A Look at the Early Internet

By PAUL KING ©2006

1. UNIX, Networks, and Open Standards

Back in 1968-1969, Kernighan and Ritchie were developing the C programming language at the University of California in Berkeley for what was a new, cutting-edge operating system known as BSD UNIX.

The original BSD UNIX was an operating system which acted as an intermediate between software and hardware, serving the same essential functions as Microsoft Windows does today. For over 35 years, UNIX has survived and grown into several nearly identical "Flavours". There is *AT&T UNIX*; *IBM's AIX*; SUN Microsystem's *Solaris*; and many more. Many of these operating systems are no longer used, due to the immense popularity of the newest addition to the flavours of UNIX: the introduction in 1992 of the *GNU/LINUX* operating system, usually called *Linux*. *GNU/Linux* was a joint effort of Linus Torvalds (kernel – the "*Linux*" part) and Richard Stallman (*UNIX* commands – the "*GNU*" part). Both Torvalds and Stallman headed large projects. The product was to be downloadable for free, and anyone can join the thousands already volunteering on this open source project. The source code is free for anyone, and anyone can modify it for themselves.

Many people run many different computers made by different manufacturers. These different computers would have different hardware and perform either different functions or perform the same functions differently. In the 1960s, those with a large number of computers: government, universities, and large businesses, saw the need for communication between computer networks.

The designers of the original UNIX had this very goal in mind. They didn't care too much about how your computer was built, or what operating system it ran. While it too could run UNIX, it could also be running any of the other operating systems available at the time. And if MS-Windows was available, it would recognize that too (as it does today). Today, if a Windows machine "talks" to (sends messages to) a UNIX machine over a network, it must follow one of several network protocols.

For example, if a user wanted to download a file, there was a couple of ways to do that: either using rcp (remote copy), uucp (UNIX-to-UNIX copy) or the protocol that survives today: FTP (file transfer protocol).

A **protocol** is a list of rules a remote machine must follow if it is to successfully communicate with the server. It is up to the remote machine to understand these rules; and it usually does so if the right software is installed. Today, the major protocols used are: FTP, NNTP (net-news transfer protocol), SMTP (simple mail transfer protocol) for sending electronic mail (e-mail), and of course: HTTP (hypertext transfer protocol) for the World-Wide Web. At no point do we ever concern ourselves with whether the server is running Microsoft, UNIX, or even MAC OS. Everything just "seems to work". And this is because all these different systems understand the necessary protocols for communication over the Internet.

An old protocol that is in wide use today is called Telnet. These days, it has been referred to as "shell access" to a server. It allows a user to remotely log on to a remote computer and operate it through keyboard commands as if he were sitting at the computer. These days, people more often use a more secure form of Telnet called "Secure Shell", or SSH.

All of these protocols involve the passing of data packets to remote computers in a network. The remote computer would re-assemble the packets to create things such as a data file, a remote computer command, or an e-mail message.

For a message to reach the intended destination, each computer must be uniquely identified on a network. Computers are normally given names, called host names (such as www.google.com). However, computer hardware requires a unique numerical address called an **IP (Internet Protocol) address**, which servers recognise, especially **domain name servers** (DNS). When you type a hostname on to your web browser, a DNS tries to match that with an IP address. If it can't find one, it returns an error message back to your browser.

Each node, as shown on the map in Figure 1, represented a cluster of computers in a major organization, joined by a server, which acted as a gateway to other networks across the United States. While each node, therefore, was a vast network consisting of hundreds of computers and workstations (sometimes called an **intranet**), each of these networks were also joined to each other as a network of networks, called an **internet**. In the beginning, most communication including email was done by UUCP.

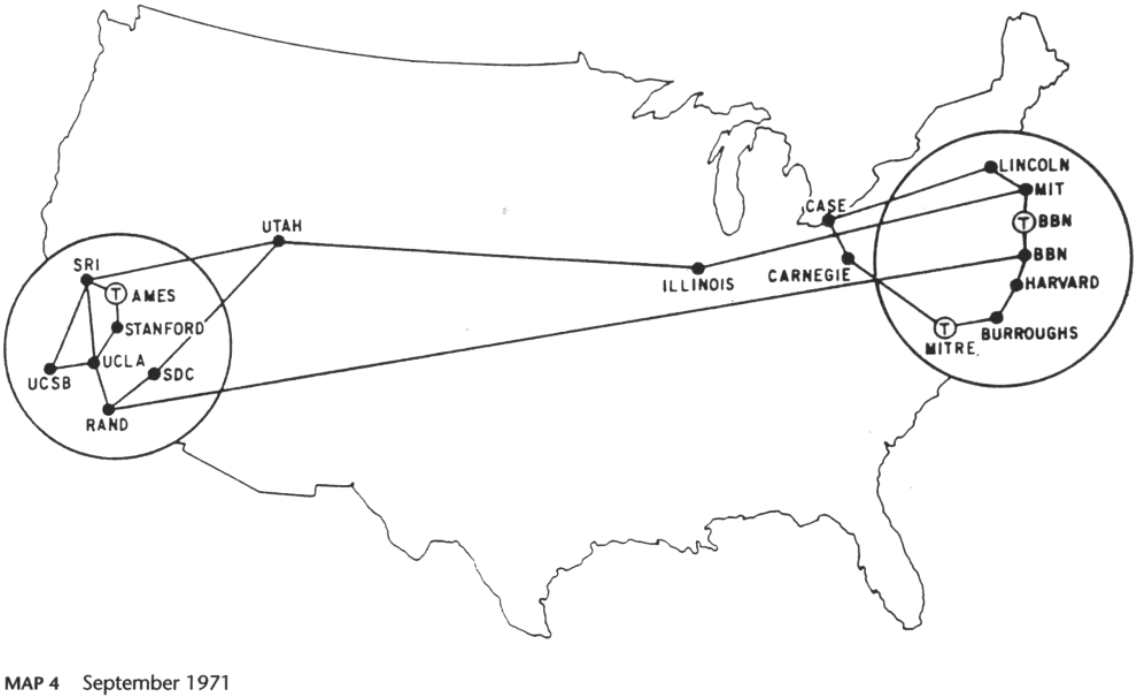

Figure 1 - The ARPANET in 1971 – this illustration is from the US Department of Defense

In 1969, the **ARPANET** consisted of four nodes; by 1971, that number grew to 18. Among other nodes, the 1971 ARPANET had consisted of several universities, such as UCLA (University of California, Los Angeles), UCSB (University of California, Santa Barbara), Stanford University and Stanford Research Institute (SRI), UIUC (University of Illinois, Urbana-Champaign), MIT (Massachusetts Institute of Technology), CWRU (CASE Western Reserve University), and Harvard University. It also consisted of research institutes such as the RAND institute, Burroughs-Rice, the Carnegie Institute and Lincoln Labs.

This Internet was called **ARPANET**, or the **Advanced Research Projects Agency Network**. The ARPANET was an invention of the United States Department of Defense, who wanted an interconnected computer network that could deliver messages along any route from source to destination. At the time, Russia was still considered a military threat. So, in the event of a nuclear attack, if a bomb took out one or more nodes, it should be possible to still send messages to any surviving node. The ideal situation was to have all nodes connected by an edge (a complete network), but this was expensive. It

was thought instead that messages should be allowed to bounce from node to node on its way to its destination.

But that does not tell the whole story. The Internet was just as much a product of Red-Scare paranoia as it was of Flower Children and the Civil Rights Movement growing on many university campuses at the time. The idea of several computer architectures made by competing companies needing now to communicate with each other required **open standards**. That is to say, if a Mac user has to send an electronic mail (**email**) message to a Windows user, both systems are made by different manufacturers, yet it is still possible for the MAC user to send email to the PC user over the Internet. This is because both operating systems obey the same rules for sending and receiving email. If Microsoft decides to change the standards for email (which it has in the past, regarding MIME attachments), then it has to share this with others. To this day, if a Mac and a PC are directly connected, MAC users cannot send print jobs to Windows users and vice versa, due to differing protocols on local networks.

Local network protocols have been regarded as trade secrets in order to lock out competitors and lock in customers. It is this kind of competition (called **consumer lock-in**) that has made *Microsoft* dominant in the software industry, and *IBM* before them. *Apple* was actually the worst offender, for not even allowing their hardware protocols to be known. This was a shame, since that meant that no one except an *Apple* employee could write software for it, and only *Apple* could manufacture the computer parts. This was not true for *IBM* or *Microsoft* hardware. *Apple* was dominant for a while, but later, customers became attracted to *Windows*, as more software seemed to run on those systems than for *Apple*. Thus, open standards for hardware were one major influence in *Microsoft* becoming the dominant player in the industry, and for the PC becoming the dominant architecture. Today, the only way to send print jobs or view files between *Macintosh* and *Windows* is through a *Linux* file and print server called *Samba*.

In order to be a player on the Internet, companies finally had to get along. This idea was popular with people in the Civil Rights Movement who were attracted to anything that induced people to be friends; and were suspicious of the evils of corporate competition. Open standards had a democratizing influence on corporations, and this attracted the attention of people at the time such as Eric Raymond and Richard Stallman who appreciated the negative effect competition had on the advancement of technology, and still write articles and go on speaking tours about open standards and the need for **open-source software** today.

The development of the Internet using open standards (and later *Linux*, which they both were prominent in developing) proved to them that technology develops faster if knowledge is shared among people rather than hoarded as private property and patented. While privatizing information with patents benefit companies in the short term, technological advancement for the good of society slows down in the long term. Indeed, *Linux* developed faster in its first 5 years than Microsoft Windows did in twenty-five. Today, *Linux* is robust enough that the Canadian Broadcasting Corporation (CBC) has trusted it to all of its web content, and internal computing needs. Governments of nations such as Brazil, France, and Germany have now turned their computers entirely over to *Linux*.

It can be fair to say that without open standards set by its far-seeing architects of the 1960s, there would never have been an Internet or for that matter, *Linux*. Companies would only insist on *their* standards, since that would mean people would buy *their* software. They would never agree to a shared resource (such as the Internet) that everyone could use, since they know it would be too big for any one company to control.

It must be said also that Open Source will not spell the end of computer companies. Open Source has its own business model which companies can still make money. They will just be given one less way to kill off their competitors, and will likely not be making money in the amounts that companies such as Microsoft have been doing.

2. Message Passing on the ARPANET

Figure 2 below displays the network as **planar (no crossing edges)**. Our purpose here is to find out, given the simplifying assumption that all Internet connections are 2-way, what is the maximum number of relay servers necessary to send a message to its destination in the 1971 design of ARPANET?

If a user at SDC wanted to send a message to a user at UCLA, you can see that there are several routes to the destination. The path through Utah/SRI/UCSB to UCLA passes through 3 relay servers (4 **hops**); the route through AMES requires 5 hops; Utah/SRI requires 3 hops; while the route through RAND requires only two hops. If all paths are considered equivalent (they are not, but we shall assume they are for simplicity) then clearly the best choice of a route would have to be through RAND.

Paths from one node to another could start by representing all nodes as a **matrix**. The matrix would be represented by a "1" if an edge directly connected one node to the next; and by a "0" if there was no single edge connecting the node to another. For example, a connection between MIT and UIUC would receive a "1"; but a connection between UIUC and UCLA would receive a "0", since they are not directly connected by an edge. Instead, a message passed between UIUC and UCLA would have to undergo at least 3 hops (Utah/SRI/UCSB) before reaching UCLA.

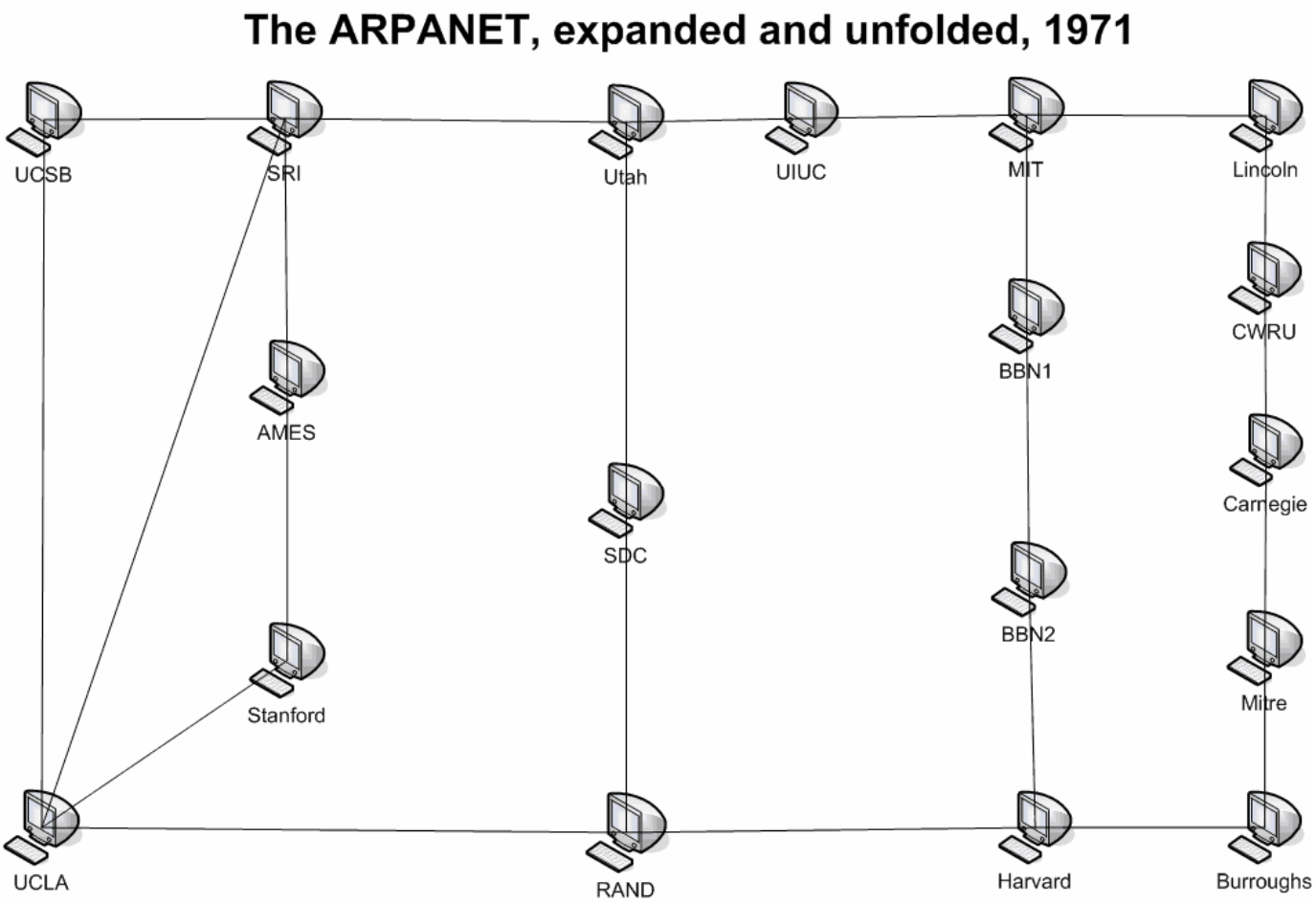

Figure 2 - The same ARPANET un-folded to show the connections more clearly. The positions of the nodes on this graph bear no relationship to their actual geographical positioning in the United States.

Supposing we arranged the servers in an 18x18 matrix to show how the computers were connected. A "1" is a **edge** between a pair of **nodes**, and a "0" is used when an edge does not link a pair of nodes. Rows and columns are arranged in the same order:

ucsb	sri	utah	uiuc	mit	linc	ames	bbn1	cwru	stan	sdc	bbn2	carn	mitr	ucla	rand	harv	burr
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

You can see there are a lot of zeroes in the matrix. Such a matrix with few edges is called a **sparse matrix**. If the above graph were **complete** (all pairs joined by one edge), the entire matrix would be filled with 1's.

Does the matrix get less sparse as we increase the number of hops? The number of servers now connected by two hops will be the matrix multiplied by itself.
It would be impractical to work out the square of this matrix by hand, so we need a software package. We shall use ***Matlab® 7*** as the software package for this work.

Nodes connected by exactly two hops:

Ucsb	sri	utah	uiuc	mit	linc	ames	bbn1	cwru	stan	sdc	bbn2	carn	mitr	ucla	rand	harv	burr
2	1	1	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0
1	4	0	1	0	0	0	0	0	2	1	0	0	0	1	1	0	0
1	0	3	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0
0	1	0	2	0	1	0	1	0	0	1	0	0	0	0	0	0	0
0	0	1	0	3	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	1	0	2	0	1	0	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	2	0	0	0	0	0	0	0	2	0	0	0
0	0	0	1	0	1	0	2	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	2	0	0	0	0	1	0	0	0	0
1	2	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	2	0	0	0	1	0	1	0
0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	1	0	1
0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	1	0
1	1	1	0	0	0	2	0	0	0	1	0	0	0	4	0	1	0
1	1	1	0	0	0	0	0	0	1	0	1	0	0	0	3	0	1
0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	3	0
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	2

We can see that the matrix is starting to gradually fill with numbers 1 and greater. A number greater than 1 means, for example, that for a message to go from SRI to Stanford (stan), there are two paths it can choose for a message to travel two hops. This can be verified by looking at the graph. The message can pass either through AMES or UCLA. Note that the numbers increase greatly along the **main diagonal** of the matrix. These are ways servers can pass a message to itself by two hops by bouncing the message off a neighbouring node.

Typically, though a person is lucky to receive a message that only underwent one or two hops. Especially in today's internet, messages can undergo several hops before reaching its destination. This matrix doesn't really become full until we consider message-passing after 8 hops. The graph below shows the original matrix multiplied by itself 8 times (N^8) using ***Matlab® 7***. The columns denote "from" and the rows denote "to" with regards to message passing.

	ucsb	sri	utah	uiuc	mit	linc	ames	bbn1	cwru	stan	sdc	bbn2	carn	mitr	ucla	rand	harv	burr
ucsb	535	700	460	142	109	19	404	34	15	405	251	108	14	18	700	473	159	96
sri	700	1260	432	353	67	83	393	108	6	711	461	127	27	29	839	676	243	125
utah	460	432	530	67	212	6	423	20	53	245	175	133	13	26	661	390	144	92
uiuc	142	353	67	203	5	118	74	130	2	165	186	17	40	22	216	129	133	25
mit	109	67	212	5	239	1	95	2	105	45	21	130	11	39	128	145	27	63
linc	19	83	6	118	1	119	7	107	1	27	64	10	66	11	41	26	63	31
ames	404	393	423	74	95	7	432	31	15	189	210	44	2	27	712	246	191	32
bbn1	34	108	20	130	2	107	31	132	9	31	107	2	38	41	109	21	142	10
cwru	15	6	53	2	105	1	15	9	80	3	11	40	1	58	27	23	40	11
stan	405	711	245	165	45	27	189	31	3	433	208	95	15	5	393	437	79	93
sdc	251	461	175	186	21	64	210	107	11	208	273	20	12	54	472	175	239	20
bbn2	108	127	133	17	130	10	44	2	40	95	20	144	39	9	66	213	6	120
carn	14	27	13	40	11	66	2	38	1	15	12	39	72	1	6	52	11	67
mitr	18	29	26	22	39	11	27	41	58	5	54	9	1	81	81	6	118	1
ucla	700	839	661	216	128	41	712	109	27	393	472	66	6	81	1274	434	434	52
rand	473	676	390	129	145	26	246	21	23	437	175	213	52	6	434	594	71	201
harv	159	243	144	133	27	63	191	142	40	79	239	6	11	118	434	71	335	5
burr	96	125	92	25	63	31	32	10	11	93	20	120	67	1	52	201	5	132

The numbers in the matrix represent the number of ways by which it takes 8 hops to get a message to its destination. They by no means denote the most practical route or the shortest route. So, if a “1” appears in a **cell** it does not mean that other, shorter routes don’t exist between the destinations. For example, a message from “linc” to “mit” has an entry of “1” for 8 hops, yet they lie next to each other on the graph and are joined by an edge. However, we can now see that the matrix no longer has any 0’s in it, meaning that messages can be passed from any node to any other node after **at most** 8 hops.