FIT3142 Laboratory #5: Client-Server Applications Using Socket Inter-Process Communications Part 1

Dr Carlo Kopp, SMIEEE, AFAIAA, FLSS, PEng
Faculty of IT, Clayton.
Email: `Carlo.Kopp@monash.edu`

August 29, 2016

Revision Status:

```
$Id: FIT3142-Laboratory-5.tex,v 1.3 2016/08/29 06:26:37 carlo Exp carlo $
```

Contents

1 Introduction

The objective of this weeks laboratory exercise is to gain some basic exposure to the BSD Socket networking API, using the C language. Being able to program with sockets is a useful skill if you intend to develop any networked client-server application, where you do not intend to use, or for other reasons cannot use higher level APIs.

It is strongly recommended that you prepare for the lab by reading through all of the reference materials and coding and testing the application before you attend the lab, because the time required to complete the lab tasks is only sufficient if you have actually prepared for the lab. Do not paste from example code and tutorials as you will not learn how the code actually works if you so so. If you attend the Laboratory without any preparation it is unlikely you will be able to complete the work on time.

The aim of this lab will be to familiarise you with the behaviour of sockets, using the Linux system and the C language.

It is important that you do this work independently. Collaboration, copying and plagiarism will attract zero marks in this subject, without exception.

Opening a socket connection requires the creation of the socket with a `socket()` system call, binding a socket address to the socket with a `bind()` call and initiating the connection with a `connect()` call.

The `socket()` call returns an index into the process file table, termed a *file descriptor* in `Unix/Linux/BSD`.

Once the connection is open, the programmer may use both socket specific calls or the established `Unix/Linux/BSD read()` and `write()` system calls.

The BSD Socket has become the defacto standard low level programming interface for networked IPC, although in most contemporary applications it is hidden below other protocols.

Nearly all operating systems will provide a BSD socket API, although some will use the newer POSIX standard.

2 Lab Task 1

The completion of the following steps will meet the objectives of this lab.

1. Read through the list of Socket tutorials in the **Notes** section and study how these applications are constructed.
2. Produce a very simple client / server application using BSD Socket calls, where the client accepts user keyboard input, and the server prints it to the screen;
3. Demonstrate your code, and then email your completed lab exercise to your lab demonstrator. The email subject must contain the unit code `FIT3142` and the week.
4. Marking criteria:
   (a) strings entered into the client are displayed by the server (30%)
   (b) client nicely terminates on some input (`Ctrl-D/EOF` preferred) (10%)
   (c) server detects client disconnection/termination (10%)
   (d) code is well commented (10%)
   (e) code is compiled with `-Wall` option and produces no warnings (10%)
   (f) return values of networking functions checked for errors (10%)
   (g) client can redirect input from `stdin` and `server` outputs correctly (20%)
5. **Save all of your program files as you will need them in a future lab.**

3 Notes

The programming task can be done directly on any `Linux`, `*BSD/MacOSX`, of `Unix` system.

If you do not have access to any of these platforms, several options are available:

1. The trial version of `Ubuntu Linux` is available (`http://www.ubuntu.com/download/help/try-ubuntu-before-you-install`);
2. You can install `CygWin` under MS Windows (`http://cygwin.com/install.html`);
3. An alternate approach is to configure an external USB hard disk as a bootable `Linux system` on your `MS Windows` system;

4. You can install `VMWare Workstation` and install any `Linux` or `*BSD` variant as a VM on your `MS Windows` system.

Please note that `Linux/Unix` skills are frequently required in many industry jobs. Most large web servers and distributed applications are not hosted on `MS Windows` platforms.

There are many online tutorials for BSD socket programming, with working examples of code. Also the `nttcp` application you will use in a future Laboratory is a socket application. Useful tutorials are:

1. Tutorials on Socket programming (Ch.11) and file I/O (Ch.3) `http://www.bitsinthewind.com/about-dac/publications/using-c-on-the-unix-system`;
2. Tutorials on Socket programming `http://beej.us/guide/bgnet/`

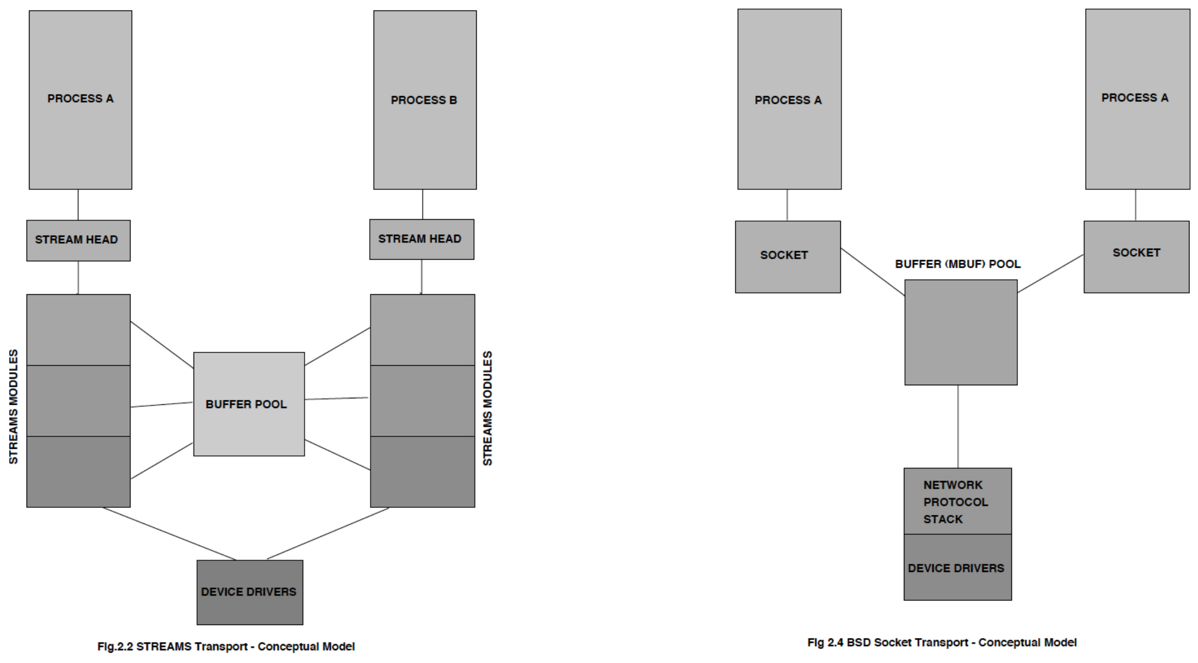

Figure 1: *STREAMs vs. Sockets.*

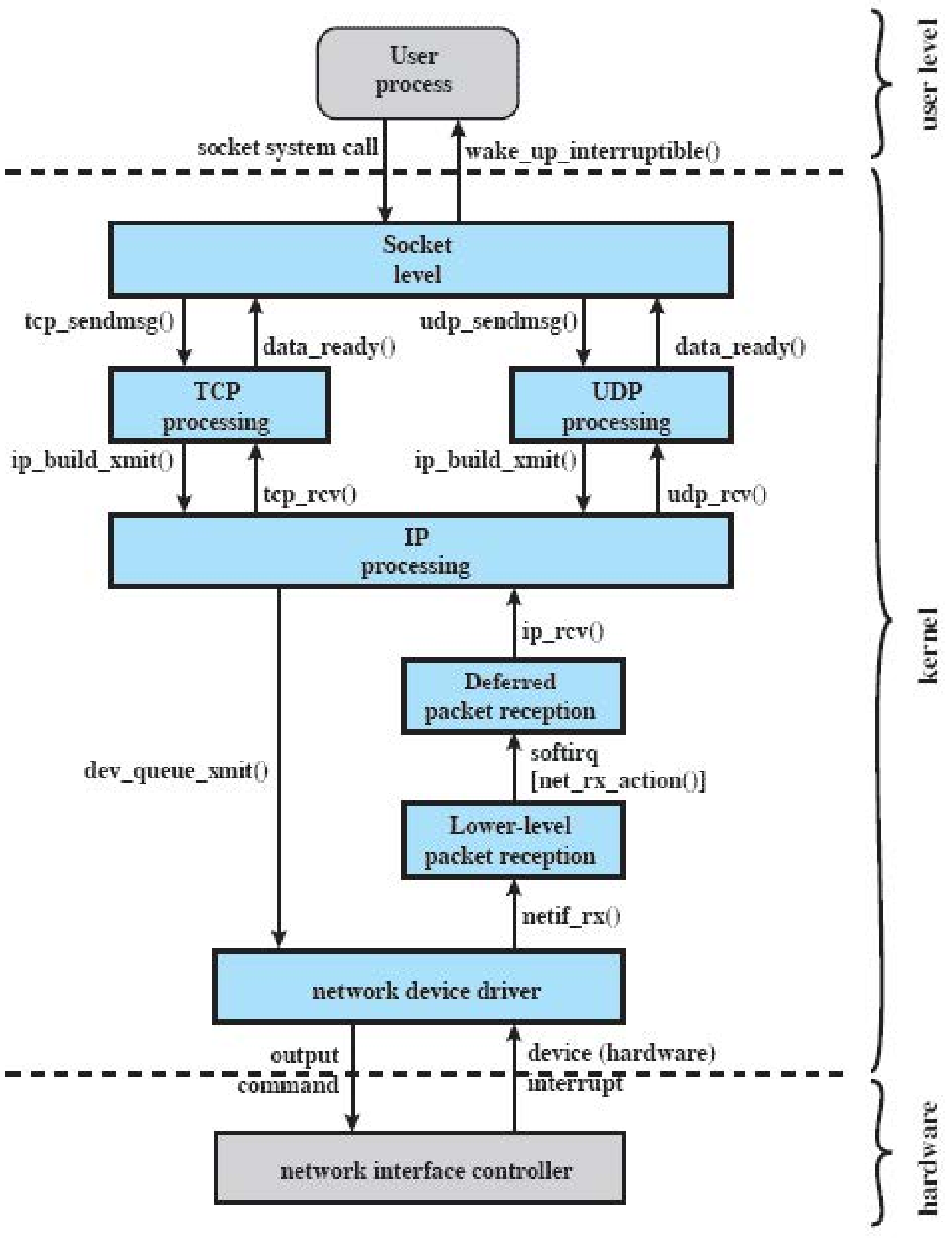

Figure 17.7 Linux Kernel Components for TCP/IP Processing

Figure 2: *Socket API internals - Linux.*