Ode to Prof TCP and Mr IP

So what really makes the Internet work? Why does the WWW work so well? How can we run so many applications over the Internet at the same time? How do we know that our data has been received? How does the data actually know how to get to a certain destination? Well, it's to do with TCP, IP and routing protocols. These three parts make the whole of the Internet work, and work reliably. The IP part is responsible for getting the data packets from the source to the destination (using the IP address), the TCP part is then responsible for sending the data to the required application program (using TCP sockets and sequence numbers), and the routing protocols are responsible for passing on information about how to get to destinations (using protocols such as RIP). Isn't it wonderful how a user can run a few WWW browsers, a TELNET session, an FTP session, a video conferencing session, and it all works, seamlessly, even if there are multiple destinations. Well it's really Mr IP who's sending the data to the right place, and deciding if the data on the network is for them, and Dr TCP who either tags all the outgoing data and clearly identifies the virtual connection, or reorders and passes the received data to the required application. But, Dr TCP is no egg-head who lives in the realms of academia, he makes sure that all the data is properly received, and makes sure that anything that he sends, he gets a signed receipt for. If he doesn't get a receipt, he'll re-send his data. But, what if the place that he's sending the data has blown-up, or there's a postal strike. Well Dr TCP has that side covered also; he sends the data again, and then waits for a time-out period. If no data is received, he gives up.
TCP and IP unite the world and allow everyone in the world to communicate, no matter which computer they use, which operating system they are running, which language they speak, or which network they use. It fits onto virtually every type of networking technology. It'll work with Ethernet, ATM (although with some difficulty), FDDI, ISDN, Modems, RS-232, blah, blah, blah. So, whom should we thank for giving us these two great protocols. Of course DARPA should be congratulated for conceiving it, but the main award must go to the shy, but dependable workhorse of the computing industry: UNIX. Through UNIX, TCP, UDP and IP have been allowed to blossom, and show their full potential.

UDP transmission can be likened to sending electronic mail. In most electronic mail packages the user can request that a receipt is sent back to the originator when the electronic mail has been opened. This is equivalent to TCP, where data is acknowledged after a certain amount of data has been sent. If the user does not receive a receipt for its electronic mail then it will send another one, until it is receipted or until there is a reply. UDP is equivalent to a user sending an electronic mail without asking for a receipt, thus the originator has no idea if the data has been received, or not.

TCP/IP is an excellent method for networked communications, as IP provides the routing of the data, and TCP allows acknowledgements for the data. Thus, the data can always be guaranteed to be correct. Unfortunately there is an overhead in the connection of the TCP socket, where the two communicating stations must exchange parameters before the connection is made, then they must maintain and acknowledge received TCP packets. UDP has the advantage that it is connectionless. So there is no need for a connection to be made, and data is simply thrown in the network, without the requirement for acknowledgements. Thus UDP packets are much less reliable in their operation, and a sending station cannot guarantee that the data is going to be received. UDP is thus useful for remote data acquisition where data can be simply transmitted without it being requested or without a TCP/IP connection being made.

The concept of ports and sockets is important in TCP/IP. Servers wait and listen on a given port number. They only read packets which have the correct port number. For example, a WWW server listens for data on port 80, and an FTP server listens for port 21. Thus a properly set up communication network requires a knowledge of the ports which are accessed. An excellent method for virus writers and hackers to get into a network is to install a program which responds to a given port which the hacker uses to connect to. Once into the system they can do a great deal of damage. Programming languages such as Java have built-in security to reduce this problem.

So why does TCP have a PhD, and IP doesn't? Well TCP operates at a higher layer and allows the whole system to operate reliably. It does an excellent job, whereas IP is a child that has grown up too quickly for its own use. It's excellent the way that IP has created a world-wide addressing structure, but it's limited. Why is it that my IP address is 146.176.151.130, while the address of a computer in the same street, that uses the same Internet connection has the address of 138.154.33.100. Well it's because the IP address gives no indication about the location of a node. Thus, we need complex routing protocols, in which routers use to pass information about the best way to get to a node. Some of these routing protocols, like IP, have grown up too quickly, and have outgrown their usage. The worst offender is **RIP**, which basically defines the number of hops (the number of routers in the path to the destination) that it takes to get to a destination. Unfortunately the maximum number of hops is defined at 15, thus if a destination is more that 16, the destination is not reachable. The other problem with RIP is that it's a bit lazy, and basically doesn't try too hard. It doesn't want to know about the bandwidth of a connection, or reliability, or its cost. Hop count is hardly very taxing on computing the best way to get to a destination. Just imagine if you were a car driver, and when you looked at a map you choose the route which has the minimum number of junctions. This could take you around country roads, or through congested roads. Normally we would pick the route that allows the highest average speed (the highest bandwidth), rather than the one with the minimum number of junctions.

So the Internet we have is the Internet we have. TCP, IP and RIP are there, and they're not going to be moved for a long time. But the great thing about the

Internet is that it's not going to go away either. It also allows for migration. The key to this is found in IP which can exist on its own with any other transport protocol above it, and it allows for different version, which will allow the Internet to migrate away from the current setup towards a more sensible structure (such as one based on area codes). IP addresses are also very static. If you move your computer from one network to another you must change your IP address. This normally requires skilled operators to make it work. Why can the network do it for you? In the next chapter, let's have a bit of fun, and look at how the Internet could look in the future.

In developing TCP/IP programs there are two opposite ends for code development. C++ code is complex, but very powerful, and allows for a great deal of flexibility. On the other hand, the Visual Basic code is simple to implement but is difficult to implement for non-typical applications. Thus, the code used tends to reflect the type of application. In many cases Visual Basic gives an easy-to-implement package, with the required functionality. I've seen many a student wilt at the prospect of implementing a Microsoft Windows program in C++. 'Where do I start', is always the first comment, and then 'How do I do text input', and so on. Visual Basic, on the other hand, has matured into an excellent development system which hides much of the complexity of Microsoft Windows away from the developer. So, don't worry about computer language snobbery. Pick the best language to implement the specification.

--- William J.Buchanan, June 6, 2001