Software Projects for ECE:544 (Spring 2006)

[Back to course homepage]

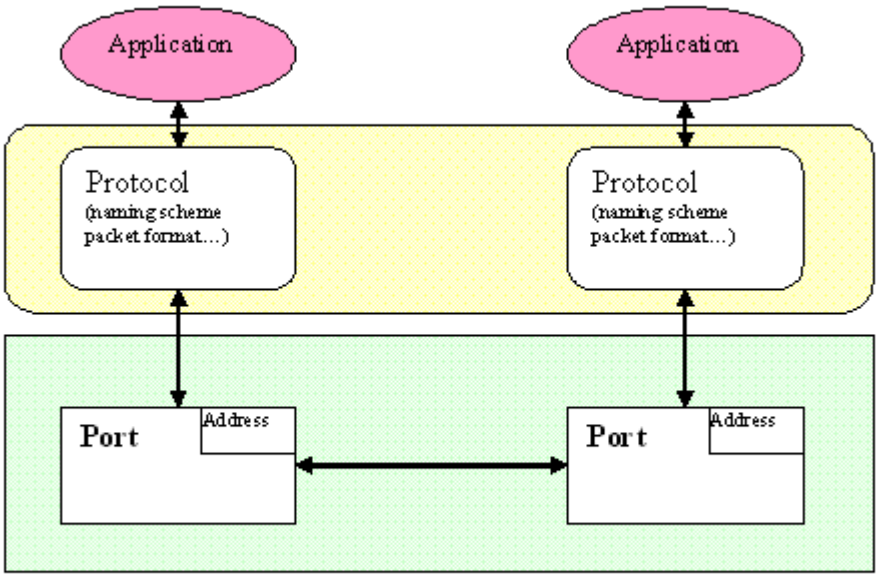
Objective & Approach

The software project of this class provides a unique chance to practice writing basic communication protocols and test your protocol in real environment.

The students are suggested to write and compile C++ programs in Linux OS. **Note that the students are supposed to focus most on the protocol functions and primitives instead of learning how to write TCP/IP applications on Unix/Linux**. Therefore, some predefined classes are provided to abstract the concept of sockets and avoid students to tackle with Unix socket programming directly.

A platform is provided to emulate a point-to-point communication link by abstracting the real physical connection and Unix communication sockets in the form of "*Port*". Your program's task is to configure the *Ports* appropriately and design protocol signaling/functions upon them.

Software Architecture



The following classes are already designed:

- Packet
- Address
- Port
    - SendingPort
    - ReceivingPort

Documents for the interface/functions of those common classes are here.

Prerequisite

1. basic C++ knowledge/skill

- class, objects, inherency...
- command-line I/O, file I/O
- multi-thread programming
- g++ compiler

2. familiar with Linux OS and basic commands

List of Projects

Detailed descriptions for each software project

- Example 1: Simple packet sender and receiver (Download example1 package)
- Example 2: Coming Soon.
- Project 1: Fragmentation and Assembly
- Project 2: ARQ Scheme
- Project 3: Router

Program Coding and Compile Instructions

You can write your code in any Windows/Unix/Linux text editors. Try to put useful comments with your C++ code.

The program needs to be compiled in Linux. When the functions of common classes are used, you need to include the common.cpp in your g++ command:
For example, on any Linux OS:

```
g++ common.cpp sender.cpp -o sender
g++ common.cpp receiver.cpp -o receiver
```

Mailing List & Project Submissions

For all matters related to this class and especially the project, send email to comnet2@winlab.rutgers.edu. Please do not send email to the instructors if you know that the answer would be useful to others.

To subscribe to the mailing list, first send a message to mailservice@winlab.rutgers.edu with the following in the body (not subject): subscribe comnet2 you@somewhere.edu

Projects are submitted by sending a message to zhibinwu@winlab.rutgers.edu with your name (and that of all group members) in the body, and all the results in a single attachment . Please follow the instructions in the project description on how to prepare the attachment.

References

- To learn about sockets and network programming, Beej's Guide helps.
- Linux Introduction

Example 1

[Back to Software Project homepage]

Code

Sender.cpp

```
#include "common.h"
#include <iostream>
using namespace std;

class Packet;
class Port;
class Address;
class SendingPort;

int main(int argc, const char * argv[])
{

try {
  const char* hname = "localhost";
  Address * my_addr = new Address(hname, 3000);
  Address * dst_addr =  new Address(argv[1], (short)(atoi(argv[2])));
  SendingPort *my_port = new SendingPort();
  my_port->setAddress(my_addr);
  my_port->setRemoteAddress(dst_addr);
  my_port->init();

  Packet * my_first_packet = new Packet();
  my_first_packet->fillPayload(10,"this is just a test");
  my_port->sendPacket(my_first_packet);
  cout << "packet is sent!" <<endl;

} catch (const char *reason ) {
    cerr << "Exception:" << reason << endl;
    exit(-1);
  }

 return 0;
}
```

Receiver.cpp

```
#include "common.h"
#include <iostream>
using namespace std;

class Packet;
class Port;
class Address;
class ReceiverPort;

int main(int argc, const char * argv[])
{

try {
  const char* hname = "localhost";
  Address * my_addr = new Address(hname, (short)(atoi(argv[1])));
  ReceivingPort *my_port = new ReceivingPort();
  my_port->setAddress(my_addr);
  my_port->init();

  cout << "begin receiving..." <<endl;
  Packet *p = my_port->receivePacket();
  /**
  * Post-processing received packet
  */
  cout << "receiving a packet:" << endl;
  cout << p->getPayload() << endl;
} catch (const char *reason ) {
    cerr << "Exception:" << reason << endl;
    exit(-1);
}

return 0;
}
```

The code just configure port with addresses and then initialize it. After that, a packet is sent.

Command-line arguments

sender program takes two arguments, destination hostname and destination port number.
receiver program takes only one argument, local port number of the receiving port.

To have a communication link, the sending port's destination port number must be same as the receiving port's local port number.

Compile

```
g++ common.cpp sender.cpp -o sender
g++ common.cpp receiver.cpp
```

Test

```
./receiver 4000
./sender localhost 4000
```

Tip: How to Set up communication links by configuring ports

Each port (sending port and receiving port) could have two addresses. One is the local address, the other is the remote address. The remote address corresponds to the local address of the "port" at the other end of communication link.

To configure a sending port, both addresses need to be configured. But to configure a receiving port, only the local address is necessary. Because it is as long as sending port configure the destination correctly, the receiving port could remains "dumb" and needn't to be aware of the remote addresses.

In the above example, we have successfully set up a simplex (not duplex) link from the sending port to the receiving port. Note that the reverse link does not exist!