Storing Traffic Data

In this lab, you will create an application that gathers traffic data from an interface on a device in the network. Your application will poll for this traffic data a specific number of times, at a specific interval. When complete, it will print LAN utilization statistics for the chosen interface.

Your application will:

- Receive interval and count information from the command line, as well as the target device and interface.
- Read device info from the database
- Connect to device and gather the appropriate number of traffic statistics readings from the device
- Print the results when complete

Storing Traffic Data

Step 1

Allow the user to enter command-line arguments to set the interval, count, device IP address, and interface name `gigabitethernet 0/1`. The command line should look like:

```
python main.py [interval] [count] [device IP] [interface]
```

There should be default values for each variable. In Python, command-line arguments are accessed using the variable `argv`, where `argv[0]` is the application filename (for example, 'main.py'), `argv[1]` is the first parameter, `argv[2]` is the second, and so on. You will need to import `argv` from `sys`.

Answer

The `argv` list, available once you import it from `sys`, is a list with each argument presented to the script on the command line.

```
from sys import argv

#=====================================================================
# Main program: read devices, then get traffic statistics from each

# Get arguments passed to our application
print 'argv: ',argv
interval  = int(argv[1]) if len(argv) >= 2 else 5
count     = int(argv[2]) if len(argv) >= 3 else 5
device_ip = argv[3]      if len(argv) >= 4 else '10.30.30.1'
interface = argv[4]      if len(argv) >= 5 else 'gigabitethernet 0/1'
```

Step 2

Read in your device information from the database `devices.db` located in the **~/Desktop/PRNE/section15/db** folder.

Answer

```
#============================================================================
def read_devices_db(devices_db_file):

    print '---- reading devices from db ----------------------------'

    # Connect to the database and get a connection
    db_connection = sqlite3.connect(devices_db_file) # DB connection
    db_cursor = db_connection.cursor()               # DB cursor

    db_cursor.execute('SELECT * FROM devices')
    devices_from_db = db_cursor.fetchall()

    pprint(devices_from_db)
    print '---- end devices from db --------------------------------'

    # Done adding devices to the table, close it down.
    db_cursor.close()
    db_connection.close()

    return devices_from_db
```

Step 3

Find the device access information (username and password) for the target device from the information read from the database.

Answer

```
#============================================================================
def get_device(devices_from_db, device_ip):

    for device_info in devices_from_db:

        # Create a device object with this data
        if device_info[1] == device_ip and device_info[2] == 'ios':

            return  NetworkDeviceIOS(device_info[0],device_info[1],
                                     device_info[3],device_info[4])

    return None
```

Step 4

Gather traffic data from the specified interface for the target device, for the appropriate interval and count. Write the information to a CSV log file.

Answer

```
#=========================================================================
def gather_traffic_data(logfile, device, interface, interval, count):

    # Reset the log file to begin fresh
    dev_stats_log = open(logfile,'w')
    dev_stats_log.close()

    # Loop for 'num_readings' number of times
    for _ in range(count):  # Loop the specified number of times

        print ''
        print '---- reading rx, tx data ------------------------------'

        dev_stats_log = open(logfile,'a') # open the file for appending

        # Get the rx and tx stats for a specific interface on device
        device.connect()
        stats = device.get_interface_stats(interface)

        print '    ---- ',device.ip_address,' : stats: ',stats

        # Write rx and tx information to database
        write_stats_log(dev_stats_log, time(), device.ip_address,
                            stats[0], stats[1], stats[2], stats[3])

        dev_stats_log.close()  # Done appending to file for now

        print ''
        sleep(interval)  # Pause the defined interval
```

Step 5

At each iteration, write the log entries to the log file.

Answer

```
#=============================================================================
def write_stats_log(file, time, ip_address,
                                rx_packets, rx_bytes,
                                tx_packets, tx_bytes):

     # Create a log entry from the data given
     log_entry = [time, ip_address, rx_packets, rx_bytes,
                                    tx_packets, tx_bytes]

     log = csv.writer(file)
     log.writerow(log_entry)
```

Step 6

After the gathering interval has completed, read in the log file, and calculate and print the LAN utilization for each interval.

```
#====================================================================
def print_log(device_ip_address, log_info_list):

    print '  Time        Rx Packets    Rx Bytes  Tx Packets    Tx Bytes     U
    print '  ----------  ----------  ----------  ----------  ----------  ----

    last_bytes = 0

    for log_entry in log_info_list:

        if log_entry[1] != device_ip_address: # Ignore if not passed IP addr
            continue

        if last_bytes == 0:    # if first calculation, set util to 0
            util = 0
            last_bytes = int(log_entry[3]) + int(log_entry[5])
            last_secs = float(log_entry[0])

        else:                         # else calculate lan utilization
            current_bytes = int(log_entry[3]) + int(log_entry[5])
            interval_bytes = current_bytes - last_bytes

            if_speed = 1000000000
            current_secs = float(log_entry[0])
            interval_time = current_secs - last_secs

            util = (interval_bytes*8*100) / float(interval_time*if_speed)

            last_bytes = current_bytes
            last_secs  = current_secs

        str_time = strftime('%H:%M:%S', localtime(float(log_entry[0])))
        print '  {0:10}  {1:>10}  {2:>10}  {3:>10}  {4:>10}  {5:>7.3f}'.form
                                                  log_entry[2], log_entr
                                                  log_entry[4], log_entr
                                                  util)
```

Step 7

Run your application and verify the output.

Answer

Your output should look similar to:

```
$ python main.py 2 5
argv:  ['main.py', '2', '5']
---- reading devices from db ----------------------------
[(u'ios-01', u'10.30.30.1', u'ios', u'cisco', u'cisco'),
 (u'ios-02', u'10.30.30.2', u'ios', u'cisco', u'cisco'),
 (u'ios-03', u'10.30.30.3', u'ios', u'cisco', u'cisco')]
---- end devices from db ------------------------------

---- reading rx, tx data ------------------------------
--- connecting IOS: telnet 10.30.30.1
    ----  10.30.30.1  : stats:  ('88429', '84210375', '91849', '85053465')


---- reading rx, tx data ------------------------------
--- connecting IOS: telnet 10.30.30.1
    ----  10.30.30.1  : stats:  ('88575', '84325202', '92015', '85170709')


---- reading rx, tx data ------------------------------
--- connecting IOS: telnet 10.30.30.1
    ----  10.30.30.1  : stats:  ('88713', '84427781', '92173', '85275699')


---- reading rx, tx data ------------------------------
--- connecting IOS: telnet 10.30.30.1
    ----  10.30.30.1  : stats:  ('88849', '84530234', '92330', '85380629')


---- reading rx, tx data ------------------------------
--- connecting IOS: telnet 10.30.30.1
    ----  10.30.30.1  : stats:  ('88989', '84632927', '92487', '85485559')
```

```
Device:  10.30.30.1   Interface:  gigabitethernet 0/1

  Time        Rx Packets    Rx Bytes  Tx Packets    Tx Bytes     Util
  ----------  ----------  ----------  ----------  ----------  -------
  11:11:39         88429    84210375       91849    85053465    0.000
  11:11:42         88575    84325202       92015    85170709    0.069
  11:11:45         88713    84427781       92173    85275699    0.061
  11:11:48         88849    84530234       92330    85380629    0.060
  11:11:50         88989    84632927       92487    85485559    0.060
```

You complete application utility library should look similar to:

```
#file: util.py

import csv
from pprint import pprint

import sqlite3

from time import strftime
from time import localtime
from time import sleep
from time import time

from devclass import NetworkDevice
from devclass import NetworkDeviceIOS
```

```
#=====================================================================
def read_devices_db(devices_db_file):

    print '---- reading devices from db ---------------------------'

    # Connect to the database and get a connection
    db_connection = sqlite3.connect(devices_db_file) # DB connection
    db_cursor = db_connection.cursor()               # DB cursor

    db_cursor.execute('SELECT * FROM devices')
    devices_from_db = db_cursor.fetchall()

    pprint(devices_from_db)
    print '---- end devices from db -------------------------------'

    # Done adding devices to the table, close it down.
    db_cursor.close()
    db_connection.close()

    return devices_from_db
```

```
#=====================================================================
def get_device(devices_from_db, device_ip):

    for device_info in devices_from_db:

        # Create a device object with this data
        if device_info[1] == device_ip and device_info[2] == 'ios':

            return  NetworkDeviceIOS(device_info[0],device_info[1],
                                     device_info[3],device_info[4])

    return None
```

```
#=============================================================================
def gather_traffic_data(logfile, device, interface, interval, count):

    # Reset the log file to begin fresh
    dev_stats_log = open(logfile,'w')
    dev_stats_log.close()

    # Loop for 'num_readings' number of times
    for _ in range(count):  # Loop the specified number of times

        print ''
        print '---- reading rx, tx data ----------------------------'

        dev_stats_log = open(logfile,'a') # open the file for appending

        # Get the rx and tx stats for a specific interface on device
        device.connect()
        stats = device.get_interface_stats(interface)

        print '    ---- ',device.ip_address,' : stats: ',stats

        # Write rx and tx information to database
        write_stats_log(dev_stats_log, time(), device.ip_address,
                            stats[0], stats[1], stats[2], stats[3])

        dev_stats_log.close()  # Done appending to file for now

        print ''
        sleep(interval)  # Pause the defined interval
```

```
#=============================================================================
def write_stats_log(file, time, ip_address,
                                rx_packets, rx_bytes,
                                tx_packets, tx_bytes):

     # Create a log entry from the data given
     log_entry = [time, ip_address, rx_packets, rx_bytes,
                                    tx_packets, tx_bytes]

     log = csv.writer(file)
     log.writerow(log_entry)
```

```
#=========================================================================
def print_log(device_ip_address, log_info_list):

    print '  Time        Rx Packets    Rx Bytes  Tx Packets    Tx Bytes     U
    print '  ----------  ----------  ----------  ----------  ----------  ----

    last_bytes = 0

    for log_entry in log_info_list:

        if log_entry[1] != device_ip_address: # Ignore if not passed IP addr
            continue

        if last_bytes == 0:    # if first calculation, set util to 0
            util = 0
            last_bytes = int(log_entry[3]) + int(log_entry[5])
            last_secs = float(log_entry[0])

        else:                          # else calculate lan utilization
            current_bytes = int(log_entry[3]) + int(log_entry[5])
            interval_bytes = current_bytes - last_bytes

            if_speed = 1000000000
            current_secs = float(log_entry[0])
            interval_time = current_secs - last_secs

            util = (interval_bytes*8*100) / float(interval_time*if_speed)

            last_bytes = current_bytes
            last_secs  = current_secs

        str_time = strftime('%H:%M:%S', localtime(float(log_entry[0])))
        print '  {0:10}  {1:>10}  {2:>10}  {3:>10}  {4:>10}  {5:>7.3f}'.forma
                                                  log_entry[2], log_entr
                                                  log_entry[4], log_entr
                                                  util)
```

You complete application device library should look similar to:

```
#file: devclass.py

import pexpect

#================================================================================
#---- Class to hold information about a generic network device --------
class NetworkDevice():

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        self.name = name
        self.ip_address = ip
        self.username = user
        self.password = pw
        self.os_type = None

    def connect(self):
        self.session = None

    def get_interfaces(self):
        self.interfaces = '--- Base Device, does not know how to get interfa

    def get_interface_stats(self,interface):
        return (0,0,0,0)
```

```
#===============================================================================
#---- Class to hold information about an IOS network device --------
class NetworkDeviceIOS(NetworkDevice):

    #---- Initialization ---------------------------------------
    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)
        self.os_type = 'ios'

    #---- Connect ----------------------------------------------
    def connect(self):
        print '--- connecting IOS: telnet '+self.ip_address

        self.session = pexpect.spawn('telnet '+self.ip_address, timeout=20)
        result = self.session.expect(['Username:', pexpect.TIMEOUT])

        self.session.sendline(self.username)
        result = self.session.expect('Password:')

        # Successfully got password prompt, logging in with password
        self.session.sendline(self.password)
        self.session.expect('>')

    #---- Get Stats --------------------------------------------
    def get_interface_stats(self, interface):

        stats_cmd = 'show interface ' + interface + ' accounting' \
                                      + ' | include IP'

        # Execute the show interface accounting command
        self.session.sendline(stats_cmd)
        result = self.session.expect('>')

        stats_output = (self.session.before).splitlines()

        for stats_line in stats_output:

            stats = stats_line.split()
            if stats[0] == 'IP':  # We only care about 'IP' stats
                return (stats[1],stats[2],stats[3],stats[4])

        print '--- unexpected show interface output'
        return (0,0,0,0)
```

You complete application should look similar to:

file: main.py

```
from util import read_devices_db
from util import get_device
from util import gather_traffic_data
from util import write_stats_log
from util import print_log

from sys import argv
import csv

#======================================================================
# Main program: read devices, then get traffic statistics from each

# Get arguments passed to our application
print 'argv: ',argv
interval  = int(argv[1]) if len(argv) >= 2 else 5
count     = int(argv[2]) if len(argv) >= 3 else 5
device_ip = argv[3]      if len(argv) >= 4 else '10.30.30.1'
interface = argv[4]      if len(argv) >= 5 else 'gigabitethernet 0/1'

# Read device information from database, into list of device info lists
devices_from_db = read_devices_db('devices.db')

# Get device information for our device
device = get_device(devices_from_db, device_ip)
if device == None:
    print '!!! Cannot find device in DB!'
    exit()
```

```
logfile = 'dev-stats-log'  # set output CSV log file

# Gather traffic data for the devices in the list
gather_traffic_data(logfile, device, interface, interval, count)

dev_stats_log = open(logfile,'r')
csv_log = csv.reader(dev_stats_log)

log_info_list = [log_info for log_info in csv_log]

# Print log information for our one device
print ''
print 'Device: ', device.ip_address, '  Interface: ', interface
print ''

print_log(device_ip, log_info_list)
```