comnetsemu

Star 0

324 Commits 3 Branches 4 Tags

Topics: SDN NFV Network Emul... + 1 more

A virtual emulator/testbed designed for the book: Computing in Communication Networks: From Theory to Practice (2020)

Add reference for the book
Tung Doan Van authored 5 months ago
6ed4f726

master comnetsemu History Find file Code

README MIT License CHANGELOG

Name	Last commit	Last update
.github/workflows	Port provision shell scripts to Ansible playbooks	1 year ago
app	Update flowvisor related scripts in multi_tenant_sdn_slici...	2 years ago
bin	Use sphinx for better API documentation	4 years ago
comnetsemu	Bump up to 0.3.1	1 year ago
doc	Update docs.yml	2 years ago
examples	Merge docs.yml to ci.yml	2 years ago
patch/mininet	Port provision shell scripts to Ansible playbooks	1 year ago
test_containers	Improve tools/scripts for setting up the test VM	2 years ago
util	Bump up to 0.3.1	1 year ago
.gitattributes	<code_base>: Cleanup codebase, use a separate mod...	4 years ago
.gitignore	Merge docs.yml to ci.yml	2 years ago
.pylint	Improve code quality using Pylint.	4 years ago
.shellcheckrc	<fix>: Fix warnings of shellcheck.	3 years ago
CHANGELOG.md	Bump up to 0.3.1	1 year ago
CONTRIBUTORS	Update CONTRIBUTORS.	3 years ago
LICENSE	Revise README and tests, add LICENSE	4 years ago
Makefile	Merge docs.yml to ci.yml	2 years ago
README.md	Add reference for the book	5 months ago
Vagrantfile	Bump up to 0.3.1	1 year ago
setup.py	Update documentation and dependencies to build Sphi...	2 years ago

README.md

license MIT ComNetsEmu CI passing

ComNetsEmu

A virtual emulator/testbed designed for the book: Computing in Communication Networks: From Theory to Practice

If you like or use ComNetsEmu, please cite our book:

```
@book{CompBook,
title = {Computing in Communication Networks \textendash From Theory to Practice},
author = {Frank H. P. {Fitzek} and Fabrizio {Granelli} and Patrick {Seeling}},
isbn = {9780128204887},
year = {2020},
date = {2020-05-21},
volume = {1},
publisher = {Elsevier},
edition = {1},
series = {1},
note = {https://cn.ifn.et.tu-dresden.de/compcombook/},
keywords = {},
pubstate = {published},
tppubtype = {book}
}
```

This project has a online Github page for Python API documentation and other useful documentation. Please check it when you develop or use ComNetsEmu's Python APIs.

INFO: This project is currently still under development [beta]. Version 1.0.0 has not yet been released. We try to make it stable but breaking changes might happen.

The repository is hosted both on the internal Gitlab server of ComNets TUD and Github. The GitLab ComNets TUD is **read-only** for public users. For all public users, if you have a question or want to contribute, please create an issue or send a pull request **on Github**.

Table of Contents

Description

ComNetsEmu is a testbed and network emulator designed for the NFV/SDN teaching book "Computing in Communication Networks: From Theory to Practice". The design focuses on emulating all examples and applications on a single computer, for example on a laptop. ComNetsEmu extends the famous Mininet network emulator to support better emulation of versatile **Computing In The Network (COIN) applications**. It extends and puts forward the concepts and work in the Containernet project. It uses a slightly different approach to extend the Mininet compared to Containernet. It's main focus is to use "sibling containers" to emulate network systems with **computing**. See a more detailed comparison between upstream Mininet and Containernet here.

Common facts about ComNetsEmu:

- Emulation accuracy is highly considered, but it but can not be guaranteed for arbitrary topology. All emulated nodes (processes) share the same underlying compute, storage and network resources when running it on a single system. ComNetsEmu is heavier than vanilla Mininet due to stricter node isolation. Choosing a reasonable emulation parameters is required for correct simulation results. RT-Tests is installed on the test VM. RT-Tests can be used to evaluate the real-time performance of the current emulation system.

- ComNetsEmu is mainly developed with **Python3.8**. To reduce the complexity of dependencies (third-party packages, frameworks etc.), ComNetsEmu tries to leverage as much of the powerful Python standard library as possible, and prefers simple third-party dependencies when necessary.

- Examples and applications in this repository are mainly developed with **high-level** script language for simplicity. These programs are **not** performance optimized. Please contact us if you want highly optimized implementation of the concepts introduced in this book. For example, we had a DPDK-accelerated version of the low-latency (sub-millisecond) Random Linear Network Coding (RLNC) network function.

Main Features

- Use Docker hosts in Mininet topologies.

- Manage application Docker containers deployed **inside** Docker hosts. "Docker-in-Docker" (sibling containers) is used as a lightweight emulation of nested virtualization. A Docker host with multiple **internal** Docker containers deployed is used to **mimic** an actual physical host running Docker containers (application containers).

- A collection of application examples for "Computing In Communication Networks" with sample codes and detailed documentation. All examples can be easily reproduced and extended.

Check the Roadmap for planed and WIP features.

Installation

Currently, only the latest Ubuntu 20.04 LTS is supported. Supporting multiple Linux distributions and versions is not the goal of this project.

It is highly recommended to run ComNetsEmu **inside** a virtual machine (VM). **Root privileges** are required to run the ComNetsEmu/Mininet applications. ComNetsEmu also uses privileged Docker containers by default. It's also safer to play it inside a VM. ComNetsEmu's installation script is a wrapper of an Ansible playbook. This playbook uses Mininet's install script to install Mininet natively from source. As described in Mininet's doc, the install script is a bit **intrusive** and may possible **damage** your OS and/or home directory. ComNetsEmu runs smoothly in a VM with 2 vCPUs and 2GB RAM. (Host Physical CPU: Intel i7-7600U @ 2.80GHz). Some more complex applications require more resources. For example, the YOLO object detection application requires a minimum of 5GB of memory.

The recommended and easiest way to install ComNetsEmu is to use Vagrant and Virtualbox. Assuming that the directory where ComNetsEmu is stored is "~/comnetsemu" in your home directory, just run the following commands to get a fully configured VM using vagrant with Virtualbox provider:

```
$ cd ~
$ git clone https://git.comnets.net/public-repo/comnetsemu.git
$ cd ./comnetsemu
$ vagrant up comnetsemu
# Take a coffee and wait about 15 minutes

# SSH into the VM when it's up and ready (The ComNetsEmu banner is printed on the screen)
$ vagrant ssh comnetsemu
```

Mainly due to performance and special feature requirements, some examples and applications can only run on virtual machines with KVM as the hypervisor. The built-in Vagrantfile provided by ComNetsEmu supports libvirt provider. Please check the detailed documentation of Option 1 here if you want to use the libvirt provider for Vagrant.

Congratulations! The installation is done successfully! You can now run the tests, examples, and **skip** the rest of the documentation in this section. Continue reading only if you are interested in the details of the installation or want other installation options.

For users running Windows as the host OS:

Warning: Main developers of ComNetsEmu does not use Windows and does not have a Windows machine to test on.

1. If you are using Windows, we recommend using Mobaxterm as the console. This should solve problems opening `xterm` in the emulator.

ComNetsEmu's installer will try to install the dependencies using a package manager (apt, pip, etc.) if the desired version is available. Unavailable dependencies (e.g. the latest Mininet) and dependencies that require patching are installed directly from source code. By default, the dependency source codes are downloaded into `"~/comnetsemu_dependencies"`. You can modify the Ansible playbook based on your needs.

Please see the detailed installation guide here for additional installation options.

Upgrade ComNetsEmu and Dependencies

ComNetsEmu's installer can only upgrade when ComNetsEmu's underlying Linux distribution is **not changed/upgraded**. For example, you can use this upgrade function when Ubuntu 20.04 LTS is used as the base VM. When the base VM is upgraded to the next LTS version, the upgrade function is not guaranteed to work since many packages are upgraded. Therefore, it's suggested to `vagrant destroy` and `vagrant up` again when a new Ubuntu LTS is used as the base VM. Thanks to Vagrant and Docker packaging, it should be not too difficult to re-create the environment after rebuild the VM.

Example screenshots for running the upgrade process in terminal:

The **master** branch contains stable/tested sources for ComNetsEmu's Python package, utility scripts, examples and applications. It is **recommended** to upgraded to **latest** published tag of the **master** branch.

The installer script has a function to upgrade ComNetsEmu automatically. And the installer script also needs to be updated firstly. Therefore, it takes **three** steps to upgrade everything. It is assumed here the ComNetsEmu is installed using option 1 with Vagrant.

Step 1: Upgrade source code of ComNetsEmu Python package, examples and applications

Use git to pull (or fetch+merge) the latest tag (or commit) in master branch:

```
$ cd ~/comnetsemu
$ git checkout master
$ git pull origin master
```

Step 2: Automatically upgrade ComNetsEmu Python modules and all dependencies

The installer script is used to perform this step. Run following commands **inside** the VM to upgrade automatically:

```
$ cd ~/comnetsemu/util
$ bash ./install.sh -u
```

The script may ask you to input yes or no several times, please read the terminal output for information.

Step 3: Check if the upgrade is successful

Run following commands inside the VM to run tests:

```
$ cd ~/comnetsemu/
$ sudo make test && sudo make test-examples
```

If all tests pass without any errors or exceptions, the upgrading was successful. Otherwise, it is recommended to redo the upgrade process or just rebuild the Vagrant VM if the situation is too bad...

Run the Docker-in-Docker example

```
$ cd $TOP_DIR/comnetsemu/examples/
$ sudo python3 ./dockerindocker.py
```

See the README to get information about all built-in examples.

Project Structure

To keep the VMs small, Vagrantfile and test_containers contain only **minimal** dependencies to start the VMs and be able to run all the built-in examples. Dependencies of specific applications (e.g. Python packages like numpy, scipy etc.) should be installed by the script or instructions provided in the corresponded application folder. Therefore, the user need to install them **only if** she or he wants to run that application.

- app: All application programs are classified in this directory. Each subdirectory contains a brief introduction, source codes, Dockerfiles for internal containers and utility scripts of the application

- bin: Commands and binaries provided by ComNetsEmu

- comnetsemu: Source codes of ComNetsEmu's Python packages

- doc: Markdown files and sources to generate ComNetsEmu Sphinx documentation

- examples: Example programs for functionalities of the ComNetsEmu emulator

- patch: Patches for external dependencies that are installed from source code via installer

- test_containers: Contains Dockerfiles for essential Docker images for tests and built-in examples

- utils: Utility and helper scripts

- Vagrantfile: Vagrant file to setup development/experiment VM environment

Development Guide and API Documentation

Please check the online documentation page.

FAQ

Check faq

Contributing

This project exists thanks to all people who contribute. List of known contributors.

For all public users, please create issues or send pull requests on Github.

Contact

Project main maintainers:

- Zuo Xiang: zuo.xiang@tu-dresden.de (office), xianglinks@gmail.com (personal)