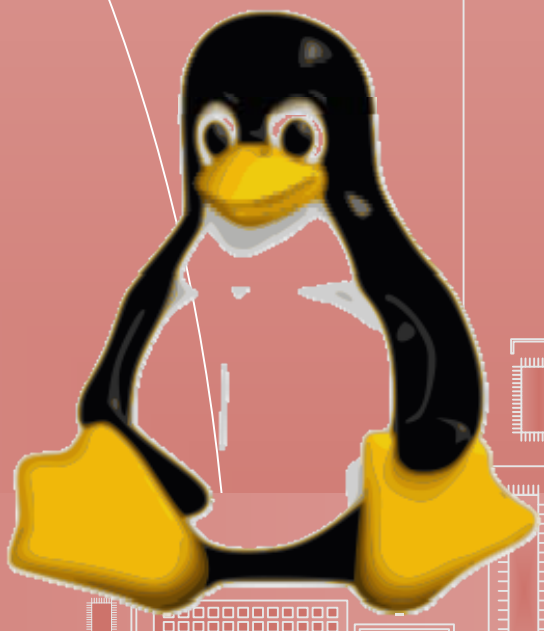
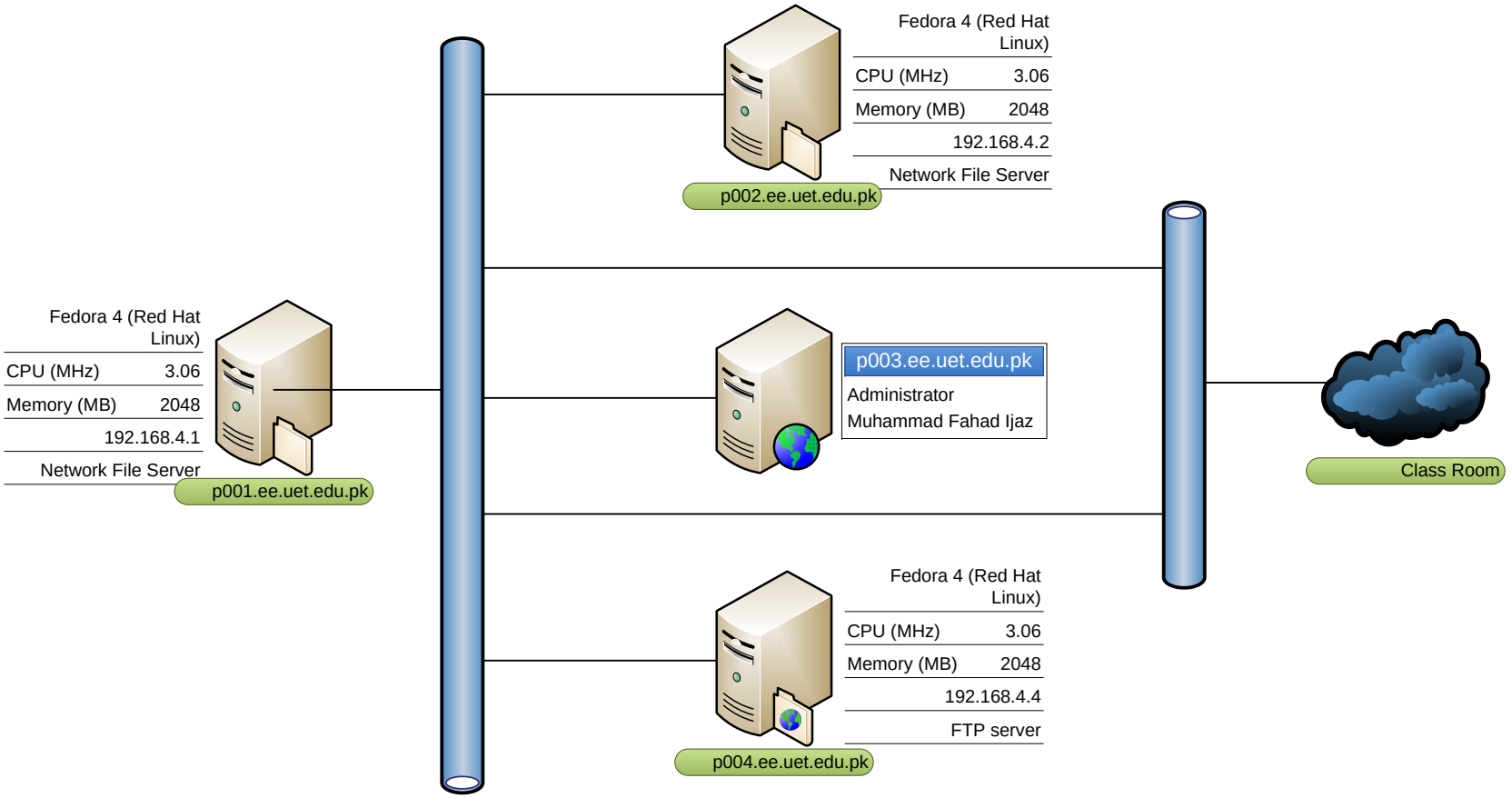
Computer Communication Systems Laboratory

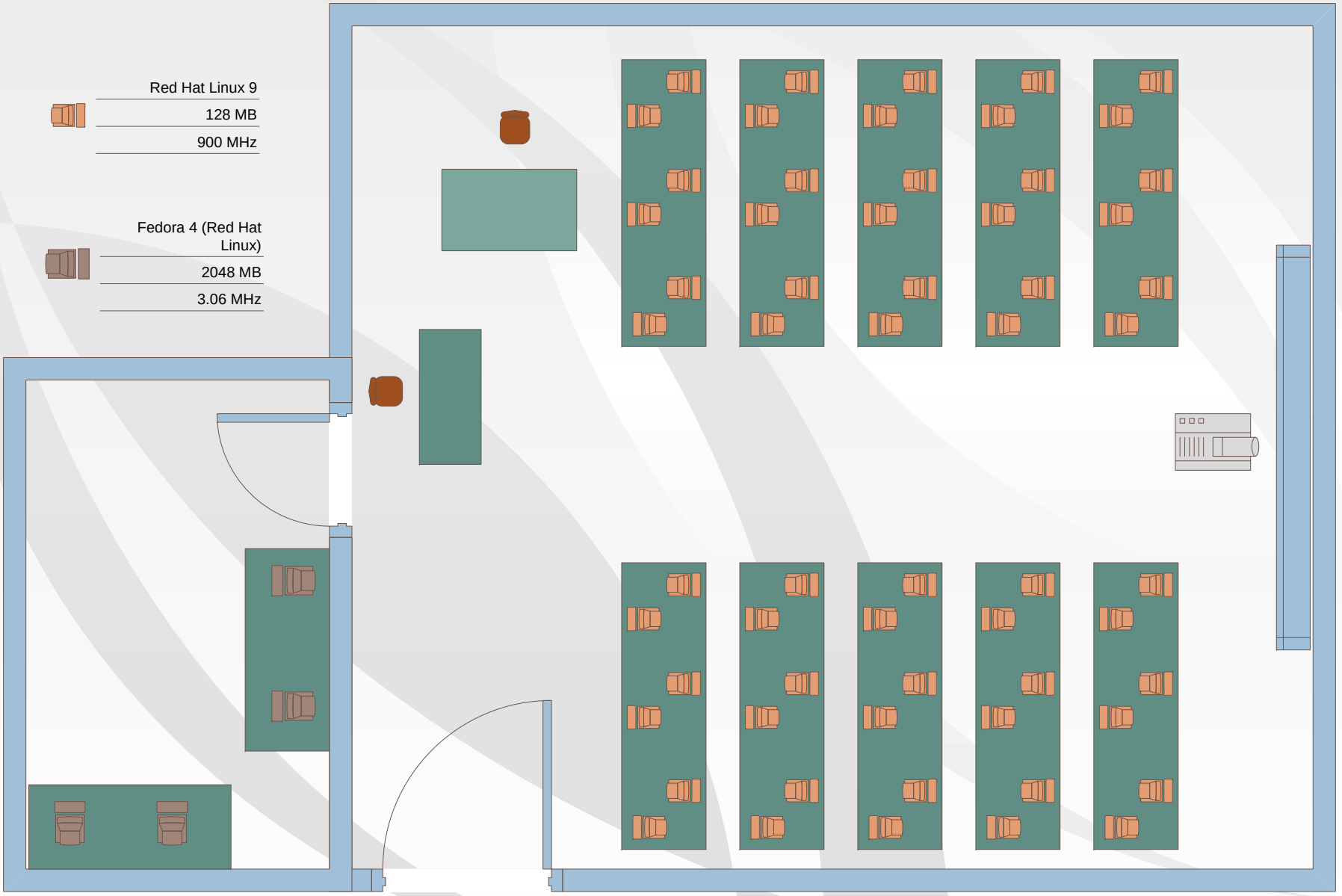


Computer Communication

Systems Laboratory
Electrical Engineering Department
UET, Lahore



Computer Communication Lab., Electrical Engineering Department , UET., Lahore. Lab contains a NFS server, DNS server, Web server, SQL server (not shown)
And a FTP server (for installation purpose).


Red Hat Linux 9
128 MB
900 MHz
Fedora 4 (Red Hat Linux)
2048 MB
3.06 MHz

Venue

The Computer Communication Laboratory is located on the first floor of Electrical Engineering Department, University of Engineering & Technology, Lahore. The associated laboratories are Computer Interfacing Laboratory and Computer Network Laboratory.

Lab Objective

Software tools are essential in engineering. Computer Aided desig has brought a revelution in engineering Design. The designing of these tools require proefficienty in programming lanuages like C and Java. These tools have introduced a new phase of desigining called simulation. The fruits of programming are not limited to simulation only but they have a vast impact on the society. Now people think in the form op programmming a dsp boad or FPGA,s rather than designing circuits. FPGA,s, ASICS, microcontrollers and dsp processors all require programming skills for the effivcient use of resources.

The objective of the commputer communication lab is to develop strong programming skills in java and C programming languages. This lab aims at providing indepth knowledge to its students by providing an enviorment suitable for development in programmong skills. The lab also provides envoirment to develop expertise in datastructures in C and Java.

Lab Staff

Kalid Mehmood
Lecture Assistant

Azhar Hussain
Lab. Attendant

Lab Equipment

Equipment	Type	Quantitiy
Computers	P4	4
	P3	64
AC	Split type	4
	Window Type	1
UPS		24
HUB		5
Switch		1
Projector		1
Amplifier		1
Speakers		4

Courses

Programming in C

Data Structures in C

Experiments: Programming in C

- Introduction to computers, Operating System and Programming Languages
- Introduction to Compiler, Linker, Loader and IDE of Turbo C
- Printing programs with printf(), concept of data types and formatted output
- Concept of scanf(), if-else, switch
- For loop, while loop, do-while loop
- Nested loops, conditions in nested loops
- Functions and return types
- Arrays
- Pointers
- Introduction to Keil µVISION2
- Basic Port I/O of 8051
- Introduction to Timers of 8051
- Introduction to Serial Port of 8051
- Introduction to Interrupts of 8051

The Development of the Laboratory

Installing Linux on a PC has been long considered a programming guru's domain. It usually takes a novice user weeks or even longer to get the system properly configured. However, with emerging installation techniques and package management, especially from Red Hat, Linux is on the verge of becoming user friendly. Yet, even with these newer methods, one aspect of Linux that is still frustrating is installation on a large scale.

Installation Automation

Imagine a system engineer who has to set up a new Linux network with a large number of machines. Now, the same issues need to be addressed and the same questions answered repeatedly. This makes the task very inefficient. Hence, a need arises to automate this parameter and option selection.

Giving Kickstart a Kick!

The Rom boot technique was adopted. The Red Hat Installation disk was merged with a very fine net-booting package, etherboot, to obtain a network-bootable image of the disk. Now, since this image was also placed on a NFS server, only a 16KB loader was needed on the floppy which would boot up in under twenty seconds. This loader would then retrieve the actual image over the network.

Setting up a Kickstart Option File

The method devised was to first install Red Hat Linux 6.1 on a machine using the “normal” CD-ROM method. All packages, options and settings for our to-be-target machine were manually specified. Once the system was up and running, it was tested for optimum performance and then used as prototype for the rest of the installations. A special package called mkkickstart also had to be installed. The mkkickstart utility can extract information from an installation and print it on the standard output. This was exactly done: mkkickstart >ks.cfg. Any Kickstart installation that is now run with ks.cfg as the configuration file will create a replica of our prototype workstation. We did some minor editing of this file to implement some changes.

Post-Install and Customization

The Kickstart technique offers provisions for executing any necessary post-install procedures needed once the installation is complete. This feature, besides allowing individual customization, is particularly useful when packages other than those included with the standard Red Hat distribution are to be installed. In our case, these included JDK (Java Development Kit) for Linux, among many others. These lines were added to the following lines to the post-install section and a separate script and Perl program was created that would execute when the Red Hat installation had finished.

Setting up the Network Boot

One question that remained was where and how to place the ks.cfg file so that the target system was able to receive it even after it had undergone a DHCP/TFTP boot. An analysis of the installation procedure revealed that the tmp directory within the initial ramdisk is one of the locations that the Kickstart system looks for a configuration file. The procedure of copying each ks.cfg to the appropriate location and then adding a kernel to initrd to make an encapsulated chunk of code was all performed by a script called superkick.

Another script was written, doitfor, to automatically customize the ks.cfg file and a post-install file for every workstation. The major task that this script performed was inserting a specific host name and IP address within each ks.cfg using the streplace utility. This script takes as input the host name and IP address and generates a boot image to be uploaded using DHCP/TFTP boot.

DHCPD Configuration

To get the installation running, either a DHCPD server or a BOOTP server was needed to be set up. DHCPD was selected because of some advantages. There is a long list of DHCPD options but basic ones were required to be configure, namely the default name-server address, starting IP and domain names. Once the DHCPD had been configured to offer the desired IP address for a target, we could proceed in two ways. The first was to burn the ROM image into an EPROM and plug the EPROM in to the network card, create a bootable floppy disk carrying the ROM image and thus get the installation running. Some initial installations were carried out with the floppy method. Later, the availability of an EPROM writer allowed to employ the EPROM technique, which worked fine.

Note that completing more than two or three installations at a time overloads the network and brings down the efficiency. With two machines installing concurrently, we were able to achieve complete installations in an average time of fifteen minutes.

Time Synchronization

With files shared among a large number of workstations, it becomes imperative that machines have their clocks synchronized so that file time stamps are globally comparable. The time of all the machines was setup to match their time to that of a reference server at every startup by utilizing the rdate utility.

Startup/Switch-off Automation

A Perl program named switch that switches one or all machines on the LAN either on or off, generates a Magic Packet and broadcasts it over the network. The target machine, on receiving this packet, switches on. To switch a machine off, a straightforward remote shell invocation of halt or shutdown -h suffices. To make things more manageable, the script parse /etc/switch.conf was being made for information regarding which host on the LAN has what MAC address.