# Part-3

# CS562_Applied Software Engineering

# Alex Groce

# By: Rafid Almahdi

**Testing file**: cont.py

**Functions Tested**:

build():

addSwitch():

get():

Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.

Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.

Mininet provides an easy way to get correct system behavior (and, to the extent supported by your hardware, performance) and to experiment with topologies.

Mininet networks run real code including standard Unix/Linux network applications as well as the real Linux kernel and network stack (including any kernel extensions which you may have available, as long as they are compatible with network namespaces.)

Because of this, the code you develop and test on Mininet, for an OpenFlow controller, modified switch, or host, can move to a real system with minimal changes, for real-world testing, performance evaluation, and deployment. Importantly this means that a design that works in Mininet can usually move directly to hardware switches for line-rate packet forwarding.

Nearly every operating system virtualizes computing resources using a process abstraction. Mininet uses process-based virtualization to run many (we've successfully booted up to 4096) hosts and switches on a single OS kernel. Since version 2.2.26, Linux has supported network namespaces, a lightweight virtualization feature that provides individual processes with separate network interfaces, routing tables, and ARP tables. The full Linux container architecture adds chroot() jails, process and user namespaces, and CPU and memory limits to provide full OS-level virtualization, but Mininet does not require these additional features. Mininet can create kernel or user-space OpenFlow switches, controllers to control the switches, and hosts to communicate over the simulated network. Mininet connects switches and hosts using virtual ethernet (veth)

pairs. While Mininet currently depends on the Linux kernel, in the future it may support other operating systems with process-based virtualization, such Solaris containers or !FreeBSD jails.

Mininet's code is almost entirely Python, except for a short C utility.

Once a design works on Mininet, it can be deployed on hardware for real-world use, testing and measurement.

To successfully port to hardware on the first try, every Mininet-emulated component must act in the same way as its corresponding physical one. The virtual topology should match the physical one; virtual Ethernet pairs must be replaced by link-level Ethernet connectivity. Hosts emulated as processes should be replaced by hosts with their own OS image. In addition, each emulated OpenFlow switch should be replaced by a physical one configured to point to the controller. However, the controller does not need to change. When Mininet is running, the controller "sees" a physical network of switches, made possible by an interface with well-defined state semantics.

I test many function in Mininet system.I found mininet is a great system and helpful to design and implementation network.  I test controller.py and test build (), add Switch (), get () function. But I think there are some error grammar in my tstl, but I will solve this error in the next part.