# FLOW TECHNICAL GUIDE

Named Data Networking IoT Demo

**by UCLA REMAP**

Rev. Jan 30, 2017 (ZS)

## Table of Contents

## Summary

Flow is an IoT-augmented home entertainment system over NDN that provides a game experience in which a player navigates and interacts with a virtual environment via person tracking (OpenPTrack), mobile webpage and handheld devices such as gyroscopes.

To develop flow, we designed and implemented Named Data Networking of Things (NDN-IoT) framework, a set of libraries that implements naming, trust and bootstrap, discovery, and application level pub/sub in the NDN team's IoTDI '16 paper \cite{iotdi2016}, to facilitate application development in a home IoT environment

This is a technical guide that includes links to the source code, equipment used, installation, usage and troubleshooting instructions for the system.

## Contact Information

Jeff Burke: jburke@remap.ucla.edu

Zoe Sandoval: zoe@remap.ucla.edu

Zhehao Wang: zhehao@remap.ucla.edu

## Source Code and Versioning

The two main software components of FLOW are:

1. **Named Data Networking (NDN) forwarder and libraries**
    - **ndn-cxx**
    - **NFD**
    - **ndn-ccl:** 5 libraries in Python, C++, JS, C# and Java, and needed respectively for each NDN-IoT framework library, explained below)

    **Named Data Networking Internet of Things (NDN-IoT) Framework**
    The NDN IoT framework libraries (Python, C++, JS, and C#) are in Github under "framework" folder: https://github.com/remap/ndn-flow/tree/master/framework.
    An overview of functionalities can be found here: https://github.com/remap/ndn-flow/tree/master/framework#functionalities, and a detailed interface description can be found here: https://github.com/remap/ndn-flow/tree/master/design/docs.
    Each piece's own dependencies, installation guide, and examples can be found in their own folders respectively.
    - **ndn-iot-dot-net (C# library) :** https://github.com/remap/ndn-flow/tree/master/framework/ndn_iot_dot_net
    - **ndn-iot-js (JavaScript library):** https://github.com/remap/ndn-flow/tree/master/framework/ndn_iot_js

- **ndn-iot-cpp (C++ library):** https://github.com/remap/ndn-flow/tree/master/framework/ndn_iot_cpp
- **ndn-iot-python (Python library):** https://github.com/remap/ndn-flow/tree/master/framework/ndn_iot_python

## 2. Application

The *FLOW* application components are described in Github under "application folder": https://github.com/remap/ndn-flow/tree/master/application. Each component's functionalities, required devices, and installation guide can be found in their own folders respectively.

### RFDuino

- **The RFduino code :** https://github.com/remap/ndn-flow/tree/master/application/rfduino

### Rasberry Pi

- **The Raspberry Pi helper for RFduinos:** https://github.com/remap/ndn-flow/tree/master/application/rfduino/rpi_helper
- **The Raspberry Pi controller:** https://github.com/remap/ndn-flow/tree/master/framework/ndn_pi

### Mobile Phone

- **The mobile website:** https://github.com/remap/ndn-flow/tree/master/application/website

### OpenPTrack (OPT)

OpenPTrack can be found on Github: https://github.com/OpenPTrack/open_ptrack. For this installation the **development** branch was used. Specifically commit 55fa9ef0697a335f5279329d4b72c38bec442324 was used in this installation.

### Unity

The installation was tested on Unity version 5.3.2.f1 and running on Mac OS X version 10.11.5. Detailed information can be found on Github: https://github.com/remap/ndn-flow/tree/master/application/unity/WWBlimp

*Dependencies:*

- ndn-cxx, NFD
- PyNDN, IoT framework (Python device bootstrap, C# library)

## Startup Guide

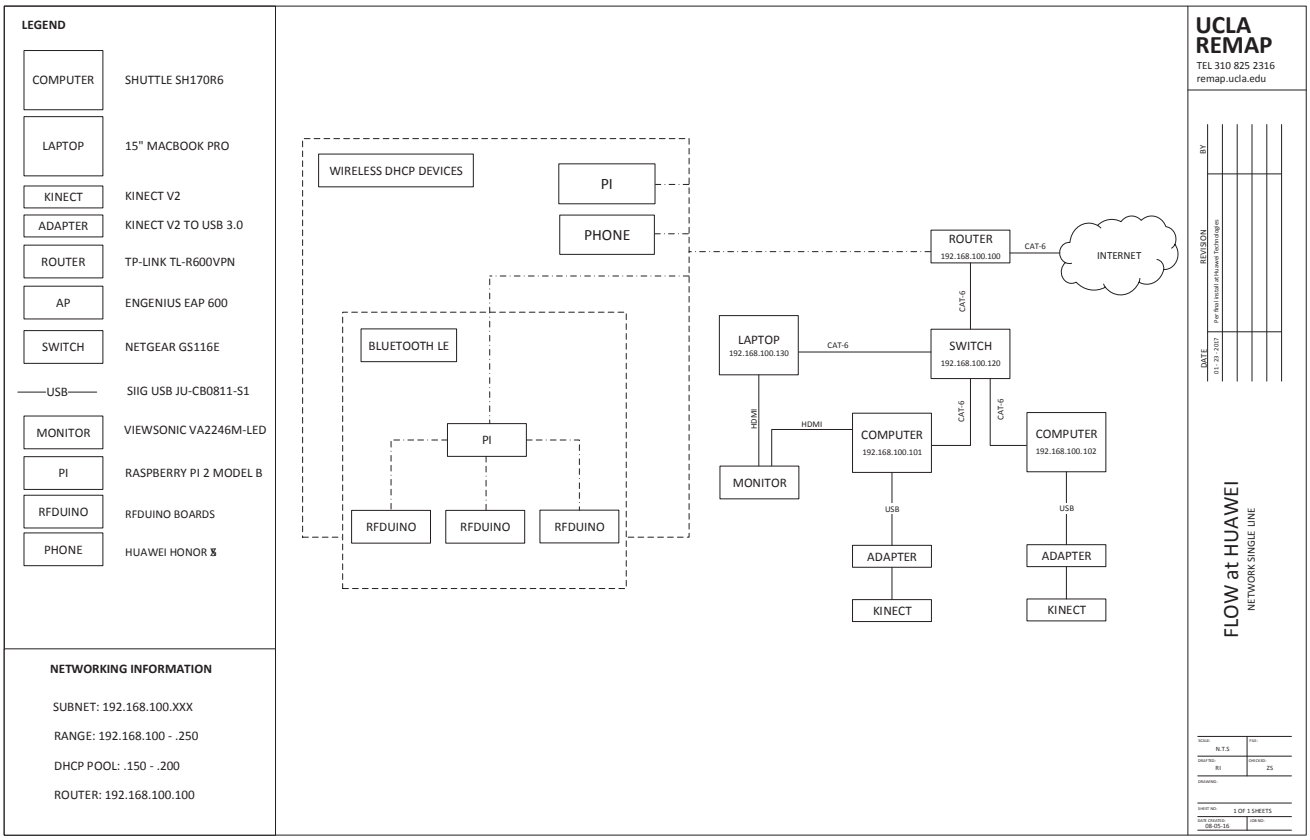
For a Startup Guide please refer to **20170120_OPT_FLOW** on the SHUTTLE .101's desktop and our GitHub: https://github.com/remap/ndn-flow/blob/master/application/start-guide.md

Roughly the process should go as follows:

1. Turn on all machines (Shuttles, Macbook, Pis & Gyros)
2. Check if OpenPTrack is running & publishing NDN tracks on the Shuttle .101
3. If it's not, START TRACKING per **20170120_OPT_FLOW**
4. Then, START NDN PUBLISHER per **20170120_OPT_FLOW**
5. Once you see tracks published, start NFD on the MacBook
6. Next access the RaspberryPi helper, and start NFD
7. Then run RPi helper to get Gyroscopes connected:
    - `python rpi_helper.py --addr e1:16:f4:d2:f6:0f,cd:87:e7:83:90:85 --namespace /home/flow1/gyros/gyro1,/home/flow1/gyros/gyro2`
8. If you would like to add Phones/devices to the application:
    - Start NFD on Gateway
    - `cd ~/ndn/ndn-flow/framework/`
    - `export PYTHONPATH=$PYTHONPATH:$(pwd)`
    - `cd ndn_pi`
    - `python iot_controller.py`
9. If not, just run Unity and click PLAY button
    - Click Scene in the top panel if you want to see the landscape view & not the point of view "game."
10. Launch Firefox on mobile phone and load 192.168.100.101/ndn-flow/application/website
11. After Connecting Face, click "Associate me with a Track"
12. On new page you will be able to click drop images

# Hardware & System Configuration

## System Diagram




## Interaction

The interaction systems are:

1. OpenPTrack (OPT), which is an open source distributed people tracking system. This OPT installation comprises of (2) Shuttle SZ170r6 V2 running NVIDIA GeForce GTX750 Ti graphics cards, and (2) Microsoft Kinect Ones.
2. Sensors and devices (gyroscopes/RFduinos and mobile phones), which are connected wirelessly to the laptop running Unity and control the position and movement of the virtual camera along with “dropping” of images on the landscape, respectively.

## Visualization

The visualization system is Unity a 3D game engine used to create the application for *FLOW.* The engine will run on a 15” Macbook Pro, and the output of the visualization is a monitor. Currently the settings are that HDMI-1 is for OPT and HDMI-2 is for the laptop running Unity.

## Networking

The networking configuration for the system comprises a router, a Netgear 16-port switch, and an EnGenius 600 wireless access point. Components of the system will be connected to the network via wired connections (OPT & Unity) and wireless connections (Sensors & Devices).

### IP Addresses:

| | |
|---|---|
| Router: | 192.168.100.100 |
| Shuttle (Master): | 192.168.100.101 |
| Shuttle (Node): | 192.168.100.102 |
| MacBook: | 192.168.100.130 |
| Raspberry Pi (Gateway): | 192.168.100.140 |
| Raspberry Pi (Helper): | 192.168.100.141 |

### WIFI for Devices:

SSID: opt-flow
Pass: flowdemo2016

## Equipment List

| Quantity | Part Number | Manufacturer |
|---|---|---|
| | **OpenPTrack** | |
| 2 | SH170R6 16gb ram | Shuttle |
| 2 | One Kinect Sensor | Microsoft |
| 2 | Kinect Adapters for Windows | Microsoft |
| 2 | 60’ USB 3.0 Extensions | SIIG |
| 1 | Gigabit Broadband VPN Router | TP-LINK |
| 1 | GS316 Network switch | Netgear |
| 2 | Pavilion HDMI VGA Monitor | HP |
| 1 | M500 Corded Mouse | Logitech |
| 1 | K120 Keyboard | Logitech |
| 2 | Power Strips | Tripp Lite |
| 4 | Cat6 Ethernet Cable | Cable Matters |
| | **FLOW** | |
| 1 | 15" Macbook Pro | Apple |
| 2 | Cat6 Ethernet Cable | Cable Matters |
| 2 | Rasberry Pi 2 - Model B | Rasberry Pi |
| 2 | Bluetooth Dongle | Plugable |
| 1 | 8 GB SD Cards | Sandisk |
| 3 | Rfduino boards | RF Digital Wireless |
| 1 | USB Shield | RF Digital Wireless |
| 3 | AAA Battery Shields | RF Digital Wireless |
| 1 | Alkaline AAA Batteries | Procell |
| 3 | Gyroscopes | SMAKN |
| 1 | Breadboards | Frentayl |
| 1 | 120pc Multicolored wires | Kalevel |
| 1 | Honor 5X | Huawei |

# Installation Guide

## Named Data Networking forwarder

- **NFD installation:** https://github.com/named-data/NFD/blob/master/docs/INSTALL.rst
- **NFD getting started:** https://named-data.net/doc/NFD/current/INSTALL.html

## Sensors & Devices

### RFDuino

- **Setting up the gyroscope on RFduino:** https://github.com/remap/ndn-flow/tree/master/application/rfduino/rfduino-flow-producer#installation

### Raspberry Pi

- **Setting up the Pi and running the gyroscope helper:** https://github.com/remap/ndn-flow/tree/master/application/rfduino/rpi_helper#installation
- **Running the ndn-pi home controller:** https://github.com/remap/ndn-flow/tree/master/framework/ndn_pi

### Mobile Phone

- **Setting up the mobile website:** https://github.com/remap/ndn-flow/tree/master/application/website#installation

## Unity

- **Installing the dependencies:** NDN-IoT (C#) https://github.com/remap/ndn-flow/tree/master/framework/ndn_iot_dot_net#compile
- **Installing Unity:** https://unity3d.com/cn/get-unity/download
- **Downloading and using the WWBlimp Unity project:** detailed information for the installation and its dependencies can be found on GitHub: https://github.com/remap/ndn-flow/tree/master/application/unity/WWBlimp

## OpenPTrack

This section will explain how to calibrate; start detection and tracking; refine calibration; troubleshoot OPT issues. Generally, the calibration should not be necessary, but it will be outlined in this guide. If one is unfamiliar with OPT, a good place to start is the Github Wiki, here[1], which has all of the current installation, calibration, tracking, and troubleshooting information. Additionally, there is a deployment guide as part of the Wiki, here[2], which has photos and additional information on calibration and resolving detection and tracking information. If calibration is not needed and only information on how to start tracking, please proceed to the detection and tracking section.

### Calibration

Calibration should only be performed if calibration refinement cannot resolve track splitting, loss in tracking, or issue related to ground planes. Regarding the process of calibration, much of the

[1] https://github.com/OpenPTrack/open_ptrack/wik
[2] https://github.com/OpenPTrack/open_ptrack/wiki/Deployment-Guide

information can be found on the Wiki, here[3], and in the deployment guide, here[4]. The general calibration process is:

On opt-flow-01, the desktop of the master computer, there is a file named **20170120_OPT_FLOW**. This file contains basic information and the commands for starting and completing calibration. Calibrate the system by:

- Placing the checkerboard on the ground in front of each of the (2) Kinects, allowing ~10 images to be captured
- Then, place the checkerboard in the center of the space with the (0,0) at the center maker.

### Calibration Refinement

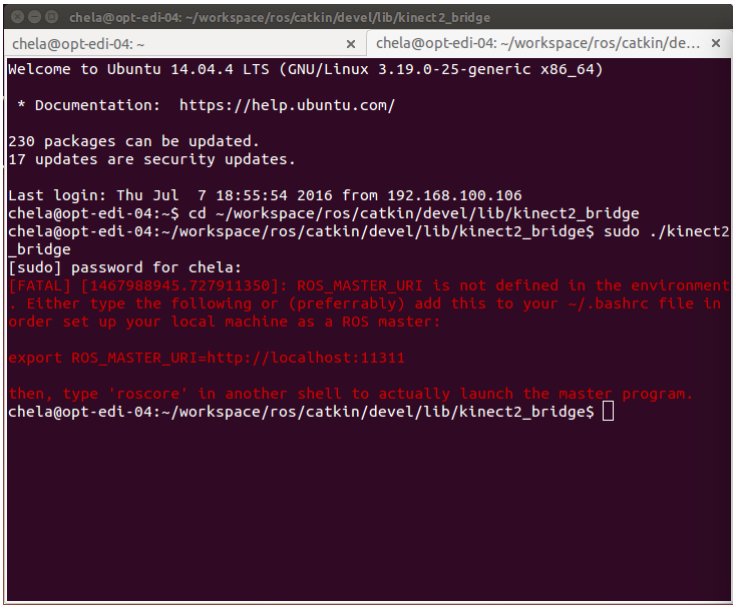
If, after calibration is complete, there is track splitting then calibration refinement can be performed. Details to conducting calibration refinement can be found here[5].

### Detection and Tracking

To start tracking:

1. First turn on the (2) Shuttle computers
2. Verify that all of the Kinect adapters have a white light
   On the desktop of opt-flow-01 (master laptop), there is a file named **20170120_OPT_FLOW**.
   This file contains basic information and the commands for starting detection and tracking.
3. Once the terminal windows are opened, check the NTP offset (this only needs to be completed after a computer has been turned off or restarted).
4. Then proceed with the commands in file **20170120_OPT_FLOW**.
5. NOTE: when running the following command an an error will occur, THIS IS OKAY!
   a. **sudo ~/workspace/ros/catkin/devel/lib/kinect2_bridge/./kinect2_bridge**
   b. This is what the error looks like:

[3] https://github.com/OpenPTrack/open_ptrack/wiki/Camera-Network-Calibration
[4] https://github.com/OpenPTrack/open_ptrack/wiki/Calibration-in-Practice
[5] https://github.com/OpenPTrack/open_ptrack/wiki/Calibration-Refinement

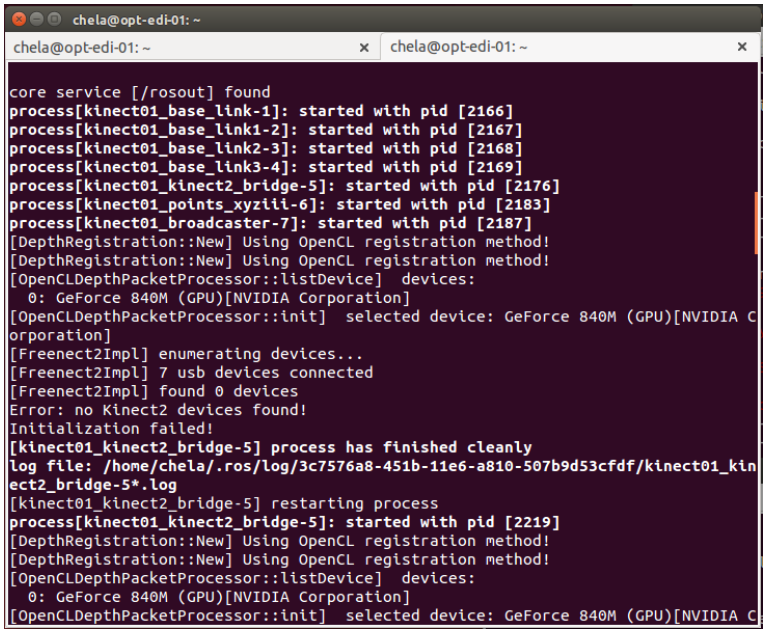



## Troubleshooting (Hardware & Software)

If there are trouble with software there are a few ways to troubleshoot:

## Error: No kinect2 devices found

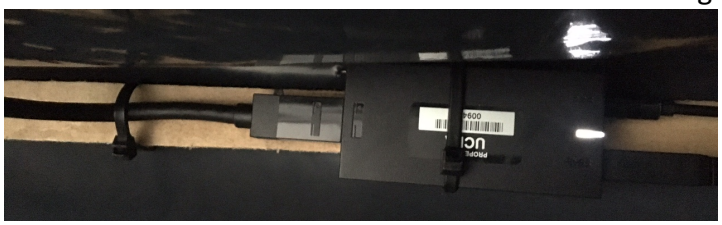
If when starting a Kinect, the terminal prints out “Error: No Kinect2 devices found” there are steps that can be done to resolve this:




1. Verify that the Kinect2 is connected via USB.
    a. If the connect does NOT have power, and orange light will be shown on the USB adapter, which looks like:

i. This means the Kinect is not properly connected to the USB port. Check at the Shuttle computer that the Kinect's USB connector is properly seated, then do the same at the USB adapter.

b. If the USB connection is OK then it will have a white light such as this:



c. If there is no light, then the Kinect does not have power. Check the power sources at the USB adapter and the wall.

2. One can also verify that the Kinect is being identified by the computer by entering the command **lsusb** into a terminal window. If the Kinect is being seen properly by the computer, the terminal print out a line with "Microsoft"

a. There is another way to check as well:

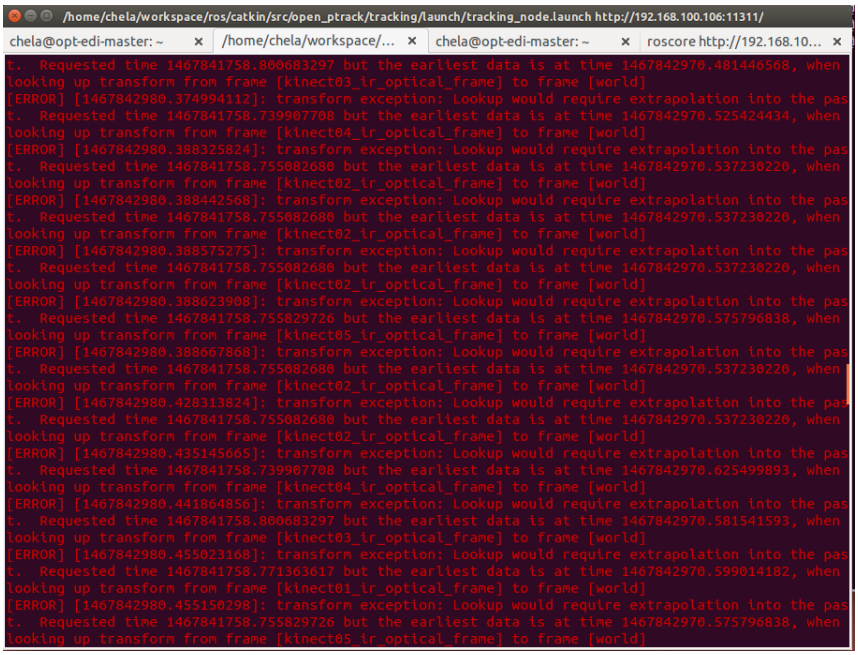
i. First disconnect the Kinect USB from the laptop in question. Then, plug it back in.

ii. In a terminal window quickly enter the command **dmesg**. If the computer is properly recognizing the Kinect, there should be a line that stays "Xbox NUI Sensor", and serial number:

```
chela@opt-edi-01: ~
chela@opt-edi-01: ~ ×  chela@opt-edi-01: ~ ×
m="apparmor_parser"
[  317.112875] vgaarb: this pci device is not a vga device
[  606.019703] nvidia_uvm: Loaded the UVM driver, major device number 249
[  606.090379] kinect2_bridge[2124]: segfault at 0 ip 00007fa526e5b9da sp 00007f
fe979b75d8 error 4 in libc-2.19.so[7fa526dd3000+1ba000]
[ 1276.206008] vgaarb: this pci device is not a vga device
[ 1344.689186] usb 2-1.2: new SuperSpeed USB device number 5 using xhci_hcd
[ 1344.707007] usb 2-1.2: New USB device found, idVendor=045e, idProduct=02d9
[ 1344.707012] usb 2-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber
=0
[ 1344.707014] usb 2-1.2: Product: NuiSensor Adaptor
[ 1344.707016] usb 2-1.2: Manufacturer: Microsoft Corporation
[ 1344.710739] hub 2-1.2:1.0: USB hub found
[ 1344.710980] hub 2-1.2:1.0: 1 port detected
[ 1345.109393] usb 1-2.2: new high-speed USB device number 7 using xhci_hcd
[ 1345.199938] usb 1-2.2: New USB device found, idVendor=045e, idProduct=02d9
[ 1345.199941] usb 1-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber
=0
[ 1345.199943] usb 1-2.2: Product: NuiSensor Adaptor
[ 1345.199944] usb 1-2.2: Manufacturer: Microsoft Corporation
[ 1345.200548] hub 1-2.2:1.0: USB hub found
[ 1345.200779] hub 1-2.2:1.0: 1 port detected
[ 1347.122961] usb 2-1.2.1: new SuperSpeed USB device number 6 using xhci_hcd
[ 1347.140041] usb 2-1.2.1: New USB device found, idVendor=045e, idProduct=02c4
[ 1347.140044] usb 2-1.2.1: New USB device strings: Mfr=1, Product=2, SerialNumb
er=4
[ 1347.140046] usb 2-1.2.1: Product: Xbox NUI Sensor
[ 1347.140047] usb 2-1.2.1: Manufacturer: Microsoft
[ 1347.140048] usb 2-1.2.1: SerialNumber: 103371641047
[ 1347.153967] usbcore: registered new interface driver snd-usb-audio
```

### OpenPTrack is started, but there is no tracking

This can happen from time to time. The first thing to do is verify there are no error in the ./opttrack terminal window. An error will look similar to this:




If this happens, then:

1. Stop ./opttrack
2. Stop **all** the ./optdect functions
3. Restart ./opttrack
4. Restart all of the ./optdect
    a. Rarely, this will not fix the issue and all of the computers in the OpenPTrack network will need to be restarted

### NTP Synchronization

From time to time the NTP offset will be over the +/- 15ms threshold. There are three ways of correcting this problem, to bring the NTP offset into the acceptable tolerance.

1. Wait 5 minutes and check again. Time will generally bring the system into tolerance.
2. On each computer, starting with the master, run sudo service ntp restart
    a. After this command is run on all machines, check **ntpq –p** again to check the offset
3. The third was is to run on all of the OpenPTrack computers, starting with the master:
    a. sudo service ntp stop
    b. sudo ntpd -gq
    c. sudo service ntp start
    d. Then again run **ntpq –p** to check the offset