# ECE566 Enterprise Storage Architecture
# Lab #3: HA, DR, Virtualization

Directions:

- This assignment will be completed in your groups. However, **every member of the group must be fully aware of every part of the assignment**. Further, while you can discuss concepts with other groups, actual steps and answers should not be shared between groups.
- The assignment will ask for short answers or screenshots; this material should be collected in a PDF file submitted via GradeScope. *Word documents will not be accepted*. Anything you need to include in this document is highlighted in cyan.

# 1 High availability [50pts]

You have a storage system that can handle disk failures. That's great. What about network failures?

Each of your group servers have at least two NIC interfaces, but high availability takes more effort than just plugging both in. If one NIC fails, how does the client application find the other?

Routing around network failures usually happens at Layer 2 or 3 of the OSI model, the data-link and network layers. Fault-tolerance is also commonly included at Layers 4 and 7 (transport layer, application layer).

For this part of the homework, you will implement fault-tolerance at Layer 2. NICs bonded at this level are commonly called a LAG (Link Aggregation Group). To do this, you will create an interface that bonds two NICs together using LACP, an IEEE specification for link aggregation.

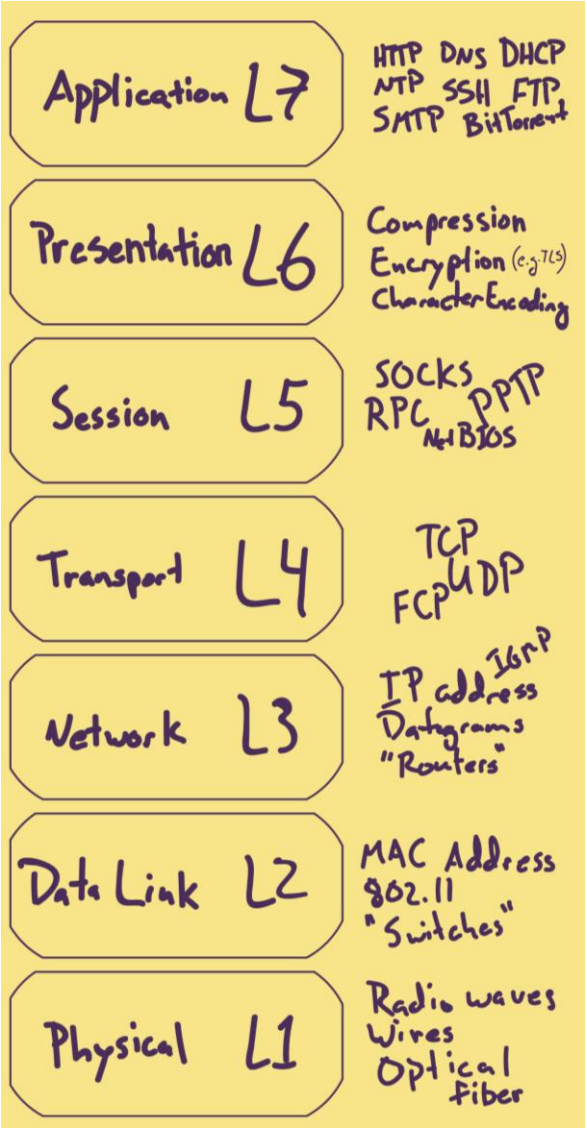

Illustration 1: OSI model layer names and examples

## 1.1 Verify remote access

~~Before you do this, **verify remote access** through the BMC (IPMI, iDRAC, iLo, etc). You will modify the server's active NIC, which will disconnect active SSH sessions. Include a screenshot showing that you have terminal access through your BMC (*not* SSH).~~

NOTE: Most students aren't using remote access as described above due to the number of Java security rules you need to mess with. As such, going to the datacenter will be needed if you make a mistake and lose your network connection.

## 1.2 Record MAC addresses

When you create the bonded interface, the MAC addresses for the NICs may change. Before making any changes, take a screenshot of the MAC address of all your network interfaces. The output of `ip link` includes the current MAC address of each network interface.

Why might the MAC address be affected by creating the bonded interface?

## 1.3 Cable the server for multi-path network access

Refer to the "Cabling" tab of the inventory spreadsheet. The Gb1 and mgmt links should already be cabled – double check this – and you should just have to add the cable for the redundant link to port Gb2 on your server. Refer to the spreadsheet for where the other end of the cable goes and connect it accordingly. Spare Ethernet cables provided in the rack for this purpose.

## 1.4 Create a bonded network interface

Ubuntu and other Debian-based Linux distributions store interface configuration in `/etc/network/interfaces`. Create a new bonded interface called `bond0`. This page should guide you through the process. We'll be doing plain active/passive bonding without LACP.

Note: The DHCP server is just configured with the MAC of Gb1. To ensure DHCP gives you an IP address no matter which interface is up, make sure that the MAC address inherited by the bond0 device is that of your Gb1 port (`eno1`). The easiest way to do this is to explicitly set the MAC address of the bond with the `hwaddress` directive. See here for more. (Alternately, we could have IT add both MACs to the IP address reservation, but this is easier.)

When you are done, reload the configuration changes and re-initialize your network interfaces by running `sudo systemctl restart networking`. If your lights-out access isn't working and you made a mistake in the config changes, this is the point where you will have to go visit the datacenter, plug it a keyboard and monitor, and fix your mistake in the noisy, hot/cold server room.

Include screenshots of the contents of /etc/network/interfaces and /proc/net/bonding/bond0 to show that you've successfully created bond0, and also show the full output of ifconfig.

## 1.5 Record MAC addresses (again)

After creating the bonded interface, look at the MAC addresses of `bond0`, `eno1`, and `eno2`. Did they change? Include a screenshot.

Does your new bonded interface have an IP address? Show the results of `ip addr show bond0`. If your server doesn't acquire an IP address, ensure that it has the MAC address of Gb1 as recorded in the inventory spreadsheet – this is what gets your server the IP you expect. MAC addresses that are unknown to IT do not receive IP addresses via DHCP.

## 1.6 Live-fire exercise

When your server is connected to the internet through the bonded interface, prove that it is not affected by a failure in one of the NICs. To do this, you'll need:

1. To be in the datacenter, physically next to your server
2. A laptop, SSHed into your server (not via the BMC, but through the usual IP address)

3. Something to record a video of your group performing the exercise.

On the back of the server, you should see three network cables plugged in. Find the one corresponding to your lights-out remote access. Have you found it? Good. Now ignore it. We're only concerned about the other two.

You should have two cables plugged into the two switches at the top of the rack. If you trace the cable, one should be connected to one switch and the other should be connected to the other switch. If the switches have independent uplinks to the internet, this means this network topology is resilient to failures in (1) the physical NIC itself, (2) breaks/shorts in the connector at the NIC, (3) breaks/shorts in the cable, (4) breaks/shorts in the connector *at the switch*, (5) switch failure, and (6) uplink failure.

Now we're ready for the test.

1. SSH into the server from your laptop
2. Start downloading a large file
   (e.g., `wget https://releases.ubuntu.com/22.04.4/ubuntu-22.04.4-desktop-amd64.iso` or similar)
3. Have one group member keep an eye on the SSH session during the next step
4. Record a video showing the following (running commentary appreciated):
   a. Show that you are SSHed in and downloading a large file
   b. Show which ports your server is connected to
   c. Disconnect one of your NICs from the network
   d. Show that the SSH session is still live, and that the large file is still being downloaded
5. Plug it back in when you're done
6. Attach the video to the assignment in Canvas, along with your PDF report (if you have issues uploading the video to Canvas, you can upload it to YouTube or another video host and include a link in your report)

Was there any jitter or delay in the SSH session or download when the plug was pulled? How was the download speed affected? What sort of side effects do you anticipate to network failover of this kind in a busy production environment?

# 2 Backups with rsnapshot [50pts]

Let's use our storage controller to build an automated backup system. The thing we'll be backing up is a VCM machine. You can make a new VCM virtual machine running Linux, or use an existing one. We'll refer to this machine as the **backup client**, and we'll pull the content of the home directory on it.

You can use this guide[1], and note the following:

- **Backup source**: We're going to back up your user home directory on the backup client machine. This is usually /home/<USERNAME>, where <USERNAME> is whatever user you created when you installed Linux. (If you somehow installed Linux without creating a non-root user, make a user account now, login as the user, and put some stuff in its home directory.)
- **Backup destination:** We need to use a native Linux filesystem for this, because rsnapshot depends on hard links. Therefore, prepare a RAID (your choice of hardware vs software, RAID level, etc.) using your server's hard disk drives and create and ext4 filesystem on it mounted at /backup. We will put our backups there.
- **Frequency:** Set up nightly and weekly intervals.
- **Retention**: You should retain seven nightly backups and four weekly backups.
- **Automation:** You should configure cron-based automation. However, for testing, you can just run "rsnapshot nightly" to perform the backup that would happen nightly, and "rsnapshot weekly" to perform the backup that would happen weekly. You can run these commands as often as you wish; there's nothing in the software that actually needs the backups to be done nightly/weekly as opposed to every few dozen seconds (i.e., rsnapshot does not actually care that they're called "nightly" or "weekly").

Provide screenshots or a terminal log of these steps in your writeup.

Provide the relevant portions of rsnapshot.conf and your cron configuration.

Let's test it. Open one terminal window as root, and another terminal window as the non-root user. Do the following, **and for each step, show the process via screenshots or terminal logs. Label or otherwise identify each step you're doing in your screenshot/log.**

1. As the user on the client, add some content to the user home directory.
2. As root on the backup server, do several nightly and weekly backups to populate your backups.
    - Observe how the hard-link count changes for files as you take more and more backups.
    - Check the disk free space on the root partition as you take backups; the storage increase should be very small (just metadata).
3. As the user, delete an important file.
4. As root, take another few nightly backups to simulate time passing.
5. Copy the appropriate backup from the backup server to the client's home directory, thus restoring the damaged file.

[1] The installation step is distro-specific, but everything else is distro-neutral.