CS4090 Programming Project — Software Setup

Responsible TAs: Janice van Dam (j.vandam-3@tudelft.nl)
Soubhadra Maiti (s.maiti-1@tudelft.nl) Bethany Davies (b.j.davies@tudelft.nl)

March 17, 2025

1 Introduction

The purpose of this document is to help you (the student) set up the required software for the programming project. We first guide you through the installation of a virtual machine (VM) that includes all the required software packages you need for this project (Section 2). We then explain how to use the software packages and how to access the documentation for such packages (Section 3). Additionally, we give a set of useful tips for completing the project (Section 4).

> **Note**
>
> You are free to install the required software packages on your own machine. The steps to do that in a Linux system are given in Section 5. However, we WILL NOT provide support for this type of installation, as we cannot account for all possible problems that may arise in all possible systems. Proceed without the virtual machine at your own risk.

2 Setting up the Virtual Machine

To run the virtual machine, you will need:

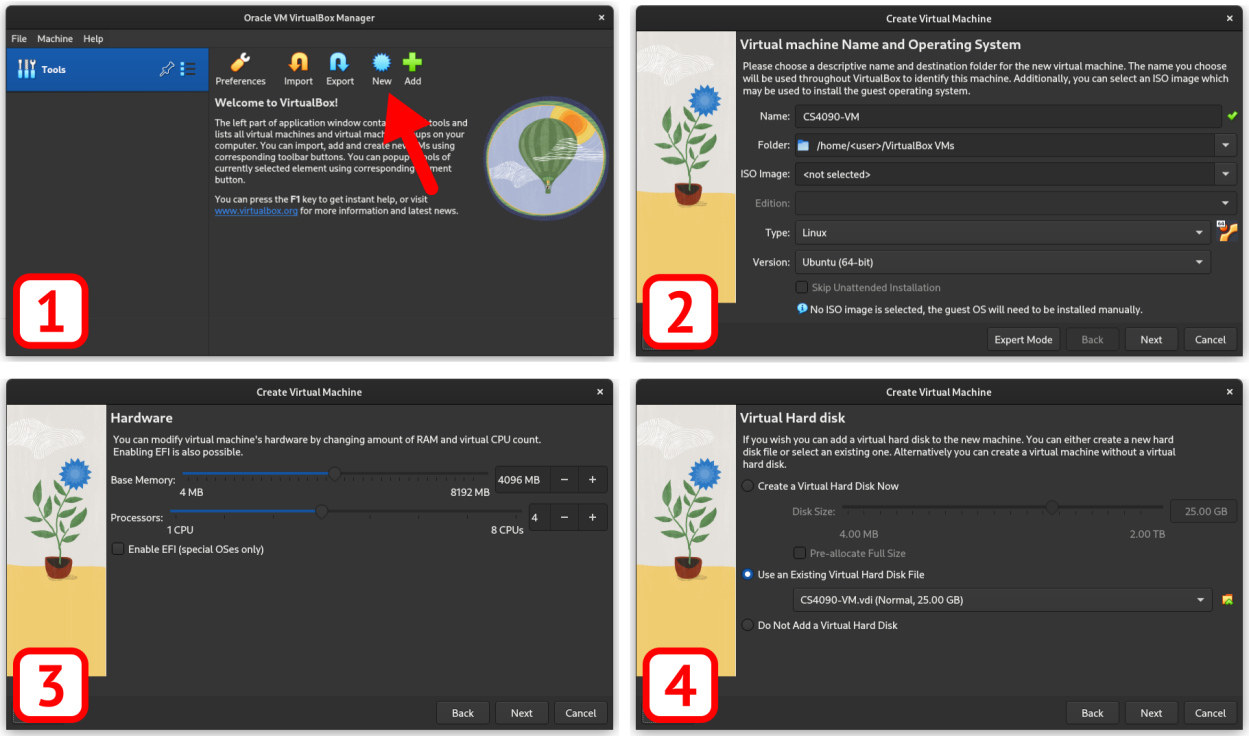
- A computer with a 64-bit operating system installed
- The VirtualBox virtualization software, which you can obtain from the following link: https://www.virtualbox.org/wiki/Downloads
- The virtual disk image (VDI) of the virtual machine. This is a large (13 GB) file so make sure you have disk space for it. Download the VDI from the following link (download password: `cs4090`): https://surfdrive.surf.nl/files/index.php/s/2KkZhXV1fgX2ZO3. If nothing happens upon entering the password, try a different browser.

> **Note**
>
> Mac users should be aware that there is a short window (30 minutes) of opportunity to enable the correct permissions for VirtualBox upon installation. See this guide if you get a kernel driver error: https://medium.com/@Aenon/mac-virtualbox-kernel-driver-error-df39e7e10cd8

Once VirtualBox has been installed, launch the VirtualBox application, and then set up the virtual machine following these steps:

- Step 1: On the VirtualBox starting screen, select "New" to start creating a new virtual machine.
- Step 2: On the next screen, give the virtual machine a name, e.g. "CS4090-VM", and select "Linux" as type and "Ubuntu (64-bit)" as version.

- Step 3: On the next screen, allocate the preferred amount of memory and CPU processors to the virtual machine. We recommend at least 4096 MB of memory and 4 CPU processors. Allocating less may make the system less responsive.
- Step 4: On the next screen, choose "Use an Existing Virtual Hard Disk File", and click the folder icon to navigate to the VDI file you have downloaded.
- Final step: On the final screen, click "Create" and you should be brought back to the main VirtualBox window, with an entry added for the new virtual machine.



You can now start the virtual machine by selecting it and pressing "Start". Once it loads you should be brought to a login screen for user `cs4090`. The password is `cs4090`.

Note

When trying to run the virtual machine, if you run into errors like "VT-x is disabled" (or "AMD-V is disabled" if you have an AMD CPU), it might mean that the hardware acceleration settings required by the CPU to support virtualization are currently disabled in your BIOS. You should be able to find some guide online that explains the process of entering the BIOS and enabling the virtualization features needed to run virtual machines. Try to search for "<computer model> virtualization BIOS".

Note

When you are finished using the virtual machine you should power it down by going to the upper right corner WITHIN the virtual machine and choosing "Power Off...". This just makes sure that the operating system shuts down properly and prevents file corruption from unexpected shutdown.

3 Software and Documentation

For this project, you will be simulating quantum communication applications written in NetQASM [1], which will be run on NetSquid—a simulator for quantum networks [2]. The SquidASM package provides a way for

NetQASM applications to be run on NetSquid.

The software packages needed to run simulations are installed inside of a Python virtual environment. Don't forget to activate the virtual environment before running your simulations (or configure the IDE to do so). See https://realpython.com/python-virtual-environments-a-primer/ (optional) to know more about how virtual environment works. Open up the terminal and run the following commands:

```bash
$ cd ~/CS4090
$ source venv/bin/activate
```

The documentation for NetQASM and SquidASM includes usage of the software and some examples. From the virtual machine, open the following HTML files from the file manager to view the NetQASM documentation and the SquidASM documentation in a web browser:

```
~/CS4090/libs/netqasm/docs/build/html/index.html
~/CS4090/libs/squidasm/docs/build/html/index.html
```

You can find the project template files at the following location:

```
~/CS4090/project
```

4 Extra Tips

Here are some useful tips to keep in mind when going through the programming project:

- In this programming project you will be writing Python code. We strongly recommend the use of an IDE for writing your code as it will make your life a bit easier than using a standard text editor. A popular IDE for Python is PyCharm, which you can find installed in the virtual machine already.
- Getting the state of a qubit in the simulation can be done using the following:

  ```python
  from netqasm.sdk.external import get_qubit_state
  density_matrix = get_qubit_state(qubit, reduced_dm=False)
  ```

  This will give you the state of a qubit as a density matrix. Using `reduced_dm=False` will make sure you get a multi-qubit state if the qubit is entangled with others. If you want the reduced density matrix (i.e. all other qubits are traced out), then you can simply set `reduced_dm=True`.
- To ensure you get the most up-to-date state of a qubit, you should call the method `.flush()` on the `NetQASMConnection` object. This makes sure all the gates and measurements you have applied to the qubits are accounted for in a subsequent call to get qubit state.
- Qubit states are represented as `numpy` arrays. If you want to compute the fidelity of a density matrix returned by get qubit state to some other state, you can simply make a `numpy` array for the target state and compute the fidelity as we have seen in class.
- You can play around with the fidelity of EPR pairs produced by EPR sockets as well as the fidelity of gates by modifying the relevant parameters in the `network.yaml` files within your application code.

5 Manual Installation of Software Packages

You are free to install the required software packages on your own machine. The steps to do that in a Linux system are given here. However, we WILL NOT provide support for this type of installation, as we cannot

account for all possible problems that may arise in all possible systems. Proceed at your own risk.

5.1 Registering for NetSquid

In this project we will be using NetSquid, a quantum network simulator geared at modeling the physics of quantum hardware [2]. We will not be using this software directly, but it is a tool needed by the software we expect you to use for the assignment. In order to install NetSquid, it is necessary to register an account at https://forum.netsquid.org/ucp.php?mode=register. The username and password you choose here will be needed in order to complete software setup in the next step.

5.2 Setting Up the Software Packages

First of all, download the compressed archives `netqasm.zip` and `squidasm.zip` from the "Programming Project" page on Brightspace.

Then, create a directory with a name of your choice. Here, we'll create a directory named `CS4090`, placed directly under the home directory. Then, unzip the compressed archives into a subdirectory `libs`:

```bash
$ mkdir ~/CS4090 && cd ~/CS4090
$ mkdir libs
$ mv /path/to/netqasm.zip libs
$ mv /path/to/squidasm.zip libs
$ unzip libs/netqasm.zip -d libs
$ unzip libs/squidasm.zip -d libs
```

Then, create and activate a virtual environment, which you'll be using to install the required packages (here, we named the virtual environment `venv`):

```bash
$ virtualenv venv
$ source venv/bin/activate
```

Then, install NetQASM (the last two commands verify that everything is working properly):

```bash
$ cd libs/netqasm
$ make install
$ make tests
$ make examples
$ cd ../..
```

Then, install SquidASM (the last two commands verify that everything is working properly):

```bash
$ cd libs/squidasm
$ export NETSQUIDPYPI_USER=<username_for_netsquid_registration>
$ export NETSQUIDPYPI_PWD='<password_for_netsquid_registration>'
$ make install
$ make tests
$ make examples
$ cd ../..
```

> **Note**
>
> The single quotes around the password for NetSquid ensure that any special characters that may be present in your password are treated correctly.

You can then download the project template files from the "Programming Project" page on Brigthspace. Don't forget to activate the virtual environment before running your simulations.

References

[1] A. Dahlberg, B. van der Vecht, C. Delle Donne, M. Skrzypczyk, I. te Raa, W. Kozlowski, and S. Wehner, "NetQASM-A low-level instruction set architecture for hybrid quantum-classical programs in a quantum internet," *Quantum Science and Technology*, 2022.

[2] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. de Oliveira Filho, M. Papendrecht, J. Rabbie, F. Rozpędek, M. Skrzypczyk, *et al.*, "Netsquid, a network simulator for quantum information using discrete events," *Communications Physics*, vol. 4, no. 1, pp. 1–15, 2021.