**Лекция 1**

**Основы сетевой концепции**

Network programming is no longer the province of a few specialists. It has become a core part of every developer’s toolbox. Today, more programs are network aware than aren’t. Besides classic applications like email, web browsers, and remote login, most major applications have some level of networking built in. For example:

* Text editors like BBEdit save and open files directly from FTP servers.
* IDEs like Eclipse and IntelliJ IDEA communicate with source code repositories like GitHub and Sourceforge.
* Word processors like Microsoft Word open files from URLs.
* Antivirus programs like Norton AntiVirus check for new virus definitions by connecting to the vendor’s website every time the computer is started.
* Music players like Winamp and iTunes upload CD track lengths to CDDB and download the corresponding track titles.
* Gamers playing multiplayer first-person shooters like Halo gleefully frag each other in real time.
* Supermarket cash registers running IBM SurePOS ACE communicate with their store’s server in real time with each transaction. The server uploads its daily receipts to the chain’s central computers each night.
* Schedule applications like Microsoft Outlook automatically synchronize calendars among employees in a company.

Java was the first programming language designed from the ground up for network applications. Java was originally aimed at proprietary cable television networks rather than the Internet, but it’s always had the network foremost in mind. One of the first two real Java applications was a web browser. As the Internet continues to grow, Java is uniquely suited to build the next generation of network applications.

**Networks**

A *network* is a collection of computers and other devices that can send data to and receive data from one another, more or less in real time. A network is often connected by wires, and the bits of data are turned into electromagnetic waves that move through the wires. However, wireless networks transmit data using radio waves; and most longdistance transmissions are now carried over fiber-optic cables that send light waves through glass filaments. There’s nothing sacred about any particular physical medium for the transmission of data. Theoretically, data could be transmitted by coal-powered computers that send smoke signals to one another. The response time (and environmental impact) of such a network would be rather poor.

Each machine on a network is called a *node*. Most nodes are computers, but printers, routers, bridges, gateways, dumb terminals, and Coca-Cola machines can also be nodes. You might use Java to interface with a Coke machine, but otherwise you’ll mostly talk to other computers. Nodes that are fully functional computers are also called *hosts*. I will use the word *node* to refer to any device on the network, and the word *host* to refer to a node that is a general-purpose computer.

Every network node has an *address*, a sequence of bytes that uniquely identifies it. You can think of this group of bytes as a number, but in general the number of bytes in an address or the ordering of those bytes (big endian or little endian) is not guaranteed to match any primitive numeric data type in Java. The more bytes there are in each address, the more addresses there are available and the more devices that can be connected to the network simultaneously.

Addresses are assigned differently on different kinds of networks. Ethernet addresses are attached to the physical Ethernet hardware. Manufacturers of Ethernet hardware use preassigned manufacturer codes to make sure there are no conflicts between the addresses in their hardware and the addresses of other manufacturers’ hardware. Each manufacturer is responsible for making sure it doesn’t ship two Ethernet cards with the same address. Internet addresses are normally assigned to a computer by the organization that is responsible for it. However, the addresses that an organization is allowed

to choose for its computers are assigned by the organization’s Internet service provider (ISP). ISPs get their IP addresses from one of four regional Internet registries (the registry for North America is ARIN, the American Registry for Internet Numbers), which are in turn assigned IP addresses by the Internet Corporation for Assigned Names and Numbers (ICANN).

On some kinds of networks, nodes also have text names that help human beings identify them such as “www.elharo.com” or “Beth Harold’s Computer.” At a set moment in time, a particular name normally refers to exactly one address. However, names are not locked to addresses. Names can change while addresses stay the same; likewise, addresses can change while the names stay the same. One address can have several names and one name can refer to several different addresses.

All modern computer networks are *packet-switched* networks: data traveling on the network is broken into chunks called *packets* and each packet is handled separately.

Each packet contains information about who sent it and where it’s going. The most important advantage of breaking data into individually addressed packets is that packets from many ongoing exchanges can travel on one wire, which makes it much cheaper to build a network: many computers can share the same wire without interfering. (In contrast, when you make a local telephone call within the same exchange on a traditional phone line, you have essentially reserved a wire from your phone to the phone of the person you’re calling. When all the wires are in use, as sometimes happens during a major emergency or holiday, not everyone who picks up a phone will get a dial tone. If you stay on the line, you’ll eventually get a dial tone when a line becomes free. In some

countries with worse phone service than the United States, it’s not uncommon to have to wait half an hour or more for a dial tone.) Another advantage of packets is that checksums can be used to detect whether a packet was damaged in transit.

We’re still missing one important piece: some notion of what computers need to say to pass data back and forth. A *protocol* is a precise set of rules defining how computers communicate: the format of addresses, how data is split into packets, and so on. There are many different protocols defining different aspects of network communication. For example, the Hypertext Transfer Protocol (HTTP) defines how web browsers and servers communicate; at the other end of the spectrum, the IEEE 802.3 standard defines a protocol for how bits are encoded as electrical signals on a particular type of wire.

Open, published protocol standards allow software and equipment from different vendors to communicate with one another. A web server doesn’t care whether the client is a Unix workstation, an Android phone, or an iPad, because all clients speak the same HTTP protocol regardless of platform.

**The Layers of a Network**

Sending data across a network is a complex operation that must be carefully tuned to the physical characteristics of the network as well as the logical character of the data being sent. Software that sends data across a network must understand how to avoid collisions between packets, convert digital data to analog signals, detect and correct errors, route packets from one host to another, and more. The process is further complicated when the requirement to support multiple operating systems and heterogeneous network cabling is added.

To hide most of this complexity from the application developer and end user, the different aspects of network communication are separated into multiple layers. Each layer represents a different level of abstraction between the physical hardware (i.e., the wires and electricity) and the information being transmitted. In theory, each layer only talks to the layers immediately above and immediately below it. Separating the network into layers lets you modify or even replace the software in one layer without affecting the others, as long as the interfaces between the layers stay the same.

Figure 1-1 shows a stack of possible protocols that may exist in your network. While the middle layer protocols are fairly consistent across most of the Internet today, the top and the bottom vary a lot. Some hosts use Ethernet; some use WiFi; some use PPP; some use something else. Similarly, what’s on the top of the stack will depend completely on which programs a host is running. The key is that from the top of the stack, it doesn’t really matter what’s on the bottom and vice versa. The layer model decouples the application protocols (the main subject of this book) from the physics of the network hardware and the topology of the network connections.



There are several different layer models, each organized to fit the needs of a particular kind of network. This book uses the standard TCP/IP four-layer model appropriate for the Internet, shown in Figure 1-2. In this model, applications like Firefox and Warcraft run in the application layer and talk only to the transport layer. The transport layer talks only to the application layer and the Internet layer. The Internet layer in turn talks only to the host-to-network layer and the transport layer, never directly to the application layer. The host-to-network layer moves the data across the wires, fiber-optic cables, or other medium to the host-to-network layer on the remote system, which then moves

the data up the layers to the application on the remote system.

