Exercise Introduction

At the center of the IPMI architecture is the baseboard management controller (BMC), a dedicated mini-computer independent of the host server. This infrastructure is not present for the OpenStack cloud VMs used in this course.[1]

Therefore we will employ the simulator `ipmi_sim` provided by the OpenIPMI project and execute it on a cloud VM. The simulated BMC can be accessed via the IPMI protocol with utilities such as `ipmitool`.

The simulation comprises in particular the following features, which will be explored in the tasks below:

- Sensors (e.g. temperature of the host server mainboard as fixed values that are manipulated manually)
- Power controls (powering on/off the host server, which is represented by a VM created by the simulator)
- Serial over LAN (console output of the host server, i.e. the created VM)

The simulated BMC can be thought of as e.g. `gcn2960-bmc` of the Emmy compute node `gcn2960`. In this setting `ipmitool` to access `gcn2960-bmc` would be executed on the Emmy admin node `gadm1`.

The goal of this exercise is to use the simulator to learn IPMI. Setting up the simulator itself is part of the exercise but secondary.

Contents

Task 1: Basic Setup (5 min)

In this task we install the simulator and run the first IPMI commands.

1. Connect via SSH to a cloud VM running Rocky Linux 9. For this you can use the head node of the cluster set up in previous exercises.
2. Install the following package, which will be used to interact with the IPMI simulator:

   ```
   $ sudo dnf install -y ipmitool
   ```

3. We will compile OpenIPMI ourselves, which requires the packages:

   ```
   $ sudo dnf install -y tar gcc popt-devel python-devel readline-devel
   ```

[1] The project https://github.com/openstack/virtualbmc provides a virtual BMC for controlling virtual machines using IPMI commands. It supports a limited set of commands and is intended for development use only.

4. Download and extract OpenIPMI version 2.0.33 in your home directory:[2]

   ```
   $ wget -O - https://sourceforge.net/projects/openipmi/files/\
   OpenIPMI%202.0%20Library/OpenIPMI-2.0.33.tar.gz | tar -C ~ -xzf -
   ```

5. Compile and install the code to `~/openipmi`:

   ```
   $ (cd ~/OpenIPMI-2.0.33 && ./configure --prefix=$HOME/openipmi && make && make install)
   ```

6. Now we can already start the simulator:

   ```
   $ (cd ~/openipmi && ./bin/ipmi_sim -c ./etc/ipmi/lan.conf -f ./etc/ipmi/ipmisim1.emu -s ~)
   ```

   Here `lan.conf` and `ipmisim1.emu` are configuration files and `-s` specifies the path, where a state directory named `ipmi_sim` will be created.

7. In a separate terminal, colored blue from now on, we access the simulated BMC and display basic information about the management controller (MC):[3]

   ```
   $ ipmitool -I lanplus -U ipmiusr -P test -p 9001 -H localhost -C0 mc info
   Device ID                 : 0
   Device Revision           : 3
   Firmware Revision         : 9.08
   IPMI Version              : 2.0
   ...
   ```

8. Start exploring the functionality of `ipmitool` by replacing `mc info` with `help` and other IPMI commands.

Task 2: Sensors and Event Log (10 min)

The system event log is an important IPMI component as it persists after a reboot of the host server. This is in particular of use in case of HPC compute nodes, which are often redeployed on reboot. The log contains events like sensors crossing certain defined thresholds or hardware errors. In this task we configure the simulator to support sensors and observe their interplay with the event log.

1. The configuration files of the simulator already contain some sensors with fixed predefined values. However, trying to access the sensor data repository (SDR), we get the following error:

   ```
   $ ipmitool -I lanplus -U ipmiusr -P test -p 9001 -H localhost -C0 sdr list
   SDRR successfully erased
   Err in cmd get sensor sdr info
   Get SDR 0000 command failed: Requested sensor, data, or record not found
   ...
   ```

2. To fix it stop the simulation by typing `quit` in the terminal that runs `ipmi_sim`.

3. We need to copy an additional senor configuration file into the state directory:

   ```
   $ cp ~/OpenIPMI-2.0.33/lanserv/ipmisim1.bsdr ~/ipmi_sim/ipmisim1/sdr.20.main
   ```

4. Start `ipmi_sim` again with the same command as in Task 1.

5. Now, we should obtain proper sensor data via IPMI:

   ```
   $ ipmitool -I lanplus -U ipmiusr -P test -p 9001 -H localhost -C0 sdr list
   ```

[2] The simulator `ipmi_sim` of OpenIPMI 2.0.36, which is packaged with Rocky Linux 9, does not function properly for our use case.
[3] The hostname `localhost` is an unfortunate coincidence of our simulator setup. The cipher `-C0` should not be used outside of this simulation. In a bare metal setting the command executed on an admin node like `gadm1` to reach the BMC of a compute node `gcn2960` could be: `ipmitool -I lanplus -U ipmiusr -P strongpassword -H gcn2960-bmc mc info`

```
MBTemp           | 96 degrees C      | ok
SubTemp          | 0 degrees C       | ok
mm1pres          | 0x01              | ok
```

6. The value `96 degrees C` of `MBTemp` is hard-coded in the configuration files of the simulation. Change this value by typing[4] the following command[5] into the terminal running the simulation:

   ```
   sensor_set_value 0x20 0 1 0x95 1
   ```

7. Verify that the IPMI command `sdr list` now shows a different value for `MBTemp`. Which one?

8. You should find entries in the system event log (SEL), indicating that the value crossed a threshold. Check via:

   ```
   $ ipmitool -I lanplus -U ipmiusr -P test -p 9001 -H localhost -C0 sel elist
      1 |  Pre-Init  |0000003272| Temperature MBTemp | Upper Non-critical going high ...
      2 |  Pre-Init  |0000003272| Temperature MBTemp | Upper Critical going high ...
   ```

9. Inspect the event in detail using the IPMI command `sel get <ID>`, where `<ID>` is a number from the first column of the previous output.

10. Use `sensor_set_value` once more to change the simulated value of `MBTemp` below the threshold and observe the event log.

11. *Bonus:* How can you modify the simulated value of `mm1pres`, i.e. the presence indicator for the first memory module?

Task 3: Power Controls (5 min)

IPMI commands to restart and power on/off a host server are frequently used, in particular when SSH access is not working. To simulate these commands, we have to set up virtualization on our cloud VM, as the host server will be represented by a VM running on that cloud VM. This nested VM will be started/stopped every time the host server is powered on/off via IPMI.

1. To be able to run the VM, we install:

   ```
   $ sudo dnf install -y qemu-kvm
   ```

2. In addition, we need some Linux image, e.g.:

   ```
   $ wget -O ~/linux.qcow2 \
   https://dl-cdn.alpinelinux.org/alpine/v3.22\
   /releases/cloud/generic_alpine-3.22.2-x86_64-bios-tiny-r0.qcow2
   ```

3. And we have to update our configuration file. Change in `~/openipmi/etc/ipmi/lan.conf` each line beginning with `startcmd "qemu-system-x86_64` to the following, where you have to remove the line-break symbolized by `\`:

   ```
   startcmd "/usr/libexec/qemu-kvm -drive file=../linux.qcow2 \
   -nographic -serial mon:telnet::9003,server,telnet,nowait"
   ```

4. Next, quit and restart `ipmi_sim` again with the command from Task 1.

5. Try to power on the host server with:

   ```
   $ ipmitool -I lanplus -U ipmiusr -P test -p 9001 -H localhost -C0 power on
   ```

[4] Do not copy & paste because this regularly results in errors such as: `Unknown command`
[5] See the documentation with: `man ~/openipmi/share/man/man5/ipmi_sim_cmd.5`

6. Check that it is indeed powered on with the IPMI command `power status`.
7. Finally, turn it off again using the IPMI command `power off`.

Task 4: Console Output (5 min)

Employing the IPMI feature serial over LAN (SOL), we can see the console output of the host server. This is particularly valuable to debug the boot process, when SSH login is not yet available.

1. Start the host server with the IPMI command `power on`.
2. Then to see the console output (i.e. boot messages of the Linux kernel) run:

   ```
   $ ipmitool -I lanplus -U ipmiusr -P test -p 9001 -H localhost -C0 sol activate
   ...
   [    0.451393] Policy zone: DMA32
   [    0.451670] mem auto-init: stack:all(zero), heap alloc:on, heap free:off
   [    0.458572] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
   [    0.459496] Dynamic Preempt: none
   [    0.459573] rcu: Preemptible hierarchical RCU implementation.
   ...
   ```

   The boot process may not complete but hang at some point. This is all right for this exercise and actually an excellent example of a situation where one would use the IPMI SOL feature for debugging in real life.
3. Exit the SOL connection using the following key combination: `~~.`

Additional Information

More details on the IPMI simulator can be found in: `~/OpenIPMI-2.0.33/lanserv/README.*`