SYSC 4502 Assignment 1

Jessica Morris 100882290

February 5th, 2017

1. (a) Replacing a single, shared IP processing queue with queues-per-socket helps by eliminating the need for a software interrupt after the hardware interrupt, because the responsibility for sending the packet to a destination queue is shifted to the hardware interrupt handler. This reduces the amount of context-switching the CPU must perform. One of the issues with implementing this, is that it will add overhead to the hardware interrupt handler, as it will now have to do the IP and TCP processing determine which of the destination buffers must receive the packet.

   (b) Implementing protocol processing as part of the application, instead of having a software interrupt, would again reduce the amount of context-switching the CPU has to do. However, this method is ineffective for two reasons. Firstly, this would increase the size and overhead of all socket-using applications, as they would now be responsible for IP and TCP processing. Secondly, this would slow down applications, as they would have to wait until they have priority over the CPU to do the processing.

2. The thread pool architecture works best for short requests. Since creating a new thread requires more overhead than assigning an idle thread from a pool, thread pools work well when the number of packets/tasks is equal to or smaller than the size of the pool. The new-thread-per-packet architecture is better only when there are few, long-running tasks to run; in which case, spawning a large pool of threads to re-use would not have any inherent benefit over spawning threads as needed.

3. (a) Persistent connections are the default behaviour of HTTP connections. The "Connection" header field can be used by the client or the server to terminate a persistent connection.

   (b) HTTP by itself does not provide encryption.

   (c) Yes, a client can open three or more simultaneous connections with a given server. (Note that RFC 2616 recommends that they shouldn't have more than two connections at a time.)

   (d) Because HTTP is stateless, the client doesn't know the server's state and the server doesn't know the client's state. Therefore, it is possible that one side closes the connection for being idle for too long while the other side just finishes processing and starts to send data.

4. (a) http://gaia.cs.umass.edu/cs453/index.html
   (b) HTTP 1.1
   (c) A persistent connection (the last header line is Connection:keep-alive).
   (d) The IP address of the host is not included in a HTTP request, so this is indeterminate.
   (e) Mozilla Netscape v7.2
   (f) The browser needs to be known in order to determine the object's compatibility.

5. Let T_p denote the propagation delay of the channel. $T_p = \frac{10\,\mathrm{m}}{3 \times 10^8\,\frac{\mathrm{m}}{\mathrm{s}}} = 0.03\,\mathrm{\mu s}$

   Firstly, consider the time taken to download the objects over parallel, non-persistent connections. The total time taken to transmit objects over a non-persistent connection is given by the time taken to send a request to connect, plus the time to send an acknowledgement, plus the time taken to request an object, plus the time taken to send the object. Let $T_1(s, b)$ represent the download time for an object over this connection, where s is the size of the object being downloaded, and b is the bitrate for the download:

$$T_1(s,b) = (T_{connect}+T_p)+(T_{acknowledge}+T_p)+(T_{request}+T_p)+(T_{object}+T_p)$$

   Which in this case, simplifies to:

$$T_1(s,b) = (\frac{200\,\mathrm{bit}}{\mathrm{b}\,\frac{\mathrm{bit}}{\mathrm{s}}} + T_p) + (\frac{200\,\mathrm{bit}}{\mathrm{b}\,\frac{\mathrm{bit}}{\mathrm{s}}} + T_p) + (\frac{200\,\mathrm{bit}}{\mathrm{b}\,\frac{\mathrm{bit}}{\mathrm{s}}} + T_p) + (\frac{\mathrm{s\,bit}}{\mathrm{b}\,\frac{\mathrm{bit}}{\mathrm{s}}} + T_p)$$

$$T_1(s,b) = \frac{600+s}{b} + 4T_p$$

   Thus, the total time for the request is given by the time taken for downloading the first object, and the time for downloading one of the 10 referenced objects:

$$T_{total} = T_1(100\,\mathrm{kbit}, 150\,\frac{\mathrm{bit}}{\mathrm{s}}) + T_1(100\,\mathrm{kbit}, 15\,\frac{\mathrm{bit}}{\mathrm{s}})$$

$$T_{total} = 7377\,\mathrm{s} + 8T_p$$

For a persistent HTTP connection, the setup, acknowledge, and request packets are not needed for the 10 referenced objects. So, the time for a persistent connection, with downloads in series, is given by:

$$T_{total} = T_1(100\,\text{kbit}, 150\,\frac{\text{bit}}{\text{s}}) + 10(\frac{200\,\text{bit}}{150\,\frac{\text{bit}}{\text{s}}} + T_p + \frac{100\,\text{kbit}}{150\,\frac{\text{bit}}{\text{s}}} + T_p)$$

$$T_{total} = 7351\,\text{s} + 24T_p$$

Since T_p is negligible compared to the transmission delay, and $\Delta T = 0.35\%$, the persistent HTTP connection does not have a significant advantage over doing parallel downloads over non-persistent connection.

6. (a) A WHOIS is a public database which allows you to find information regarding domains, such as who it is registered to, who the registrar is, and who you can contact about the domain name. WHOIS databases are run by registrars and registries; for example, the Public Interest Registry maintains the .org registry.

   (b) Using whois.cira.ca: google.ca has name Domain Name Servers ns1.google.com, ns2.google.com, ns3.google.com, and ns4.google.com. duckduckgo.com has Domain Name Servers ns0.dnsmadeeasy.com, ns1.dnsmadeeasy.com, ns2.dnsmadeeasy.com, ns3.dnsmadeeasy.com, and ns4.dnsmadeeasy.com.

   (c) The local Carleton DNS server is phoenix.sce.carleton.ca, address 134.117.56.1. One of google.ca's DNS servers is ns1.google.com, address 216.239.32.10. One of duckduckgo.com's DNS servers is ns0.dnsmadeeasy.com, address 208.94.148.2.

   Ugh not quite done this yet

   (d) The Carleton web server only has one IP address. I was unable to locate a web server with multiple IP addresses.

   (e) 134.117.0.0 - 134.117.255.255 (n.b. Carleton does not appear on the ARIN whois database, so I used nirsoft.net to find this.)

   (f) The Carleton University DNS servers are: ns1.carleton.ca, ns2.carleton.ca, ns1.d-zone.ca and ns2.d-zone.ca

   (g) An attacker can use WHOIS databases and the nslookup tool to determine the IP address of a web site's server. Once they have obtained the IP address, they can flood the web server with requests in what is called a Denial-of-Service attack. A server under attack will be unable to service other clients' requests, because it has been bogged down by requests from the attacker. A reasonable web server should limit the number of requests from one client over a specific time period, but that would not stop a Distributed Denial-of-Service attack.

(h) WHOIS databases should be publicly available. They are the primary reference for finding the contact information of the registrants of domain names, as well as the IP blocks registered to those domains. This is useful if there is an issue such as a domain using more IPs than its designated block, and a primary point of contact must be located.