EECS 211 Spring 2016
Description of Major Project for the Term

The remaining programming assignments will build an application that simulates the passing of messages around the internet. The processes we will simulate are gross simplifications of the processes and structures in the real internet so that the project can be feasible to do within the quarter time frame. Nonetheless, the sequence of programs should give you a fair picture of:

- The information that has to be in a message passed from one computer to another computer in a network.
- The methods used by the internet to transfer messages along a path from the source to the destination.

In addition, I have organized the sequence of assignments to illustrate several basic principles of software design and development. Of course, I have done almost all the design, but as each new assignment is given and also at the end of the project you should reflect on how each assignment adds a logically "next" piece to the overall project. Software development principles illustrated in this project include:

- top-down design and development;
- the utility of a good test harness at the beginning of the development;
- the benefits of modularization;
- the use of classes and objects to model a domain of interest.

Of course, the assignments give you practice in basic C++ programming techniques such as how to implement classes, how to use dynamic memory allocation, how to arrange data in basic data structures, how to process non-numeric data, and many more.

Background

The Internet is a large collection of computers. Each computer is "connected" to one or more (but, of course, not all) other computers in the network. The connection can be physical (e.g., those purple internet cables) or wireless. Computers can enter the network (become connected to at least one other computer, e.g. you plug the purple cable into your laptop or turn your wireless on) and can leave the network (you unplug or turn your wireless off). The network is, therefore, a constantly changing collection of loosely connected computers. This is illustrated in Figure 1.

The Internet contains computers of many different kinds. Examples of the different kinds of computers that can exist in the internet include: personal computers and laptops, servers (special machines for hosting web sites), local-area network servers (special machines for managing a small group of computers in relatively close physical proximity), domain-name servers (DNS – special machines that relate web site names to their physical addresses in the network), and many others. Each type of computer has its special features and capabilities.

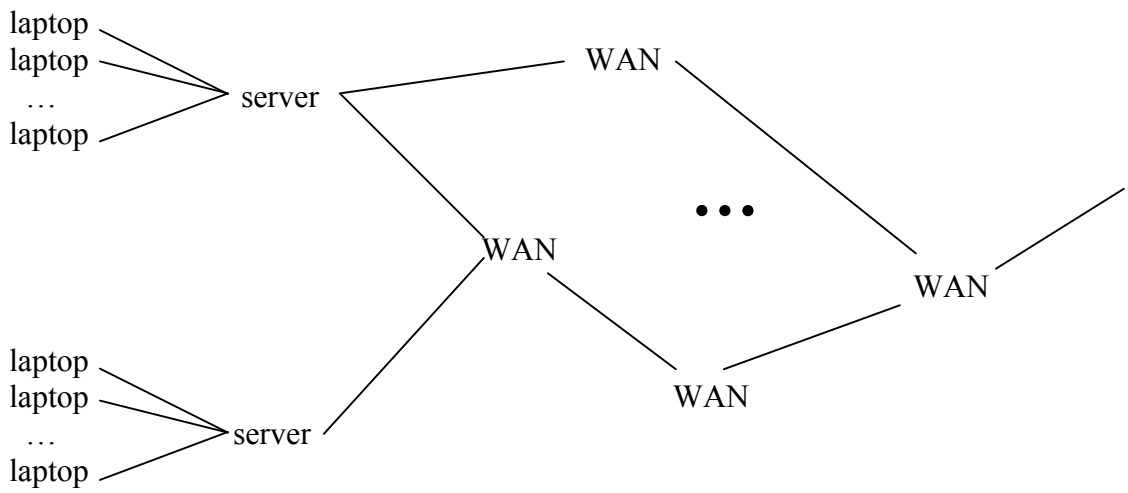

Figure 1. Sample Network of Machines

In our project we will simulate three kinds of machines – laptops (which will be able to generate and receive messages), local-area network servers (which connect to a small group of laptops), and wide-area network machines (which connect groups of local networks together).

When you send email to a friend or download a video or log in to Canvas, your computer sends messages to the other computer and *vice versa*. Because your computer may not be (and, in fact, almost surely is NOT) connected directly to the other machine, the message has to contain information about the computer that is supposed to receive the message. Your message makes several "hops" along a route that eventually (hopefully) ends at the target computer. Because the packet that is passed along contains the address of the target computer, each machine along the path of hops will know where the message is supposed to go. If the target machine needs to reply, it needs to know what computer sent the message. So, message packets also contain the address of the computer that generated the message. So, in our project, we will simulate these packets with a class, called **datagram**, that contains the address of the sender, the address of the target computer, a message (in our case, a character string), and the size of the message. In the real Internet, packets contain much more information, but this is all we will use in our project.

The Internet has sophisticated algorithms for determining how to route a message from one computer to another. The algorithms take into account the organization of the machines in the network (how they are connected), the current traffic in the network (e.g., which links currently have a lot of messages), and many other aspects. It is interesting to note that a large message sent from one computer to another will likely be partitioned into smaller packets, and the individual packets may not even take the same path to the destination. In our project we will implement a very simple routing algorithm.

Modeling the system

We will build a simulator for computers connected in a network that can generate datagrams and pass them from one machine in the network to other machines. The system we will implement is a gross simplification of the real Internet but will include many of the features found in the Internet, such as:

- messages;
- multiple computers connected into a network;
- computers in a network communicating with computers to which they are directly connected;
- messages traveling along a path from one computer to another computer.

Sequence of Programs

We will develop our simple simulator in a series of steps. Here is the tentative sequence.

- Program 3 – **IPAddress** class, **datagram** class. We begin with two simple classes representing IP addresses (the format used in the internet to specify the addresses of computers) and a simple model of a message packet that might be sent from one computer to another.
- Program 4 – A simple command line parser. A fundamental part of the software development task is to exercise the classes and code that are being developed to uncover errors, i.e., to test the software. To facilitate testing and to provide a means for using the simulator when the project is finished, we will implement a simple command line parser. This function will accept a line of text and break it into a list of tokens. The first token should represent a command, such as create_machine, destroy_machine or datagram. The remaining tokens represent additional information needed to execute the command. By the end of the project we will be parsing and simulating commands like

  create_machine *computer_type computer_ name IP_address*
  datagram 2.96.255.3 27.0.0.254 "Dear Mom and Dad"
- Program 5 – Command recognizer and main switch. This assignment will add functions providing the means to store command strings and look them up. It will also create a main switch with a case for each command. When this assignment is completed, your project will be able to recognize every command that will be used in the project and branch to a corresponding case for each command. Later assignments, then, will fill in those cases with the functions needed to actually simulate the commands.
- Program 6 – This assignment will add classes for basic computer machine and three special kinds of computers – laptops, local-area network servers, and wide-area network servers. We will also provide a storage mechanism for storing several machines. (In the next program we will connect them into a network.) Machines can be added or removed from the list at any time, simulating for example the process of a person turning his or her laptop on and off.

- Program 7 – In this assignment we will connect machines into a network. We will also add the capability of laptops to generate datagrams and machines to store several datagrams.
- Program 8 – Finally, we implement the passing of a datagram through the simulated network from the computer that generated it to the computer that is supposed to receive it.