotAll(object):
    '''
  classdocs
  '''


    def __init__(self, token_file , dir_list,filedate = None):
        '''
    Constructor
    '''
        #File for dropbox key
        home = expanduser("~")

        self.TokenFile = home+token_file

        # List of directories to check
        self.DirList = dir_list

        self.filedate = filedate



    def ConnectDropbox(self):
        """
    here we establish connection to the dropbox account
    """
        #f=open(self.keyfile,"r")
        #self.key =f.readline() #key for encryption
        #self.key = pad(self.key,16)
        #f.close()

        f=open(self.TokenFile,"r")
        self.data =f.readline() #key for encryption
```



```
        #connect to dropbox
        self.dbx=dropbox.Dropbox(self.data.strip('\n'))

        self.myaccount = self.dbx.users_get_current_account()
        print('**************************dropbox******************\n\n')
        print( self.myaccount.name.surname , self.myaccount.name.given_name)
        print (self.myaccount.email)
        print('\n\n **************************dropbox******************\n')


    def GetFiles(self):
        """
    This loops over the list of dropbox directories and gets the files for the current day if available
    """

        #First make the part of the file which is depending on the date
        self.MyFileName=MyFileName = self.GetCurrentFileName()

        #next block determines how many graphs we will do
        graph_count = 0

        for k in range(len(self.DirList)):
            temp = '/LCWA/'+self.DirList[k]+'/'+self.DirList[k]+MyFileName # fil
e on dropbox
            print(temp)
            if self.DropFileExists(temp):
                graph_count = graph_count+1

        self.graph_count=graph_count
        print ('we have',graph_count,' plots')

        # setup the canvas

        self.PlotSetup(graph_count)

        #Here starts the loop
        for k in range(len(self.DirList)):
            temp = '/LCWA/'+self.DirList[k]+'/'+self.DirList[k]+MyFileName # fil
e on dropbox
            temp_text = '/LCWA/'+self.DirList[k]+'/'+self.DirList[k]+MyFileName.
replace('csv','txt')
            temp_local = self.SetTempDirectory()+'/'+self.DirList[k]+MyFileName
            temp_local_text = self.SetTempDirectory()+'/'+self.DirList[k]+MyFile
Name.replace('csv','txt')
            if self.DropFileExists(temp):
                print ("getting file " ,temp, '  and storing it at :',temp_local)

                self.dbx.files_download_to_file(temp_local,temp)
                self.MyIP ='' #will be overwritten by readtextfile

                if self.DropFileExists(temp_text):
                    self.dbx.files_download_to_file(temp_local_text,temp_text)
                # Read the local file
```



```
                    self.ReadTextFile(temp_local_text)

                self.ReadFile(temp_local)

                self.ReadTestData()

                self.PlotTestData(k)
        #plt.show()
        #self.fig.savefig(self.pdffilepath, bbox_inches='tight')

        #plt.show()
        with PdfPages(self.pdffilepath) as pdf:
            pdf.savefig(self.fig)
            pdf.savefig(self.fig1)
            pdf.savefig(self.fig2)
            pdf.savefig(self.fig3)
            pdf.savefig(self.fig4)
            pdf.savefig(self.fig5)


        #self.pdf.savefig(self.fig)
        #self.fig.show()
        plt.close()

    def DropFileExists(self,path):
        try:
            self.dbx.files_get_metadata(path)
            return True
        except:
            return False


    def GetCurrentFileName(self):
        """
    this creates the part of the current filename which depends on the date
    """

        if(self.filedate == None):
            self.current_day = datetime.date.today()
            a = datetime.datetime.today().strftime('%Y-%m-%d')
            return a+'speedfile.csv'
        else:
            return self.filedate+'speedfile.csv'


    def ReadFile(self, InputFile):
        """ reads the csv file from the speedfile directory"""



        self.temp_name = self.SetTempDirectory()+'/temp.txt'
        self.temp_file = open(self.temp_name,'w')
        counter = 0
        for line in open(InputFile, 'r'):
            a = line.split('.')
            if(len(a)< 9):
                print ('problem',a)
```



```
                print ('ignore data point at line ',counter+1)
            else:
                self.temp_file.write(line)

            counter = counter+1



        self.temp_file.close()

    def ReadTestData(self):
        """
    Reads the results with Matplotlib
    """



        #self.legend = legend #legend is a dictionary'


        x1,y1,y2 , y0 = np.loadtxt(self.temp_name, delimiter='.',
                     unpack=True,usecols=(1,7,8,9),
                     converters={ 1: self.MyTime},skiprows = 1)

        self.x1 = x1 #time
        self.y1 = y1 #download
        self.y2 = y2 #upload
        self.y0 = y0 # before vs 6: packet loss, with vs 6: latency measured in
ms (averaged over 10)


    def SetTempDirectory(self):
        """
    this sets the directory for storing temporary files
    if the directory dos not exist it gets created. It is the scratch directory
    below the home directory
    """
        home = str(Path.home()) # get the home directory
        MyTempDir = home+'/scratch'
        # Now check if it exists, if no create it
        if(Path(MyTempDir)).exists():
            return MyTempDir
        else:
            Path(MyTempDir).mkdir()
            print("Creating ",MyTempDir)
            return MyTempDir

    def MyTime(self,b):
        """ conversion routine for time to be used in Matplotlib"""


        s=b.decode('ascii')

        a =md.date2num(datetime.datetime.strptime(s,'%H:%M:%S'))

        return a


    def PlotSetup(self,graph_count):
```



```
        """
    Creates the plotting environment
    """
        # we will have a max of 5 plots/ canvas
        #graph_count gives us the number we have
        #In a first tage we just get 4 plots on a canvas
        #create plotarrays
        row,column = 2,2
        self.fig, self.axarr = plt.subplots(row,column)  # this plot will have x
rows and y columns
        #if graph_count > 4:
        self.fig1, self.axarr1 = plt.subplots(row,column)  # this plot will have
x rows and y columns
        #if graph_count > 8:
        self.fig2, self.axarr2 = plt.subplots(row,column)  # this plot will have
x rows and y columns
        self.fig3, self.axarr3 = plt.subplots(row,column)  # this plot will have
x rows and y columns
        self.fig4, self.axarr4 = plt.subplots(row,column)  # this plot will have
x rows and y columns
        self.fig5, self.axarr5 = plt.subplots(row,column)  # this plot will have
x rows and y columns



# the nex section is for pyhhtonizing the plotting



        #create output file
        self.pdffile=pdffile=self.MyFileName.replace('csv','pdf')
        self.pdffilepath = self.SetTempDirectory()+'/LCWA_TOTAL_'+pdffile

    def NotCall(self):
        counter = 0
        # we will create a large array of figures and axarr
        self.figarray = []*6
        for h in range(1,graph_count+1,4):
            print (counter)
            self.figarray[counter],self.axarray=plt.subplots(row,column)
            counter = counter +1


    def PlotTestData1(self,k):
        """
    Plots the tests
    x1: date
    y1: download
    y2:upload
    k_spectrum # number of graph we have done
    """

        np.set_printoptions(precision=2)

        #Add Ip address
```



```
        #ax.text(.1,.36,'Average $\mu$ and Standard deviation $\sigma$',weight='
bold',transform=ax.transAxes,fontsize=13)
        #ax.text(.1,.23,r'$\mu_{up}     = $'+str(np.around(np.mean(y2),2))+' '+'
[Mb/s]'+r'   $\sigma_{up} =     $'+str(np.around(np.std(y2),2)),transform=ax.tra
nsAxes,fontsize=12)
        #ax.text(.1,.3,r'$\mu_{down} = $'+str(np.around(np.mean(y1),2))+' '+'[Mb
/s]'+r'   $\sigma_{down} = $'+str(np.around(np.std(y1),2)),transform=ax.transAxe
s,fontsize=12)

        #add legend
        #print(self.legend)
        x1,y0,y1,y2 = self.x1,self.y0,self.y1,self.y2


        msl=3. #markersize
        xpos = .05 #text position
        ypos = 1.02
        ylow = 0. #regular y axis limits
        yhigh = 30.
        ylow1=0. # limits for packet loss
        yhigh1=30.

        yhigh2 = 12.
        yhigh3 = 7.

        bbox=(0.03,.03,1.,0.25)
        print('number',k)


    def PlotTestData(self,k):
        """
    Plots the tests
    x1: date
    y1: download
    y2:upload
    k_spectrum # number of graph we have done
    """
        np.set_printoptions(precision=2)

        #Add Ip address


        #ax.text(.1,.36,'Average $\mu$ and Standard deviation $\sigma$',weight='
bold',transform=ax.transAxes,fontsize=13)
        #ax.text(.1,.23,r'$\mu_{up}     = $'+str(np.around(np.mean(y2),2))+' '+'
[Mb/s]'+r'   $\sigma_{up} =     $'+str(np.around(np.std(y2),2)),transform=ax.tra
nsAxes,fontsize=12)
        #ax.text(.1,.3,r'$\mu_{down} = $'+str(np.around(np.mean(y1),2))+' '+'[Mb
/s]'+r'   $\sigma_{down} = $'+str(np.around(np.std(y1),2)),transform=ax.transAxe
s,fontsize=12)

        #add legend
        #print(self.legend)
        x1,y0,y1,y2 = self.x1,self.y0,self.y1,self.y2
```



```
        msl=3. #markersize
        xpos = .05 #text position
        ypos = 1.02
        ylow = 0. #regular y axis limits
        yhigh = 30.
        ylow1=0. # limits for packet loss, now for latency
        yhigh1=100.

        yhigh2 = 12. #limits for 10 Mbs
        yhigh3 = 7. #limits for 5 Mbs
        yhigh4 = 80. #limits for LC19, double pppoe accounts

        #overwrite default max axis

        yhigh2=yhigh3=24.
        yveryhigh = 70.

        bbox=(0.03,.03,1.,0.25)
        print('number',k)
        if k < 2:
            i=0
            self.axarr[i][k].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr[i][k].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)

            axins2 = inset_axes(self.axarr[i][k],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr[i][k].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')

            axins2.yaxis.label.set_color('red')

            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()



            #self.axarr[i][k].plot_date(x1,y0,'r+',label='\n red packet loss ',m
s=ms1)
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)
            self.axarr[i][k].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirList
[k],weight='bold',transform=self.axarr[i][k].transAxes,fontsize=8)
            self.axarr[i][k].xaxis.set_major_locator(md.MinuteLocator(interval=3
60))
            self.axarr[i][k].xaxis.set_major_formatter(md.DateFormatter('%H:%M'
))
            print(np.around(np.mean(y1),2))
            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr[i][k].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
```



```
12.):
                self.axarr[i][k].set_ylim(ylow,yhigh) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr[i][k].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr[i][k].set_ylim(ylow,yhigh3) # set yaxis limit
                #if(k==0): self.axarr[i][k].set_ylim(ylow,200.) # for mike ault

        elif k >1  and k < 4:

            i=1
            self.axarr[i][k-2].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr[i][k-2].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr[i][k-2],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr[i][k-2].transAxe
s )
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()



            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)
            self.axarr[i][k-2].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLi
st[k],weight='bold',transform=self.axarr[i][k-2].transAxes,fontsize=8)
            self.axarr[i][k-2].xaxis.set_major_locator(md.MinuteLocator(interval
=360))
            self.axarr[i][k-2].xaxis.set_major_formatter(md.DateFormatter('%H:%
M'))
            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr[i][k-2].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarr[i][k-2].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr[i][k-2].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr[i][k-2].set_ylim(ylow,yhigh3) # set yaxis limit

        if k > 3 and k <6:
            i=0
            l=k-4
            self.axarrl[i][l].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarrl[i][l].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarrl[i][l],width="100%", height="100%",
```



```
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarrl[i][l].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)


            self.axarrl[i][l].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarrl[i][l].transAxes,fontsize=8)
            self.axarrl[i][l].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarrl[i][l].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))

            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarrl[i][l].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarrl[i][l].set_ylim(ylow,yhigh) # set yaxis limit




            elif(np.around(np.mean(y1),2) > 12.):
                self.axarrl[i][l].set_ylim(ylow,yhigh) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarrl[i][l].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarrl[i][l].set_ylim(ylow,yhigh3) # set yaxis limit

        elif k >5  and k < 8:

            i=1
            l=k-6
            self.axarrl[i][l].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarrl[i][l].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarrl[i][l],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarrl[i][l].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
```

```
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr1[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr1[i][1].transAxes,fontsize=8)
            self.axarr1[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr1[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))

            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr1[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarr1[i][1].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr1[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr1[i][1].set_ylim(ylow,yhigh3) # set yaxis limit

        if k > 7 and k <10:
            i=0
            l=k-8
            self.axarr2[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr2[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr2[i][1],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr2[i][1].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr2[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr2[i][1].transAxes,fontsize=8)
```



```
            self.axarr2[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr2[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))

            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr2[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarr2[i][1].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr2[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr2[i][1].set_ylim(ylow,yhigh3) # set yaxis limit

        elif k > 9  and k < 12:

            i=1
            l=k-10
            self.axarr2[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr2[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr2[i][1],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr2[i][1].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr2[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr2[i][1].transAxes,fontsize=8)
            self.axarr2[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr2[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))
```



```
            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr2[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarr2[i][1].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr2[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr2[i][1].set_ylim(ylow,yhigh3) # set yaxis limit


        if k > 11 and k <14:
            i=0
            l=k-12
            self.axarr3[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr3[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr3[i][1],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr3[i][1].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr3[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr3[i][1].transAxes,fontsize=8)
            self.axarr3[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr3[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))

            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr3[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarr3[i][1].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr3[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr3[i][1].set_ylim(ylow,yhigh3) # set yaxis limit
```



```
                self.axarr3[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr3[i][1].set_ylim(ylow,yhigh3) # set yaxis limit

        elif k > 13  and k < 16:

            i=1
            l=k-14
            self.axarr3[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr3[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr3[i][1],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr3[i][1].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr3[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr3[i][1].transAxes,fontsize=8)
            self.axarr3[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr3[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))

            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr3[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarr3[i][1].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr3[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr3[i][1].set_ylim(ylow,yhigh3) # set yaxis limit

############################
        if k > 15 and k <18:
            i=0
            l=k-16
            #self.axarr4[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1
)
            #self.axarr4[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr4[i][1],width="100%", height="100%",
```



```
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr4[i][1].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr4[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr4[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)



            self.axarr4[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr4[i][1].transAxes,fontsize=8)
            self.axarr4[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr4[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))

            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr4[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarr4[i][1].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr4[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr4[i][1].set_ylim(ylow,yhigh3) # set yaxis limit

        elif k > 17  and k < 20:

            i=1
            l=k-18
            self.axarr4[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr4[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr4[i][1],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr4[i][1].transAxes
)
            axins2.get_xaxis().set_visible(False)
```



```
            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr4[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr4[i][1].transAxes,fontsize=8)
            self.axarr4[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr4[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))

            if(np.around(np.mean(y1),2) >25.):
                # set yaxis limit
                self.axarr4[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 25. and np.around(np.mean(y1),2) >
12.):
                self.axarr4[i][1].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr4[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr4[i][1].set_ylim(ylow,yhigh3) # set yaxis limit
            if(k ==19):
                self.axarr4[i][1].set_ylim(ylow,yhigh4)
                #self.axarr5[i][1].set_ylim(0.,1000.)

##########
        if k > 19 and k <22:
            i=0
            l=k-20
            #self.axarr5[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1
)
            #self.axarr5[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr5[i][1],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr5[i][1].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()
```



```
            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr5[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr5[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)



            self.axarr5[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr5[i][1].transAxes,fontsize=8)
            self.axarr5[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr5[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
'))
            if(np.around(np.mean(y1),2) >30.):
                # set yaxis limit
                self.axarr5[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 30. and np.around(np.mean(y1),2) >
12.):
                self.axarr5[i][1].set_ylim(ylow,yhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr5[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr5[i][1].set_ylim(ylow,yhigh3) # set yaxis limit

        elif k > 21  and k < 24:

            i=1
            l=k-22
            self.axarr5[i][1].plot_date(x1,y1,'bs',label='\n blue DOWN ',ms=ms1)
            self.axarr5[i][1].plot_date(x1,y2,'g^',label='\n green UP ',ms=ms1)
            axins2 = inset_axes(self.axarr5[i][1],width="100%", height="100%",
              #bbox_to_anchor=(0,0,1.,.4)  )
              bbox_to_anchor=bbox   , bbox_transform=self.axarr5[i][1].transAxes
)
            axins2.get_xaxis().set_visible(False)

            axins2.spines['bottom'].set_color('red')
            axins2.spines['top'].set_color('red')
            axins2.yaxis.label.set_color('red')
            axins2.tick_params(axis='y', colors='red')
            axins2.set_ylim(ylow1,yhigh1)
            axins2.yaxis.set_label_position("right")
            axins2.yaxis.tick_right()

            axins2.yaxis.label.set_color('red')
            axins2.plot_date(x1,y0,'r+',label='\n red packet loss ',ms=ms1)

            self.axarr5[i][1].text(xpos,ypos,'MyIP = '+self.MyIP+'  '+self.DirLis
t[k],weight='bold',transform=self.axarr5[i][1].transAxes,fontsize=8)
            self.axarr5[i][1].xaxis.set_major_locator(md.MinuteLocator(interval=
360))
            self.axarr5[i][1].xaxis.set_major_formatter(md.DateFormatter('%H:%M
```



```
'))
            if(np.around(np.mean(y1),2) >30.):
                # set yaxis limit
                self.axarr5[i][1].set_ylim(ylow,yveryhigh) # set yaxis limit

            elif(np.around(np.mean(y1),2) <= 30. and np.around(np.mean(y1),2) >
12.):
                self.axarr5[i][1].set_ylim(ylow,yhigh) # set yaxis limit



            elif(np.around(np.mean(y1),2) <= 12. and np.around(np.mean(y1),2) >
7.):
                self.axarr5[i][1].set_ylim(ylow,yhigh2) # set yaxis limit
            elif(np.around(np.mean(y1),2) <= 7. ):
                self.axarr5[i][1].set_ylim(ylow,yhigh3) # set yaxis limit



        #plt.show()  #Uncomment for seeing the plot
    def PushFileDropbox(self):
        self.pdf = self.SetTempDirectory()+'/LCWA_TOTAL_'+self.pdffile
        f =open(self.pdf,"rb")
        dropdir ='/LCWA/ALL_LCWA/'
        self.dbx.files_upload(f.read(),dropdir+'LCWA_TOTAL_'+self.pdffile,mode
=dropbox.files.WriteMode('overwrite', None))

    def ReadTextFile(self,file):
        """ reads text information file"""



        with open(file, 'r') as f1:
            s = f1.readlines()
            IP = s[0].split()
            try:
                self.MyIP = IP[1]
            except:
                self.MyIP = '999.99.99.1'




if __name__ == '__main__':

    #create the list
    temp = 'LC'
    dirlist = []
    for k in range(1,21+1):
        if (k<10):
            templ = temp+'0'+str(k)+'_'
```



```
        else:
            templ = temp+str(k)+'_'

        dirlist.append(templ)
    token_file = '/git/LCWA/src/LCWA_d.txt'
    tempdir = 'scratch'
    datefile = '2022-01-18'
    # " default is none"
    PA=PlotAll(token_file,dirlist,datefile)
    PA.ConnectDropbox()
    PA.GetFiles()
    PA.PushFileDropbox()
```