NASA final project- Team 8

We strongly recommend visiting this link (https://hackmd.io/s/BkC9Jarz-) for better typesetting

Problem description

There are many IoT devices in our daily lifes. Some IoT communicates with other devices. For example, there are monitors in a department store to look out if a thief breaks in. They may exchange information to precisely monitor the movements of the thief. We use this system to make sure devices function and communicate well.

Spec

```
IoT 網路監控
1. Server-client
2. Client
    (a) Deploy in the different environments
    (b) Monitor the network status, and report it to the server
3. Server
(a) GUI to see the report
(b) Alarm on conditions like unreachable or reachable but the quality is poor
(c) Alternative to coding from scratch: integrate with Amazon AWS CloudWatch
    Push metrics to AWS with APIs
```

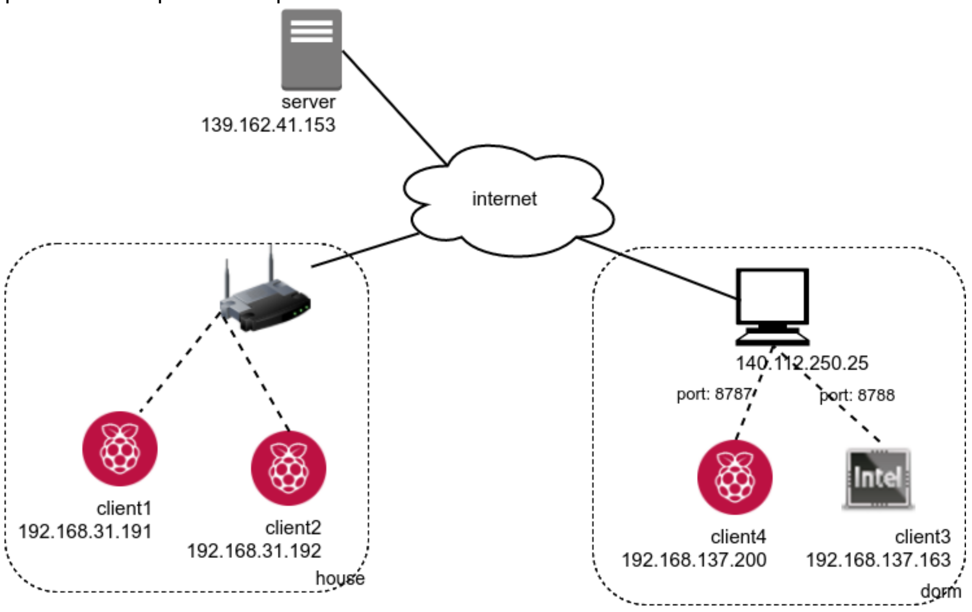
System

Overview

IoT devices are simulated with three **Respberry Pi 3** and one **Intel Edison** as clients. Each two clients are in a wlan. One of the clients can be accessed by clients in other wlan, since some IoT devices may be connected by other remote devices, while some don't.
Server receives metric data from clients and do monitor works. Two systems are deployed, one is linode workstation, the other is AWS cloudwatch. Their functionalities are pretty much the same but with slightly difference.

Network

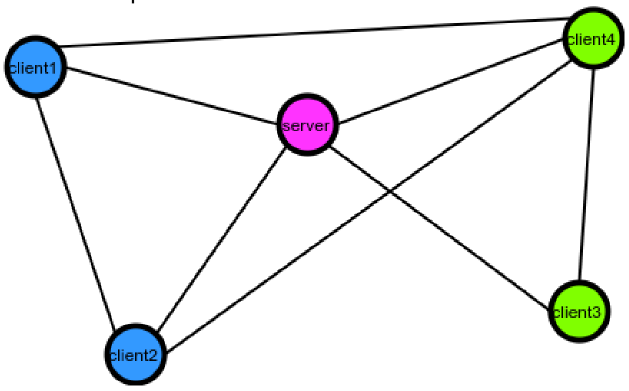
We set up two wlans. The first wlan is set up in house HiNet and wifi as router. The second wlan is set up in the dorm personal computer with public IP as the router.



topology

- client1 and client2 are in same vlan.
- client3 and client4 are in same vlan.
- client1 and client2 can connect to client4 through 140.112.250.25

- server has public IP 139.162.41.153

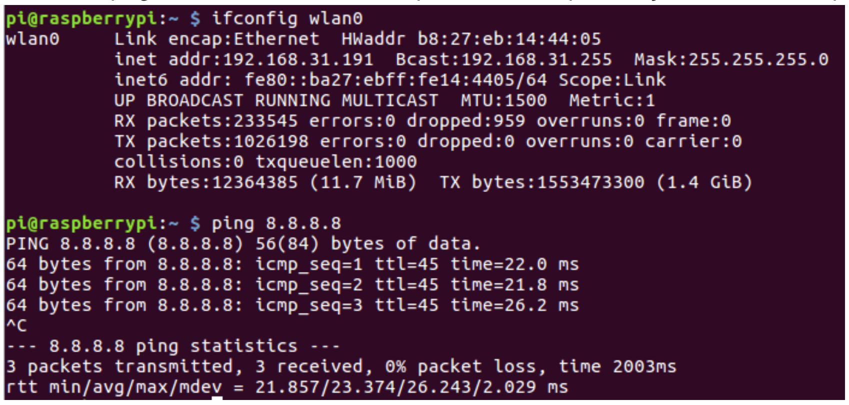


Implementation

Client: Raspberry Pi 3

Setup

- OS: RASPBIAN JESSIE LITE
  - flash image into SD card with etcher (https://etcher.io/)
- Setting Rpi's network on SD card so we can SSH to it, since we don't have monitor and keyboard for Rpi(**headless setup**)
  - set IP, gateway and connect to wifi(Tutorial referenece (http://weworkweplay.com/play/automatically-connect-a-raspberry-pi-to-a-wifi-network/)). Modify 2 files in etc/ of root partition, careful not to modify /etc on your PC
    - etc/network/interfaces
      - static IP
      - netmask
      - gateway
    - etc/wpa_supplicant/wpa_supplicant.conf
      - wifi SSID and password
  - enable SSH, since default SSH is disabled for security concern
    - at **boot/** partion, create empty ssh file
      - ex. `touch ssh`
      - ssh file will be deleted when booted, so we have to add it everytime before we boot Rpi 3
  - PC `ssh pi@staticIP` with default password `rapsberry` , make sure Rpi and PC are in same subnet/wlan



```
pi@raspberrypi:~ $ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr b8:27:eb:14:44:05
          inet addr:192.168.31.191  Bcast:192.168.31.255  Mask:255.255.255.0
          inet6 addr: fe80::ba27:ebff:fe14:4405/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:233545 errors:0 dropped:959 overruns:0 frame:0
          TX packets:1026198 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12364385 (11.7 MiB)  TX bytes:1553473300 (1.4 GiB)

pi@raspberrypi:~ $ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=45 time=22.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=45 time=21.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=45 time=26.2 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 21.857/23.374/26.243/2.029 ms
```

Feature

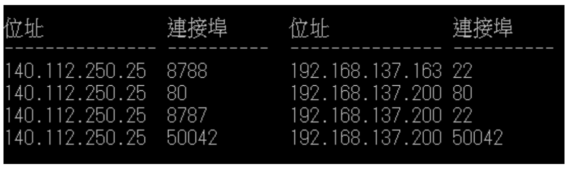
Basic

- Run client.py (https://www.csie.ntu.edu.tw/~b04705006/public/nasa-final/client.py)
  - request other nodes to get network, see topology
    - client1 – client2
    - client1 – client4
    - client1 – server
    - client2 – client4
    - client2 – server
    - client3 – client4
    - client3 – server
    - client4 – server
  - send data to server
    - client4 also sends data to AWS server with AWS API (https://www.csie.ntu.edu.tw/~b04705006/public/nasa-final/client-aws.py)

    - clients send requests on a **1 minute basis**. By our experiment, too frequent request may cause some traffic issue.
- Run th-server.py (https://www.csie.ntu.edu.tw/~b04705006/public/nasa-final/th-server.py)
    - serve as throughput responser

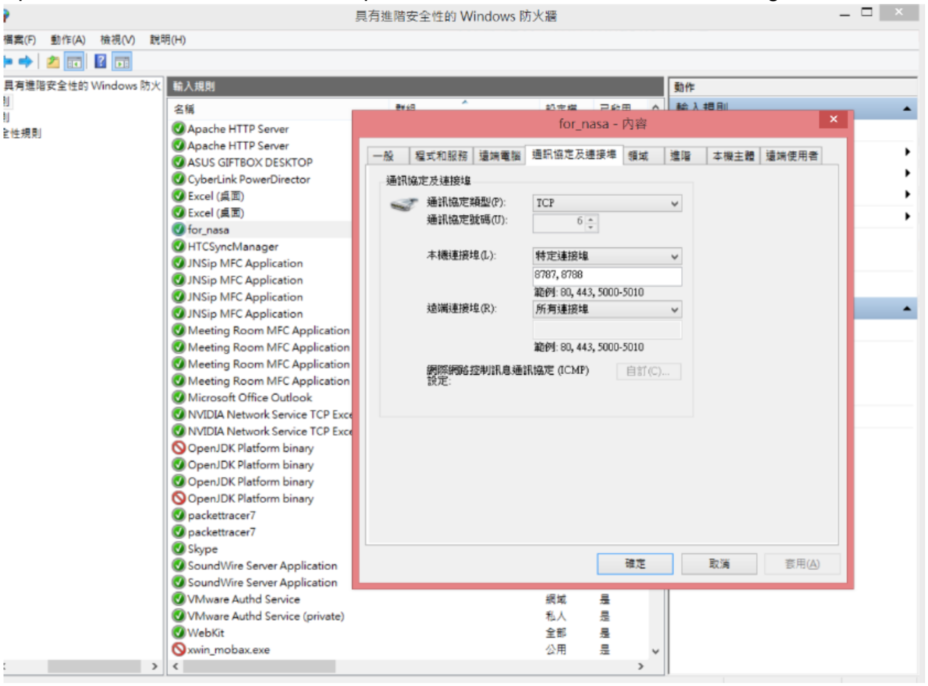
Port forwarding

- Create own wlan
  `netsh wlan start hostednetwork mode=allow ssid=SSID key=PWD`
- (on client4)Host a web server and monitor its CPU Load of Apache server, allow all access of the server-status page
    - `Require all granted`
    - Parse the html page to get the CPU usage of the apache server.
- Allow remote access(ssh) of clients with only private IP
    1. Set up port forwarding with command line
       `netsh interface portproxy add v4tov4 listenaddress=localaddress listenport=localport connectaddress=destaddress connectport=destport`



```
位址            連接埠      位址            連接埠
--------------- ----------  --------------- ----------
140.112.250.25  8788        192.168.137.163 22
140.112.250.25  80          192.168.137.200 80
140.112.250.25  8787        192.168.137.200 22
140.112.250.25  50042       192.168.137.200 50042
```

    2. Create a firewall rule to allow packet to specific port to go through. Also, in order to enable ping, we have to open another rule to enable icmp since Windows defaults to blocking all of them.



具有進階安全性的 Windows 防火牆

for_nasa - 內容

通訊協定及連接埠

通訊協定類型(P): TCP

通訊協定號碼(U): 6

本機連接埠(L): 特定連接埠

8787, 8788

範例: 80, 443, 5000-5010

遠端連接埠(R): 所有連接埠

範例: 80, 443, 5000-5010

網際網路控制訊息通訊協定 (ICMP) 設定:

Metric

- latency
    - ping(count = 5): average rtt
- reachability
    - ping(count = 5): packet loss
- throughput
    - an iperf similar throuhgput.py (https://www.csie.ntu.edu.tw/~b04705006/public/nasa-final/throughput.py). Create socket between client and server, measure sent packets size in 10 second duration.
    - For client1,2 connect to client4, 140.112.250.25:50042 will be forwarded to client4 only
- CPU load (client4 only)
    - Apache web server

We use ping to measure latency and reachability. However, for client1 and client2, ping destination 140.112.250.25 is router PC instead of client4. We use a port forwarding method to forward packet to clients. Currently we don't know how to forward ICMP packets to specific port, so these two metrics between (client1, client4) and (client2, client4) should be ignored.

Server: AWS

Setup

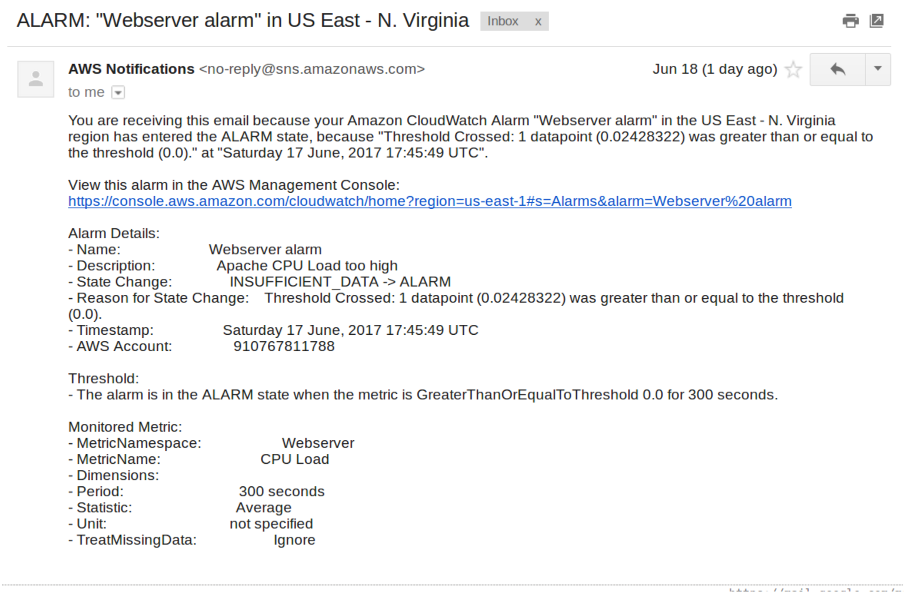
1. Create aws user and configure its according permission with AWS Identity and Access Management (IAM)
2. Install aws CLI and setup as bellow

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

3. Pip install boto3, an Amazon Web Services (AWS) SDK for Python
4. Create custom metrics and upload the data to server in json format

Feature

- Customized UI dashboard
- Alarm that sends email when a certain metric exceed a threshold
  The following image is the demonstration of the alarm that we set up. AWS will send me an email whenever the CPU Usage of the apache server exceeds 0.0003%.



ALARM: "Webserver alarm" in US East - N. Virginia

AWS Notifications <no-reply@sns.amazonaws.com>
to me
Jun 18 (1 day ago)

You are receiving this email because your Amazon CloudWatch Alarm "Webserver alarm" in the US East - N. Virginia region has entered the ALARM state, because "Threshold Crossed: 1 datapoint (0.02428322) was greater than or equal to the threshold (0.0)." at "Saturday 17 June, 2017 17:45:49 UTC".

View this alarm in the AWS Management Console:
https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#s=Alarms&alarm=Webserver%20alarm

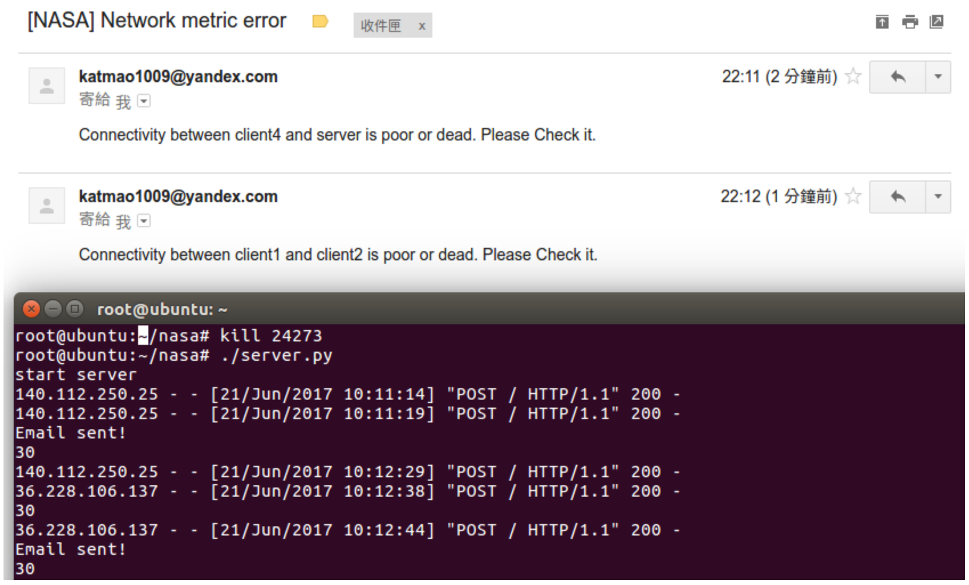
```
Alarm Details:
- Name:                 Webserver alarm
- Description:          Apache CPU Load too high
- State Change:         INSUFFICIENT_DATA -> ALARM
- Reason for State Change:   Threshold Crossed: 1 datapoint (0.02428322) was greater than or equal to the threshold
(0.0).
- Timestamp:            Saturday 17 June, 2017 17:45:49 UTC
- AWS Account:          910767811788

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanOrEqualToThreshold 0.0 for 300 seconds.

Monitored Metric:
- MetricNamespace:          Webserver
- MetricName:               CPU Load
- Dimensions:
- Period:                   300 seconds
- Statistic:                Average
- Unit:                     not specified
- TreatMissingData:         Ignore
```

Server: Linode workstation

This workstation is a linode server, its physical location is in U.S.

Feature

- Run th-server.py (https://www.csie.ntu.edu.tw/~b04705006/public/nasa-final/th-server.py)
  - Serve as throughput responser
- Run server.py (https://www.csie.ntu.edu.tw/~b04705006/public/nasa-final/server.py)
  - A simple http server, collect metric data from client, store data to file. Maintain latest 30 records for each link
  - Alarm when connections between clients is poor or dead



[NASA] Network metric error 收件匣

katmao1009@yandex.com
寄給 我
22:11 (2 分鐘前)

Connectivity between client4 and server is poor or dead. Please Check it.

katmao1009@yandex.com
寄給 我
22:12 (1 分鐘前)

Connectivity between client1 and client2 is poor or dead. Please Check it.

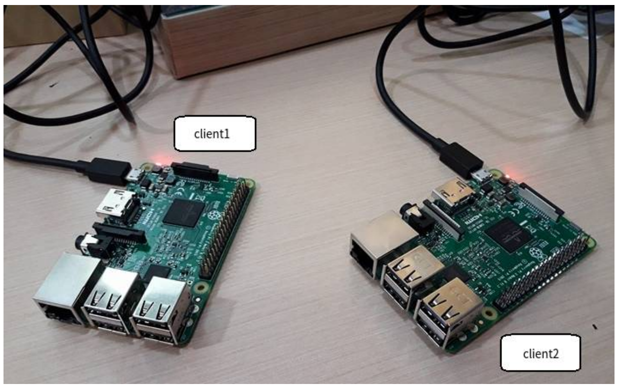
```
root@ubuntu: ~
root@ubuntu:~/nasa# kill 24273
root@ubuntu:~/nasa# ./server.py
start server
140.112.250.25 - - [21/Jun/2017 10:11:14] "POST / HTTP/1.1" 200 -
140.112.250.25 - - [21/Jun/2017 10:11:19] "POST / HTTP/1.1" 200 -
Email sent!
30
140.112.250.25 - - [21/Jun/2017 10:12:29] "POST / HTTP/1.1" 200 -
36.228.106.137 - - [21/Jun/2017 10:12:38] "POST / HTTP/1.1" 200 -
30
36.228.106.137 - - [21/Jun/2017 10:12:44] "POST / HTTP/1.1" 200 -
Email sent!
30
```

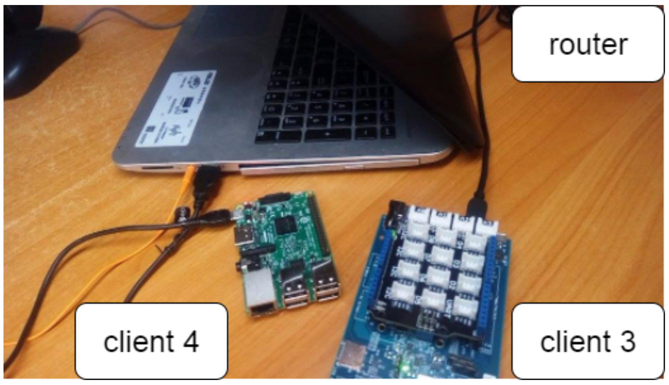
  - Use Yandex (https://mail.yandex.com) mail as smtp server
    - Create an account, eg. katmao1009@yandex.com (mailto:katmao1009@yandex.com)
    - Set **app password**. When using python smtplib to send email, use this password instead of account password

        - Send email to hiwang123@gmail.com (mailto:hiwang123@gmail.com) once when throughput between clients is lower than 10KB/s, including no throughput.
- Read data from file and visualize
    - Interactive UI (http://139.162.41.153/nasa/ui/) with d3.js (https://d3js.org/) to display topology and metric data
    - Update data automaticlly (1 minute basis)

Result

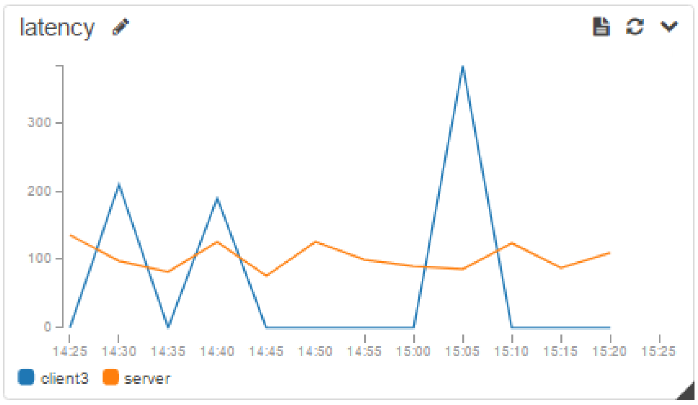
Device

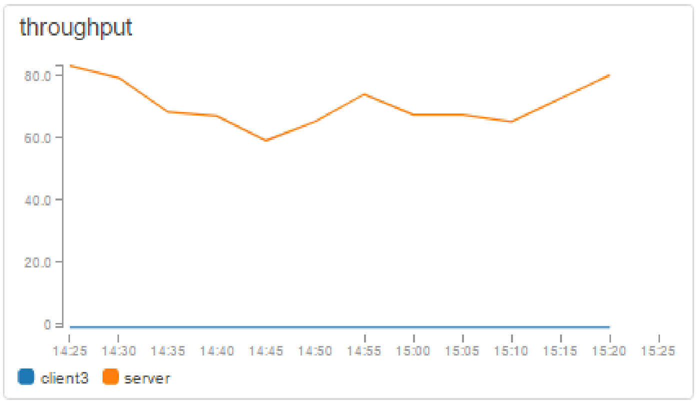


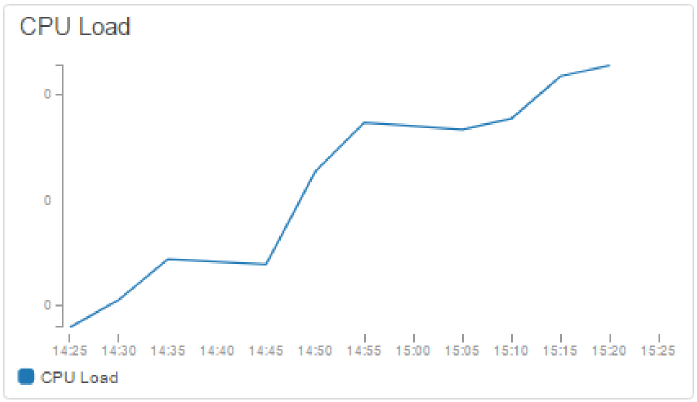


GUI

AWS

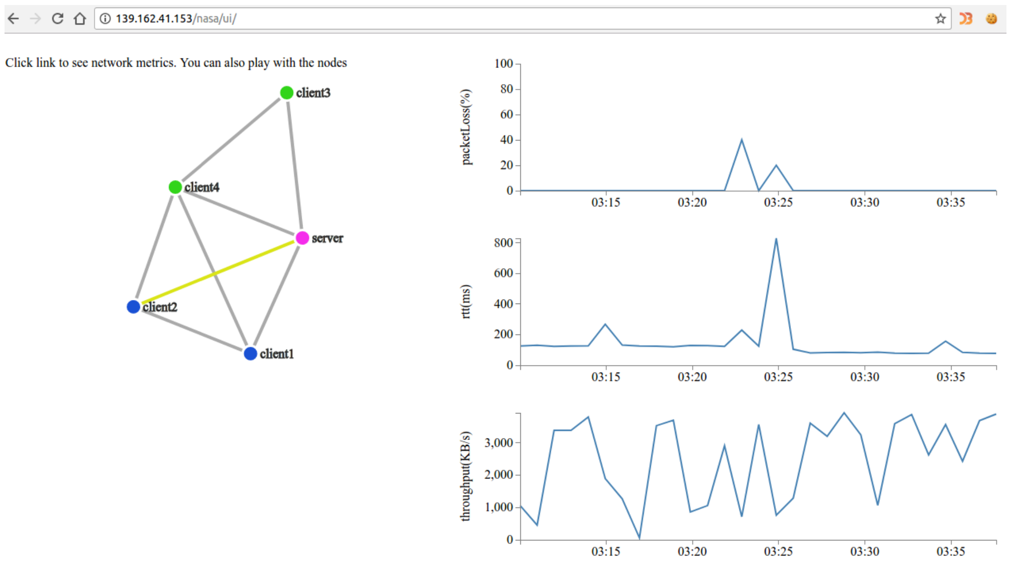






This is data from client4. Server here means 139.162.41.153.

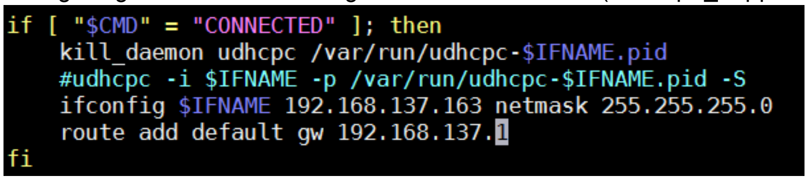
Linode Workstation


Click link to see network metrics. You can also play with the nodes

This is data from client2. It shows network metric between client2 and server(139.162.41.153)

Difficulties and Future Work

- SSH issue
  As of the November 2016 release, Raspbian has the SSH server disabled by default. We have to enable SSH manually.
- Send mail by 3rd party SMTP server
  After trying several mail service(including gmail). Some are not easy to set up due to authentication and security issue, so we decided to use Yandex mail.
- How to monitor throughput between devices that are not under the same subnet?
  Setting up port forwarding on the PC(as router) with public IP.
- How to get the loading of the apache web server dynamically in a programmatic way?
  Parsing the server-status page
- To achieve all-pair monitoring, we set up pretty much the same thing manually for the four clients. In the future, if we need to monitor more IOT devices, we can consider using Ansible to accelerate this procedure.
- In addition to Raspberry PI, we also used Intel Edison. We encountered a problem. That is, we can ssh and ping Edison, but inside Edison, it says Network Unreachable. We found that we forgot to set the gateway when configuring static IP. The configuration for Edison is: (/etc/wpa_supplicant/wpa_cli_actions.sh)



```
if [ "$CMD" = "CONNECTED" ]; then
    kill_daemon udhcpc /var/run/udhcpc-$IFNAME.pid
    #udhcpc -i $IFNAME -p /var/run/udhcpc-$IFNAME.pid -S
    ifconfig $IFNAME 192.168.137.163 netmask 255.255.255.0
    route add default gw 192.168.137.1
fi
```

Cooperation

組員	討論架構	client setup	metric收集與server監控	資料呈現	Report	Presentation
孫凡耘	✓	✓	✓(AWS)	AWS	✓	
王馨儀	✓	✓	✓(linode)	d3.js	✓	
徐嘉琪	✓			d3.js	✓	✓