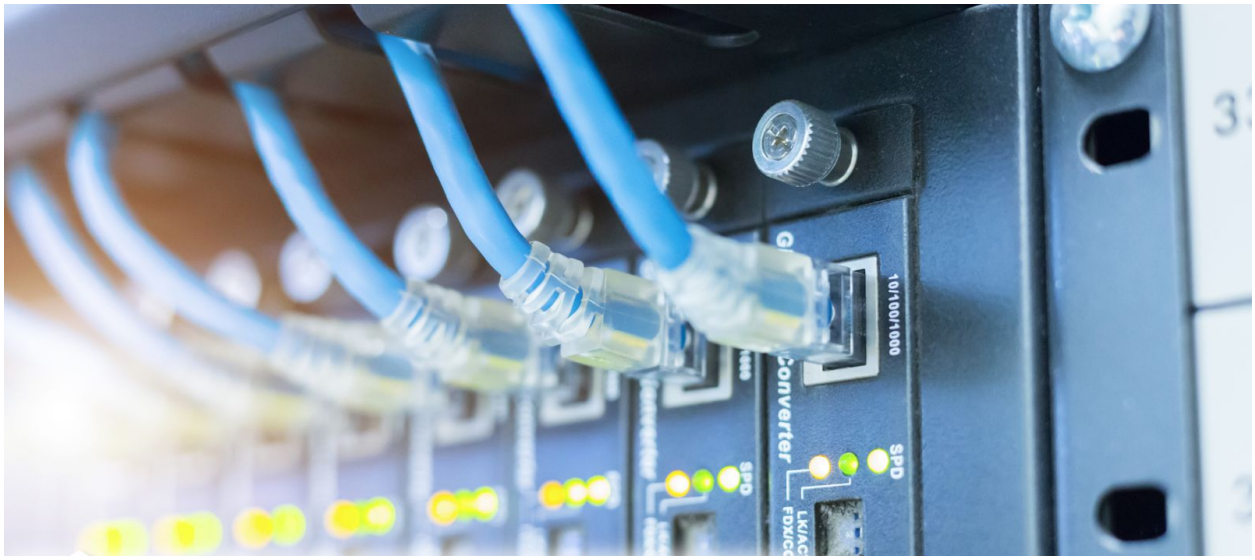


IoT Lab 3 - Infrastructure Networking for IoT

Overview

Please read the complete lab document.

Internet-of-Things devices need some way to communicate with the Internet.
To do this, some sort of infrastructure network must exist to provide coverage to these devices.
Infrastructural networks often consist of:

- a set of wireless access points providing wireless coverage to the IoT devices
- a collection of infrastructural servers providing storage and backend services to the devices, and
- a wired network backhaul enabling communication between devices, services, and the cloud.

Today's IoT devices present significant security and management hurdles to infrastructural network operators. IoT devices are often highly constrained on memory and power and this limits the ability to run more powerful heavyweight security mechanisms on them. The software foundations of many IoT devices are also often immature, having only recently been introduced to the market, making them more prone to errors and vulnerabilities. At the same time, the rapid propagation of these devices on corporate networks means operators are needing to manage end systems on a scale unlike ever before.

This lab will introduce you to the concepts of infrastructure management for IoT devices and also give you an idea of how error-prone it can be. Like mentioned before, many software foundations in the domain of IoT are relatively underdeveloped and hence you won't simply find guides to setting up some of the components in this lab. Our idea is for you to **explore and try as many new things as possible!**

To simplify your work and enable you to focus on learning the core concepts, you will construct this infrastructure in virtual space. In particular, you will leverage an open-source software package called

GNS3. GNS3 is a powerful platform which enables you to run real router images right on your local computer. GNS3 contains a hypervisor for routers, enabling you to construct highly realistic deployments and giving you a very real experience within your computer.

In particular, in this lab, you will be a large retailer similar to Walmart. You will be creating an IoT infrastructure for one of your supercenters. You will deploy a large number of sensors to track and monitor goods in your store, as well as applications such as gaze tracking to monitor shopper behavior and collect datasets to analyze how your product placement is doing. You will deploy a set of access points to communicate with your IoT devices and develop infrastructure to store this data locally in a data warehouse.

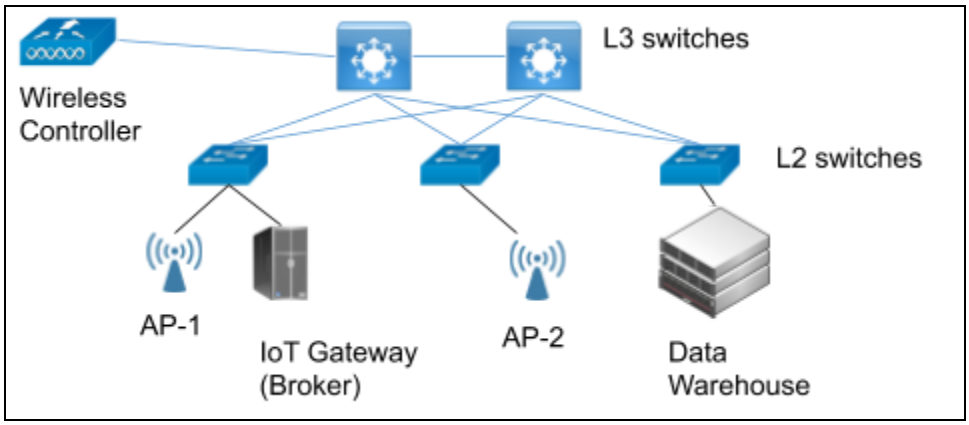
As extra credit, you can also deploy infrastructure with wired backhaul to provide Internet access, authentication, and segmentation for your IoT resources.

Please note that this lab will be a little different from the labs that you have done before. In previous labs, we mentioned specific details of what you should do to achieve the objective. However, in the real world, things usually don't work like that. No one will give you detailed instructions for what you need to do, and you'll have to use your intuition and skills to figure things out. You'll need to learn to use google to search, how to understand and read incomplete documentation of developing software with a critical eye, and you'll have to work your way through it and figure out the right tools to accomplish those tasks. In this lab, you will get practice with that, and develop stronger systems skills to make you more capable of dealing with real-world systems. Some of the steps will tell you what to achieve, but not how to do it. You will need to Google a lot, search newsgroups, post questions online, and exhaust all possible resources. You will need to think about what you need to know, then form plans on how to do it. Since this is a very open-ended lab with a lot of heterogeneity in systems: it may not even be possible for the TAs to end-to-end troubleshoot your system. After you can do this, you will be confident you can handle realistic tasks in the real world.

With that being said, it's important for you to focus on the skills that you'll take away from this lab. It's very important for you to understand the conceptual idea behind the lab and not just follow guides and hacks online to make it work. While you make your way through this lab if you find new software, directions, and suggest meaningful improvements to this lab: you'll be given participation extra credit. Please add your contribution at the end of the document.

Task 1: Deploy the Network

Please perform the following work. Create the network shown in this figure in GNS3:



Clarifications for Topology

1. The mesh-like redundancy in the topology is to add robustness to the system. Multiple links between switches make the system fault tolerant. This is just to give you a better idea of how realistic networks are usually set up. Since the routers or switches are typically on the same rack: there is little or no cost of adding redundant links.

2. We don't need L3 switches in this topology. However, having both L3 and L2 switches in the topology give you an idea of how layered architecture works. The networks usually have different layers of core, access, and distribution switches. What is the difference between an L2 and an L3 switch?

3. We have multiple APs in the topology to show you how you manage all of them using one wireless controller.

4. To virtualize the "IoT devices" you can use any kind of host (CentOS on GNS3 OR your own VM) and have wired connections to the AP. This simulates having a wireless connection.

Your group may find it easier to just build one GNS3 instance for the whole group instead of everyone doing it individually on their laptops. You can also export and share the GNS3 files and VM images with each other to collaborate. Alternatively, you can set up your system in the lab 1129 or on the cloud services (like AWS, Azure, etc). If you decide to set up your system in 1129, please make sure you use one of the unplugged machines in the north half of the room. You can wipe off the hard disk and install your preferred OS. However, please check the RAM to make sure it's enough for your needs.

You'll need to set up a virtualization software on your host. You can use VMware Player for Windows/Linux and/or VirtualBox. There's no VMware Player for Mac but you can use VMware Fusion if you have a subscription.

[Amod] Vagrant works with VirtualBox, so to standardize it for the entire class: I would recommend using VirtualBox.

GNS3 works in two parts:

1. The GUI that allows you to virtualize and deploy various appliances
2. A Linux-based VM where the images are installed and executed

Download the GNS3 application and the GNS3 VM image. Import the GNS3 VM into your virtualization software. Note that you'll have to configure your own host networks and adapters for your host machine to be able to talk to the VM. You should be able to SSH into your VM from your host machine. Please refer to GNS3 documentation for tutorials on how to get started.

In this setup, IoT devices will run beneath the APs. They will use the MQTT protocol to communicate with the IoT Gateway, via the APs. The IoT Gateway will follow the publish/subscribe model and act as an MQTT broker, forwarding the data sent by the devices. This data will be backed up on the Data Warehouse database.

Use VMs or even real IoT devices and connect them to the APs. Enable them to communicate via your APs to IoT Gateway. These devices will be used to keep track of inventory and shopper movement throughout the store. They will be affixed to items you are selling and also shopping carts. Figure out how shoppers are moving, which aisles are not getting traffic, what makes shoppers change their minds about items, etc.

Please use the following software packages for the following items. Please use the open-source versions listed.

1. **Access Point:** OpenWRT (https://docs.gns3.com/appliances/openwrt.html).
    a. An alternative option is to use the Cisco Aironet simulator, but please ask the instructor before using this alternate option.

2. **Wireless Controller:** OpenWISP (http://openwisp.org/whatis.html).
    a. A wireless controller is a central point where you can manage your wireless network. Instead of manually configuring each access point, you log into the controller, put in your configuration once, and then the wireless controller is responsible for operating the APs.
    b. Note: OpenWISP provides a *vagrantfile* to simplify the installation process (works with VirtualBox): https://github.com/openwisp/vagrant-openwisp2.
    c. Refer to troubleshooting at the end of the document
    d. Alternate option (requires instructor approval): Cisco vWLC. (http://netengu.blogspot.com/2012/12/use-cisco-vwlc-in-vmware-workstation.html).

3. **IoT Gateway:** Mosquitto Broker: http://mosquitto.org/
    a. Follow the guide to set up publisher and broker here: https://software.intel.com/en-us/node/734997
    b. Alternative: Intel IoT Gateway/Intel IoT Solutions (https://shopiotmarketplace.com/iot/index.html#/details?pix=58).

4. **Switches:** GNS3 switch
    a. Use any switch, read about different kinds from the documentation

5. **Data Warehouse:** PostgreSQL, mySQL etc...
    a. Look at the functionalities Docker image may provide
    b. Any setup is fine, no restrictions. e.g.,
        i. MySQL + Ubuntu
        ii. Oracle SQL server + Alpine
    c. We can also allow the use of data sinks rather than databases

Configure APs and switches appropriately to ensure reachability. Outside of these requirements, please configure policies according to good security and resilience practices.

1. Wireless devices connecting at the access points can reach the IoT Gateway.
2. The IoT Gateway can reach the data warehouse.

For some tips on importing VMs into GNS3 or troubleshooting network setups, please check the bottom of the lab doc.
If you find yourself totally lost and confused about the networking aspect of this lab, Here is a good intro to networking basics.

Task 2: Deploy the Application

Set up Intel IoT Gateway/Solution following the guide. Run the Gaze Tracking application. This application will be used to track where shoppers in your store are looking. You will figure out if they are looking at the shelves with the Cheerios or the Life cereal. This will help you figure out where to put products, what is distracting, how to market to children, etc.

You don't need to implement this application, you can just download it (from here: https://software.intel.com/en-us/iot/reference-implementations/shopper-gaze-monitor) and run it.

Also, set up the **Gateway** to act as an MQTT broker to store MQTT messages.

1. Setup a **64-bit 16.04 Ubuntu VM**. You can download the OS image here: http://releases.ubuntu.com/16.04/. Since the dependencies and the project itself takes quite an amount of space, make sure you at least allocate **10GB of storage** so you don't run out of storage. A generous **15GB** would be ideal if your laptop still has a lot of storage.

2. Follow the instructions provided on this page: https://software.intel.com/en-us/articles/OpenVINO-Install-Linux to setup OpenVINO toolkit in the VM. It is a required dependency before we can deploy our shopper gaze monitor. Jump right to the section "Installation Steps", and follow through the guide and stop before "optional steps". This concludes our setup for the OpenVINO toolkits.

*Note: To download the toolkit package file, they will ask you to register for an account and fill in the company name, just put random strings for that field as it doesn't matter, make sure the email is valid so they can send you the download link though.

3. Now, back to the GitHub page: https://github.com/intel-iot-devkit/shopper-gaze-monitor-cpp .
   Git clone the project directory to wherever you like and follow through the instructions from section "Setting the build environment" to the section "Running the code".
   *To make your webcam available in the VM, you will have to select a webcam in the upper left tab (for VirtualBox users) under devices -> webcams -> *Your_webcam_name*.

4. If everything was set up correctly, when you run the code, you should be able to see a pop-up video frame recording with the webcam you selected. If the frame rates are running low, or if your VM is just very laggy, try allocating more video memory for it under settings -> display -> video memory in the VirtualBox Manager.

5. For setting up MQTT, check out this guide: https://software.intel.com/en-us/SetupGateway-MQTT.
   However, be aware that it sets the Intel IoT gateway as the publisher and other computers as MQTT broker and subscribers, which is different from our setup. The MQTT library it uses can be found here: https://github.com/eclipse/paho.mqtt.python
   (However, feel free to use any other library that you're comfortable with).

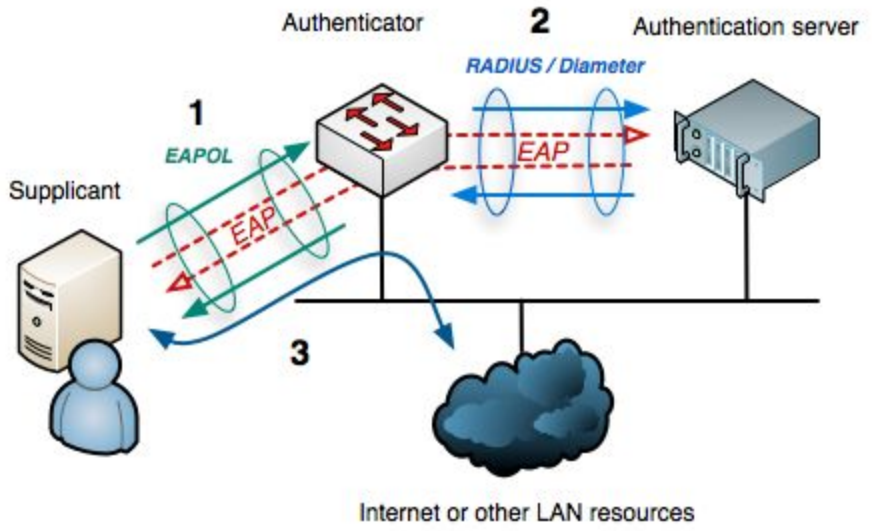
   To learn more about MQTT on Intel IoT architecture, you can watch this video:
   https://software.intel.com/en-us/videos/mqtt-protocol-on-intel-iot-architecture

Task 3: Secure the Network (extra credit)

Your network has a large number of IoT devices. IoT devices are typically resource-constrained and may run immature software. Best practice is to isolate these devices from the rest of your network and perform strong authentication before allowing them in.

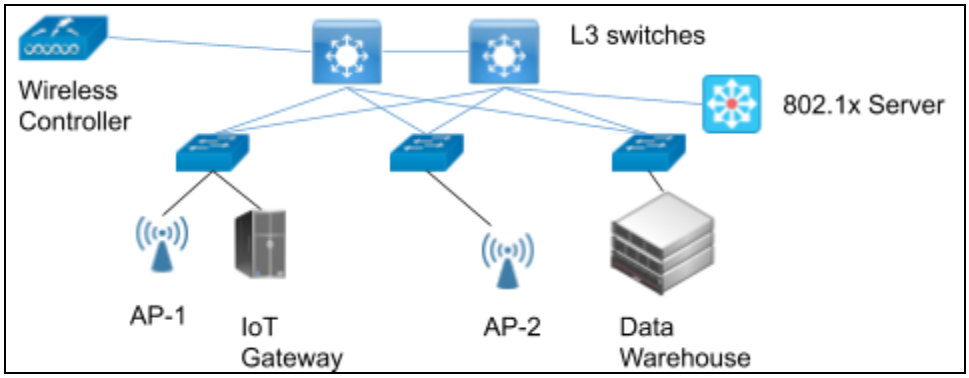
To perform authentication, configure an 802.1x directory server with MAC addressing bypass authentication. MAC address bypass authentication means that the supplicant will share its MAC address with the 802.1X server and based on the rules defined at the server (perhaps, a whitelist of MAC addresses), the supplicant will be authenticated. 802.1X will match the MAC address of the device in the list and allow the device to connect.

To perform isolation, when a new device connects (supplicant), 802.1X is attempted first. If the authentication succeeds then you put the device on a VLAN. Otherwise, you fall back to putting the device it on a restricted VLAN (instead of a regular VLAN). You will use the dynamic VLAN association for this purpose. For the 802.1x authentication server, please use FreeRadius.



In this scenario, your IoT device will be the supplicant. The AP will be the authenticator that will forward your authentication request to the authentication server (RADIUS). The FreeRadius server will authenticate the device and reply with information on VLAN.

Your topology will now look like this:



TAs will only provide you limited help with this part of the lab because it's extra credit.

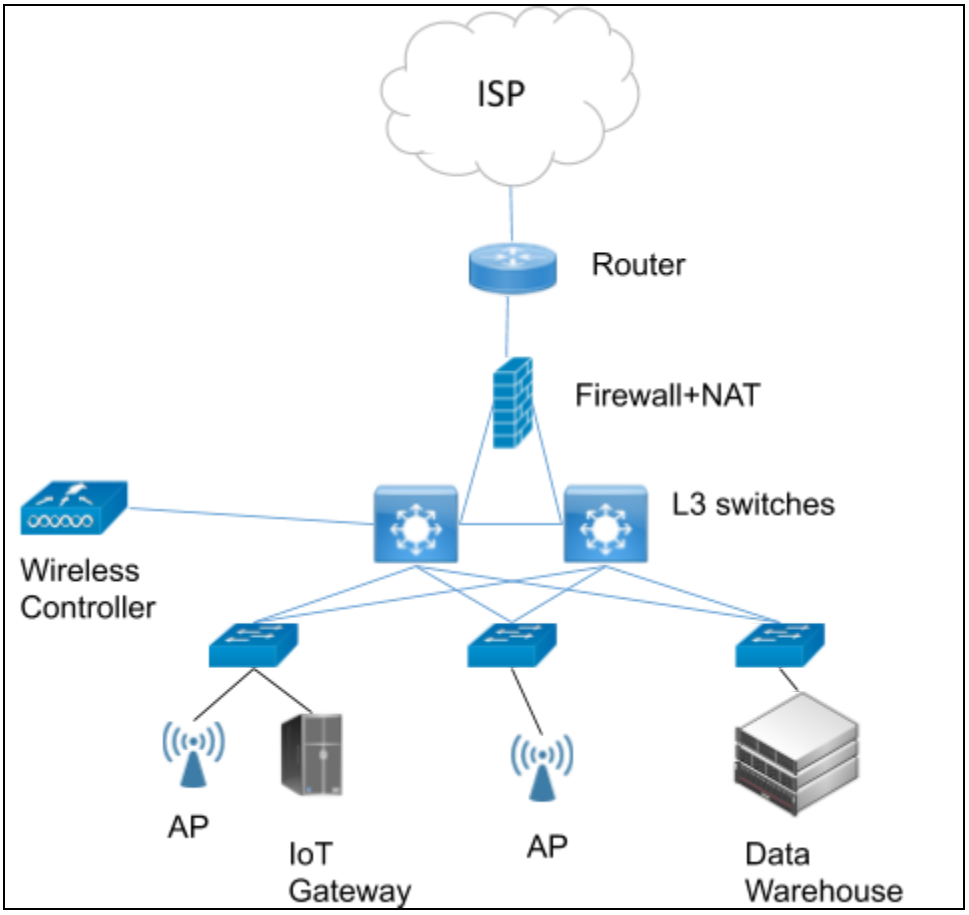
Task 4: Extend the infrastructure to allow WAN connectivity (extra credit)

Your task is now to extend your original topology and its infrastructure by adding Firewall + NAT along with routers to it.

Configure protocols appropriately to ensure following reachability. Outside of these requirements, please configure policies according to good security and resilience practices.

1. Wireless devices connecting at the access points can reach the IoT Gateway.
2. The IoT Gateway can reach the data warehouse.
3. Only the IoT Gateway can reach the public Internet (for software updates - make it sure it can only receive software updates).

Your new topology should now look like this:


ISP
Router
Firewall+NAT
L3 switches
Wireless
Controller
AP
IoT
Gateway
AP
Data
Warehouse

Firewall: pfSense.

a. For addressing and firewalling architecture, do something similar to the enterprise network example in cycle 3 lecture. For policies, ensure appropriate reachability between endpoints (specified below).
b. Alternate option (requires instructor approval: Cisco ASA).

Task 5: Secure the Application (extra credit)

Task 3 shows us how to ensure security using authentication and isolation. However, security and privacy are not only protected by network isolation (which is achieved via router configurations as above) but also with encryption (TLS) and Public Key Infrastructure (PKI).

If you're not familiar with these terms, please watch a video on PKI basics here. Also, learn more about how PKI works in regular systems here. You just need to understand the basics here and don't need to have taken any classes in cryptography.

Since the IoT devices are constrained in their computation and memory capabilities, the usual techniques fall short. MQTT works over TCP and is not encrypted by default. However, there are versions of MQTT and TLS that work together to provide end-to-end encryption.

1. There exist some techniques (like reusing existing TLS sessions with session IDs, instead of renegotiations) to make TLS possible with MQTT.
2. There have been use-cases where the client certifications can be useful in identifying the identities of IoT devices, which requires the configuration of a complete PKI infrastructure.
3. There are also techniques like payload encryption which doesn't provide the strong security guarantees on message integrity and identity, but might be useful for devices which couldn't run TLS.

Your task is to read more about these techniques and try out one of these security techniques which are different than network isolation, implement it, and show your results. This question has been purposely kept open-ended to encourage you to read about different kinds of security techniques used in IoT. You should also cite the source of information that you use to understand & implement one of these techniques.

TAs will only provide you limited help with this part of the lab because it's extra credit.

Submission Requirements

Like the previous labs, for the first three task you should work as a team of at most **5** people on this lab. Each group can assign each student different tasks, and may get together to aggregate the system at the end. Since this lab involves configurations and setups, there are no or minimal codes to write. We would need students to submit a detailed report explaining their process to set up and configure every task. While explaining the process please make sure to include all the edits in the configuration files and commands used. You can also use screenshots to show your progress

Submission should include a **demo video** clearly showing the components in each task interact and operate as expected. Besides the demo video, you should also submit a **report** containing:

1) **GNS3 screenshot with clear text notes specifying the connection and interfacing/addressing of all the appliances and VMs.**
2) **Detailed steps for each task including configuration changes, setups, and commands used. Screenshots are also acceptable as long as they clearly show the configuration process.**
3) **A section on all the problems that you encountered during the lab. Please specify all the components that work and the ones which don't.**
4) **Your contribution for each part of the lab. Please explain precisely what you did. Writing in bullet point form is not allowed.**
5) **Things you added to the project that is outside the scope of this lab (for example, adding tips and troubleshooting suggestions in this doc).**
6) **Please also specify which person in the group does which part of the lab.**

Spring 2020 (MCS Online/Coursera) Submission requirement:

1. Each team member needs to upload a submission. (If one team member is uploading all the actual files, the other team members should still at least submit a text file listing team member names, NetIDs, and so on.) nam If you haven't signed up for your team, make sure you do it before the lab 3 deadline. [IoT Class: Project Groups](https://goo.gl/forms/6JCre9vLepxAG5aD3)
2. Please submit using Coursera. The repository should contain all the required files for lab 3 and a README.md specifying what each file is.
3. Please create a report (in .pdf format) which contains all the requirements specified on above. Please upload the report on Coursera, and write the netids of ALL group members in your team at the beginning. If your netid is NOT listed on the report, you will NOT be credited.
4. You may create separate videos for each task, or a single video if everything can fit in. In the video, similar to the report, demo the operation of different GNS3 components, as well as the deployed applications. For example, how are different VMs addressed? Can the Gaze tracking application produce meaningful output? (The same for optional tasks.) (50%)
5. Please upload your videos to Google Drive and share with the TAs and Prof. Matt Caesar

(tongm2@illinois.edu and caesar@illinois.edu). (You could also make a private video share on Illinois Media Space.)

Grading:

You'll be graded for smaller components in each task as well. So try to get as many individual components running the topology even if the entire task doesn't work.

Submission			Percentage
Demo	**Task 1**	**1. IoT gateway can reach data warehouse. 2. (Wireless) IoT devices connected to AP can reach data warehouse. 3. Addresses for each switches assigned to guarantee reachability.**	**30%**
	Task 2	**1. MQTT broker, subscriber, and publisher set up correctly. 2. Gaze running correctly with pop-up webcam video.**	**30%**
	Task 3 (extra credit)		*15%*
	Task 4 (extra credit)		*10%*
	Task 5 (extra credit)		*10%*
Report			**40%**
Contributions (extra credit)			*5%*

Troubleshooting (as we go along - add your suggestions here - extra credit)

We'll give you extra credit for helping others on Discord and adding your learnings/comments in this section.

1. [Amod] For students having trouble installing pfSense in GNS3, download the ISO image from the website and set up a new VM (FreeBSD), and install the image. Make sure to remove the image after installation otherwise the machine would keep rebooting into the installer. Once the installation and reboot is complete: create new interfaces - connect one to the internet (eth0) and other to your host to access GUI. You can use the GUI to configure the firewall and NAT.

2. [Amod] You can import any new VM in GNS3 by going to Preferences > VirtualBox VMs > New.

3. [Amod] In case the topology redundancy is confusing you, that is, having multiple L3 switches/Routers/Firewalls/APs/L2 switches: build your topology with the minimal number of nodes to begin with. Configure your routes with single AP, switch, firewall, and router. Redundancy is just to make your system more robust (and realistic in terms of deployment). However, you should always focus on conceptual understanding and basic deployment first.

4. [Amod] IoT Gateway Setup Guide Intel: https://software.intel.com/en-us/SetupGateway-MQTT. Remember you have to set up database as subscriber. You can set up the database and IoT gateway (broker) together.

5. [Amod] PostgreSQL has a docker image that comes with a data warehouse service called SpringBoot. It comes with APIs capable of MQTT subscription, storing MQTT messages in the database, and a REST interface to fetch MQTT messages from the database.

6. [Mike] If, when installing the Gaze app, you get an error when running `cmake` complaining that CMake could not find the openSSL module, you can try running `sudo apt-get install libssl-dev` within the Ubuntu VM of your Gaze app and try running `cmake` again.

7. [Amod - Yao and Tianyu] For students experiencing failed connectivity when using multiple switches, keep in mind that L2 switches might fall in the trap of network looping in this case. There are ways to tackle the route looping (if you're experiencing that ping fails or takes a lot of time). Look at the spanning tree protocol for Open vSwitch or BPDU for Cisco IOU.

8. [Jeff Hong] When installing the openwisp-config on openwrt, make sure to use the nossl version of the openwisp-config package

9. [Mike] If you are having trouble connecting your MQTT entities together but you can ping the relevant nodes, you may have to adjust the firewall rules on each node. In our group, we found out that our broker (running on Ubuntu 16.04) was denying TCP packets for port 1883 (Mosquitto default port) even though ping was working (IMCP packets). We used UFW to update the firewall rules: https://help.ubuntu.com/community/UFW Wireshark helped with debugging (we saw that the SYNs of the TCP handshake were going through to the broker node, but no SYN-ACKs where being sent back)
10. If you have "**VMware Workstation and Device/Credential Guard are not compatible"** while trying to start a new VM on Windows 10, this youtube video provides the solution: https://www.youtube.com/watch?v=CGpv2Dvzyeg

More Troubleshooting Tips

Here is how to make OpenWISP2 control OpenWRT (OpenWISP2's device list has the OpenWRT). The steps are as follows:

1. Follow GitHub openwisp/vagrant-openwisp2 to install OpenWISP2 in the VirtualBox and has IP 192.168.56.2. Make you install ansible with a specific version: 2.6 or you might see failure during "vagrant up" I prefer to create a virtualenv then do: `pip install ansible==2.6`
Ansible Vagrant profile to install an OpenWISP2 server - openwisp/vagrant-openwisp2

2. Follow http://openwisp.io/docs/user/configure-device.html to install OpenWRT in Virtualbox. Note: in Step 3, I choose <url> as
http://downloads.openwisp.io/openwisp-config/latest/openwisp-config-nossl_0.4.6a-1_all.ipk

3. Build GNS3 topology. Add OpenWRT as a GNS3 appliance. The key idea of my topology is that the OpenWRT will connect to cloud's vboxnet (virtual network). So that it can be inside the *subnet* 192.168.56.* Make sure it can ping 192.168.56.2 and OpenWISP2 can ping OpenWRT (OpenWRT can have IP 192.168.56.3).

4. Follow http://openwisp.io/docs/user/configure-device.html to edit configure file and type /etc/init.d/openwisp_config start to start registering.

5. Now in openwisp2, web page should list device you just registered

6. Some other points: OpenWRT in GNS3 has 3 interfaces (set in GNS3 configure). I also keep a NAT cloud in the topology, when the OpenWRT need to have Internet access (such as downloading packages), it can connect to NAT and change its WAN IP to 192.168.122.*.

To be controlled by the openwisp2, its eth1 should connect to cloud's vboxnet and wan IP should be set to 192.168.56.*.
(Amod: Only required to connect the APs to the internet for installation. Not a requirement by the MP's guidelines.)

Matthias: The problem with openwrt was linked to curl. I tested curl on the host pc but it seemed like the -k option was just ignored. I also extracted the server certificate from openwisp (with openssl) and added it with the --cacert option to the curl call but it was still the same error. Executing the same command with wget (with the correct options) worked though.
So I modified the source code of the openwisp plugin for openwrt (which is basically a bunch of bash scripts) and changed the AP registration to work with wget, which finally worked.
I really don't understand why curl was causing this issues and unfortunately, I couldn't find anything on google..
However, after some further investigation, I found out that the openwrt image we were using the whole time was an old one. I created a new VB with the latest version which worked without any modification of the plugin scripts. So the latest openwrt version with the correct configuration is working now.

OpenWISP/OpenWRT troubleshooting:
If you follow the instructions from the documentation and get a curl error (e.g. 35 or 60), curl has issues with the server certificate from OpenWISP (which is self signed). I would recommend the following steps to troubleshoot this:

- check if the latest version of OpenWISP and OpenWRT is installed
- check if “verify_ssl” in the configuration of the OpenWISP plugin (in the OpenWRT VB) is set to 0
- try to execute curl on the command line. To make sure that not the GNS3 topology is somehow causing the issues, add a “Host Only” and “NAT” adapter to both VB (OpenWISP and OpenWRT) and try to get the connection working without GNS3 first
- If curl fails also on the command line (don’t forget the -k option to disable certificate check), use openssl to extract the server certificate
- add the previously extracted server certificate from OpenWISP to the curl call (--cacert /path/to/cert)
- if it is still not working, wget can help. Check if wget is working.
- As the OpenWISP plugin for OpenWRT is just a bunch of bash scripts, it is possible to change the curl calls to wget. If wget is working, start with the register() method in the openwisp_config file (on github, the file is called openwisp.agent: https://github.com/openwisp/openwisp-config/blob/master/openwisp-config/files/openwisp.agent).
- Use man wget and man curl to map the curl parameters to wget (e.g. -k for curl is --no-check-certificate for wget, also, GET parameters are handled different- just add them to the URL like in the browser, etc.)
- If you want to debug something, use the built in logger like
    - logger -s “the string output” -t openwisp -p daemon.err (or daemon.info)
- start the script, you can check the log output with logread (better: logread | grep openwisp to get only the openwisp related log information)

Importing into GNS3:

- Check the GNS3 marketplace for a preconfigured, premade appliance. These are nice because they are confirmed to be working by someone else and come with these configs stored in a .gns3a appliance file that you can inspect in a texteditor. This is usually the most straightforward way to import a device and make it available in your GNS3 topology. Note that these appliances are directly imported into your GNS3 VM, and therefore are running inside your GNS3 VM.
- Most other things can be run off an Ubuntu VM, and then linked with GNS3. These VMs are not running in the GNS3 VM, but rather separately in VMWare/Virtualbox. You can directly interface with them in VMWare/Virtualbox. But you can also click and drag it onto your topology on GNS3. Right clicking the device in your topology and hitting “start” or “stop” will also spin up or spin down the VM in VMWare/Virtualbox.

GNS3 Network Troubleshooting tips:

- To verify that two devices are talking to each other, try pinging from one to the other (and vice versa).
- If there is some issue with pinging, one good way to troubleshoot this is to do the following in GNS3:
    - Right click a link along the path the packet is supposed to travel
    - Start capture to open wireshark
    - Investigate the individual packets that are flowing along this link. Take note of the source and destination and packet type.
- Almost all network issues in GNS3 can be solved by tweaking just a handful of properties:
    - Ip address
    - Subnet mask
    - DNS/gateway

Layer 2 switches will stupidly broadcast anything coming in on all interfaces going out. They support VLANs, but cannot do any IP routing.

A good way to start part 1 of this lab is to use just layer 2 switches that broadcast everything. This way, you can verify your network is fully connected and working properly before adding in a layer 3 switch, which might cause traffic to stop flowing properly - but at that point you can isolate it to the layer 3 switch.