COP5614 Final Project Report

The final project for COP 5614 Operating Systems - Fall 2018.

Team #12:

Liqun Yang	PID: 6072714	email: lyang028@fiu.edu
Yuzhou Feng	PID: 4884556	email: yfeng015@fiu.edu
Tianren Yang	PID: 5966794	email: tyang009@fiu.edu

Structure

- lkm/ Network status monitoring loadable kernel module
  - lkm/lkm_process_info.c lkm program to monitoring network status
  - lkm/Makefile makefile for lkm
- print/ The program to save network status to file
  - print/bin/ Folder for print program execution files
  - print/lkm_print.c Print program source code
  - print/Makefile makefile for print program
  - print/start_print.sh batch for autorun
- visualization/ Network topology visualization program
  - visualization/static Static files folder for node js site
    - visualization/static/main.js Front end code
    - visualization/static/com.png Computer icon for visualization
    - visualization/static/data.json Generated data for network topology
  - visualization/index.html Web page for network topology visualization
  - visualization/package.json Nodejs project configuration file
  - visualization/server.js Nodejs project serving code
- README.md Introduction for project

I. Docker Environment Setup

1. Setup docker To install docker on Ubuntu, please check out this official instructions: https://docs.docker.com/install/linux/docker-ce/ubuntu/#os-requirements

2. Setup docker compose We are using docker compose to setup Docker environment. The configuration of the testing Environment has been saved in the YAML file - docker-compose.yml. To setup the system, you may need to install the docker-compose. Please check out this official instructions: https://docs.docker.com/compose/install/#install-compose
3. Build and up the docker containers

   ```
   docker-compose up -d --build
   ```

4. To list all running containers

   ```
   sudo docker ps
   ```

If you have already have a testing environment, you can simply skip this step and test the rest parts of the project.

II. Network Status Monitoring Loadable Kernel Module

This Loadable Kernel Module program is basing on netfiler framework, for which monitoring the communications between docker containers.

Usage

Compile and insert Loadable Kernel Module

1. Get into print directory

   ```
   cd print/
   ```

1. To compile and insert lkm to kernel

   ```
   sudo make
   sudo insmod lkm_process_info.ko
   ```

Compile and run the user space program to get the information from the lkm which is running in the kernel space.

1. Get into print folder

```
cd print/
```

1. Compile the print program by Makefile, a lkm_print executable file will be generated

```
Make
```

1. Run the print program, and all the information collected by the lkm will be fetched by this user level program, and then store them in the "data.json" file under visualization/static/ directory.

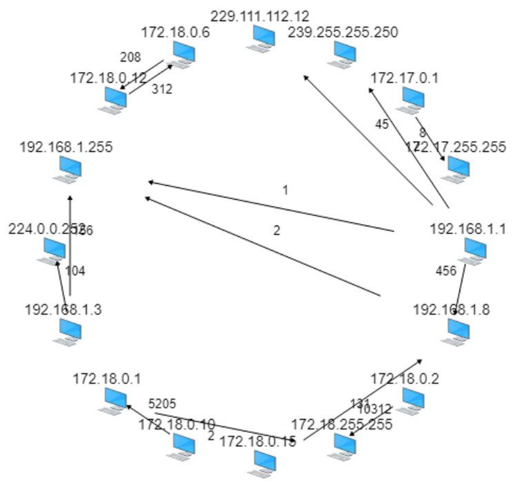
```
./lkm_print
```

1. Use Golang to automatically invoke the print program

```
go run auto_run.go
```

III. Network Topology Visualization

The network topology visualization is served on Nodejs, which automatically fetch network information generated from Part I and Part II, and output the result on a web page through 5000 port.



1. Install NodeJS First you need to install node on your machine, please check out this instructions: https://www.rosehosting.com/blog/install-npm-on-ubuntu-16-04/
2. Get into visualization directory

```
cd visualization/
```

3. Install dependencies

```
npm install
```

4. Serve the NodeJS site

```
node server.js
```

5. Check out the result Open any modern web browser, which support Canvas feature, and type in localhost:5000 at address bar. Press enter, and the network topology visualization will be rendering on the page.

IV. Docker Engine Integration

The docker development group provide a standard development environment for all users to make it easier to contribute to Docker. The standard development environment defines all build dependencies: system libraries and binaries, go environment, go dependencies, etc. After we install docker in our system, we need to do the following steps to compile the Docker source code.

1. Build the environment Run this command in current “docker” directory, and the first build will take some time to complete.

```
$ make build
```

2. Build the Docker binary Run this command to build the docker binary, and the binary file will be created in “./bundles/-dev/binary/”.

```
$ make binary
```

3. Run the Tests To execute the test cases, run this command:

```
$ make test
```

There are many Docker Engine Plugins available in Docker Engine which extend Docker's functionality. They typically come in three specific types—network plugins, volume plugins and authorization plugins. According to the instruction, the LKM function should be integrated into the docker source code. In this case, we try to install a network plugin in to the docker engine, which makes the containers from different groups to communicate with each other. With the help of the tool that can monitor the network activities, we can examine any packet that originates from or is destined to as well as record the log information into a file.

In practice, we write a golang file called auto_run.go which execute the binary file lkm_print every 3 second. In this way, we can monitor the package transmission and use the real time data for the network topology visualization. However, we didn't integrate the code into the Docker source code, so we need to run this file by this command:

```
go run auto_run.go
```

Contribution

- Liqun Yang(6072714)
  - Part 2: Network Monitoring Tool Design with LKM and C
  - Part 3: Parse network information into JSON format
- Yuzhou Feng(4884556)
  - Part 1: Docker Environment Setup
  - Part 3: Network Topology Visualization with Nodejs and Javascript
  - Documentation
  - Video
- Tianren Yang(5966794)
  - Part 2: Automatically load print program
  - Part 4: Docker Engine Integration
  - Documentation