CSE536 Project 2 – Build a communication channel that uses normal read/write system calls to communicate over IP.

Project 2 is a more challenging project since I will only provide the specification for the outcome, not the steps for completion. In theory and in practice any Linux kernel module can interface with and use any other Linux kernel module as long as the interfacing module is designed to meet all of the existing module's requirements. In this case our cse5361 module will be modified to communicate with cse5361 modules on other computers by interfacing with standard IPv4. Internet Protocol (IP) normally operates in the standard internet network stack and is able to interface with a variety of transport layer modules as well as link layer modules. Typical transport layer modules include TCP and UDP. A typical link layer module would be Ethernet. In the current world of network technology IPv4 is the internet glue and the lower layer modules interface IPv4 with local hardware while the upper layer modules provide application programs access to IPv4.

Your mission is to replace (or at least bury) the standard socket interface used by most network programs and allow cse5361app to communicate in full duplex with another cse5361app at an arbitrary network end location via ipv4 using standard read/write system calls. Ipv4 is not to be altered. A write on system A should fill a read buffer on system B and visa-versa. The calls can be blocking or non-blocking. You are to determine if this communication interface can be implemented and if so implement it. If is not possible to provide the interface then report why it is not in detail.

This is a class community effort and I expect the discussion board to be heavily used. Open source projects are often started in this manner. Multiple, even novel solutions are welcome. Leading contributors will receive extra credit at my discretion.

Some additional implementation detail and hints:

- The cse536app buffers used to exchange data should have a standard size of 257 bytes.
- The first byte of each write buffer designates the type of the remaining buffer.
- Types include 1-destination IP number in record, Type 2-data. Others may be defined as required.
- Source code for IPv4 is located in TLSD/net/ipv4.
- Hint: ipv4 expects a protocol communicating with it to provide socket data structures. You will need to construct what ipv4 requires in your cse5361 module.
- Hint: ipv4 requires a protocol to register with it using a unique protocol type number so that received buffers can be sent to the right place.
- Hint: See http://www.6test.edu.cn/~lujx/linux_networking/0131777203_ch14lev1sec2.html, http://www.ibm.com/developerworks/library/l-linux-networking-stack/, and http://www.cs.unh.edu/cnrg/people/gherrin/linux-net.html for a discussion of ip implementation. Much of the details can be ignored since you only want to submit packets and have them forwarded to your protocol.
- It may end-up being easier to start at UDP instead of IP but let's try IP first.

Please complete your project by providing the usual screenshots and discussion. In addition you must submit all of your source code and compiled modules and programs. You must do your own program coding but you may freely communicate with class members to determine what to do. You will also need to integrate project 2 into your formal project report from project 1. A final formal project report will be expected at the end of the semester.