What is a Domain?

Mark R. Horton

ABSTRACT

In the past, electronic mail has used many different kinds of syntax, naming a computer and a login name on that computer. A new system, called ‘‘domains’’, is becoming widely used, based on a heirarchical naming scheme. This paper is intended as a quick introduction to domains. For more details, you should read some of the documents referenced at the end.

1. Introduction

What exactly are domains? Basically, they are a way of looking at the world as a heirarchy (tree structure). You’re already used to using two tree world models that work pretty well: the telephone system and the post office. Domains form a similar heirarchy for the electronic mail community.

The post office divides the world up geographically, first into countries, then each country divides itself up, those units subdivide, and so on. One such country, the USA, divides into states, which divide into counties (except for certain states, like Louisiana, which divide into things like parishes), the counties subdivide into cities, towns, and townships, which typically divide into streets, the streets divide into lots with addresses, possibly containing room and apartment numbers, the then individual people at that address. So you have an address like

> Mark Horton
> Room 2C-249
> 6200 E. Broad St.
> Columbus, Ohio, USA

(I’m ignoring the name ‘‘AT&T Bell Laboratories’’ and the zip code, which are redundant information.) Other countries may subdivide differently, for example many small countries do not have states.

The telephone system is similar. Your full phone number might look like 1-614-860-1234 x234 This contains, from left to right, your country code (Surprise! The USA has country code ‘‘1’’!), area code 614 (Central Ohio), 860 (a prefix in the Reynoldsburg C.O.), 1234 (individual phone number), and extension 234. Some phone numbers do not have extensions, but the phone system in the USA has standardized on a 3 digit area code, 3 digit prefix, and 4 digit phone number. Other countries don’t use this standard, for example, in the Netherlands a number might be +46 8 7821234 (country code 46, city code 8, number 7821234), in Germany +49 231 7551234, in Sweden +31 80 551234, in Britain +44 227 61234 or +44 506 411234. Note that the country and city codes and telephone numbers are not all the same length, and the punctuation is different from our North American notation. Within a country, the length of the telephone number might depend on the city code. Even within the USA, the length of extensions is not standardized: some places use the last 4 digits of the telephone number for the extension, some use 2 or 3 or 4 digit extensions you must ask an operator for. Each country has established local conventions. But the numbers are unambigous when dialed from left-to-right, so as long as there is a way to indicate when you are done dialing, there is no problem.

A key difference in philosophy between the two systems is evident from the way addresses and telephone numbers are written. With an address, the most specific information comes first, the least specific last. (The ‘‘root of the tree’’ is at the right.) With telephones, the least specific information (root) is at the left. The telephone system was designed for machinery that looks at the first few digits, does something with it, and passes the remainder through to the next level. Thus, in effect, you are routing your call through the telephone network. Of course, the exact sequence you dial depends on where you are dialing from - sometimes you must dial 9 or 8 first, to get an international dialtone you must dial 011, if you are calling locally you can (and sometimes must) leave off the 1 and the area code. (This makes life very interesting for people who must design a box to call their home office from any phone in the world.) This type of address is called a ‘‘relative address’’, since the actual address used depends on the location of the sender.

The postal system, on the other hand, allows you to write the same address no matter where the sender is. The address above will get to me from anywhere in the world, even private company mail systems. Yet, some optional abbreviations are possible - I can leave off the USA if I’m mailing within the USA; if I’m in the same city as the address, I can usually just say ‘‘city’’ in place of the last line. This type of address is called an ‘‘absolute address’’, since the unabbreviated form does not depend on the location of the sender.

The ARPANET has evolved with a system of absolute addresses: ‘‘user@host’’ works from any machine. The UUCP network has evolved with a system of relative addresses: ‘‘host!user’’ works from any machine with a direct link to ‘‘host’’, and you have to route your mail through the network to find such a machine. In fact, the ‘‘user@host’’ syntax has become so popular that many sites run mail software that accepts this syntax, looks up ‘‘host’’ in a table, and sends it to the appropriate network for ‘‘host’’. This is a very nice user interface, but it only works well in a small network. Once the set of allowed hosts grows past about 1000 hosts, you run into all sorts of administrative problems.

One problem is that it becomes nearly impossible to keep a table of host names up to date. New machines are being added somewhere in the world every day, and nobody tells you about them. When you try to send mail to a host that isn’t in your table (replying to mail you just got from a new host), your mailing software might try to route it to a smarter machine, but without knowing which network to send it to, it can’t guess which smarter machine to forward to. Another problem is name space collision - there is nothing to prevent a host on one network from choosing the same name as a host on another network. For example, DEC’s ENET has a ‘‘vortex’’ machine, there is also one on UUCP. Both had their names long before the two networks could talk to each other, and neither had to ask the other network for permission to use the name. The problem is compounded when you consider how many computer centers name their machines ‘‘A’’, ‘‘B’’, ‘‘C’’, and so on.

In recognition of this problem, ARPA has established a new way to name computers based on domains. The ARPANET is pioneering the domain convention, and many other computer networks are falling in line, since it is the first naming convention that looks like it really stands a chance of working. The MILNET portion of ARPANET has a domain, CSNET has one, and it appears that Digital, AT&T, and UUCP will be using domains as well. Domains look a lot like postal addresses, with a simple syntax that fits on one line, is easy to type, and is easy for computers to handle. To illustrate, an old routed UUCP address might read ‘‘sdcsvax!ucbvax!allegra!cbosgd!mark’’. The domain version of this might read ‘‘mark@d.osg.cb.att.uucp’’. The machine is named d.osg.cb.att.uucp (UUCP domain, AT&T company, Columbus site, Operating System Group project, fourth machine.) Of course, this example is somewhat verbose and contrived; it illustrates the heirarchy well, but most people would rather type something like ‘‘cbosgd.att.uucp’’ or even ‘‘cbosgd.uucp’’, and actual domains are usually set up so that you don’t have to type very much.

You may wonder why the single @ sign is present, that is, why the above address does not read ‘‘mark.d.osg.cb.att.uucp’’. In fact, it was originally proposed in this form, and some of the examples in RFC819 do not contain an @ sign. The @ sign is present because some ARPANET sites felt the strong need for a divider between the domain, which names one or more computers, and the left hand side, which is subject to whatever interpretation the domain chooses. For example, if the ATT domain chooses to address people by full name rather than by their login, an address like ‘‘Mark.Horton@ATT.UUCP’’ makes it clear that some machine in the ATT domain should interpret the string ‘‘Mark.Horton’’, but if the address were ‘‘Mark.Horton.ATT.UUCP’’, routing software might try to find a machine named ‘‘Horton’’ or ‘‘Mark.Horton’’. (By the way, case is ignored in domains, so that ‘‘ATT.UUCP’’ is the same as ‘‘att.uucp’’. To the left of the @ sign, however, a domain can interpret the text any way it wants; case can be ignored or it can be significant.)

It is important to note that **domains are not routes**. Some people look at the number of !’s in the first example and the number of .’s in the second, and assume the latter is being routed from a machine called ‘‘uucp’’ to another called ‘‘att’’ to another called ‘‘cb’’ and so on. While it is possible to set up mail routing software to do this, and indeed in the worst case, even without a reasonable set of tables, this method will always work, the intent is that ‘‘d.osg.cb.att.uucp’’ is the name of a machine, not a path to get there. In particular, domains are absolute addresses, while routes depend on the location of the sender. Some subroutine is charged with figuring out, given a domain based machine name, what to do with it. In a high quality environment like the ARPA Internet, it can query a table or a name server, come up with a 32 bit host number, and connect you directly to that machine. In the UUCP environment, we don’t have the concept of two processes on arbitrary machines talking directly, so we forward mail one hop at a time until it gets to the appropriate destination. In this case, the subroutine decides if the name represents the local machine, and if not, decides which of its neighbors to forward the message to.

2. What is a Domain?

So, after all this background, we still haven’t said what a domain is. The answer (I hope it’s been worth the wait) is that a domain is a subtree of the world tree. For example, ‘‘uucp’’ is a top level domain (that is, a subtree of the ‘‘root’’.) and represents all names and machines beneath it in the tree. ‘‘att.uucp’’ is a subdomain of ‘‘uucp’’, representing all names, machines, and subdomains beneath ‘‘att’’ in the tree. Similarly for ‘‘cb.att.uucp’’, ‘‘osg.cb.att.uucp’’, and even ‘‘d.osg.cb.att.uucp’’ (although ‘‘d.osg.cb.att.uucp’’ is a ‘‘leaf’’ domain, representing only the one machine).

A domain has certain properties. The key property is that it has a ‘‘registry’’. That is, the domain has a list of the names of all immediate subdomains, plus information about how to get to each one. There is also a contact person for the domain. This person is responsible for the domain, keeping the registry up-to-date, serving as a point of contact for outside queries, and setting policy requirements for subdomains. Each subdomain can decide who it will allow to have subdomains, and establish requirements that all subdomains must meet to be included in the registry. For example, the ‘‘cb’’ domain might require all subdomains to be physically located in the AT&T building in Columbus.

ARPA has established certain requirements for top level domains. These requirements specify that there must be a list of all subdomains and contact persons for them, a responsible person who is an authority for the domain (so that if some site does something bad, it can be made to stop), a minimum size (to prevent small domains from being top level), and a pair of nameservers (for redundancy) to provide a directory-assistance facility. Domains can be more lax about the requirements they place on their subdomains, making it harder to be a top level domain than somewhere lower in the tree. Of course, if you are a subdomain, your parent is responsible for you.

One requirement that is NOT present is for unique parents. That is, a machine (or an entire subdomain) need not appear in only one place in the tree. Thus, ‘‘cb’’ might appear both in the ‘‘att’’ domain, and in the ‘‘ohio’’ domain. This allows domains to be structured more flexibly than just the simple geography used by the postal service and the telephone company; organizations or topography can be used in parallel. (Actually, there are a few instances where this is done in the postal service [overseas military mail] and the telephone system [prefixes can appear in more than one area code, e.g. near Washington D.C., and Silicon Valley].) It also allows domains to split or join up, while remaining upward compatible with their old addresses.

Do all domains represent specific machines? Not necessarily. It’s pretty obvious that a full path like ‘‘d.cbosg.att.uucp’’ refers to exactly one machine. The OSG domain might decide that ‘‘cbosg.att.uucp’’ represents a particular gateway machine. Or it might decide that it represents a set of machines, several of which might be gateways. The ‘‘att.uucp’’ domain might decide that several machines, ‘‘ihnp4.uucp’’, ‘‘whgwj.uucp’’, and ‘‘hogtw.uucp’’ are all entry points into ‘‘att.uucp’’. Or it might decide that it just represents a spot in the name space, not a machine. For example, there is no machine corresponding to ‘‘arpa’’ or ‘‘uucp’’, or to the root. Each domain decides for itself. The naming space and the algorithm for getting mail from one machine to another are not closely linked - routing is up to the mail system to figure out, with or without help from the structure of the names.

The domain syntax does allow explicit routes, in case you want to exercise a particular route or some gateway is balking. The syntax is ‘‘@dom$_1$,@dom$_2$,...,@dom$_n$:user@domain’’, for example, @ihnp4.UUCP,@ucbvax.UUCP,:joe@NIC.ARPA, forcing it to be routed through dom$_1$, dom$_2$, ..., dom$_n$, and from domn sent to the final address. This behaves exactly like the UUCP ! routing syntax, although it is somewhat more verbose.

By the way, you've no doubt noticed that some forms of electronic addresses read from left-to-right (cbosgd!mark), others read from right-to-left (mark@Berkeley). Which is better? The real answer here is that it's a religious issue, and it doesn't make much difference. left-to-right is probably a bit easier for a computer to deal with because it can understand something on the left and ignore the remainder of the address. (While it's almost as easy for the program to read from right-to-left, the ease of going from left-to-right was probably in the backs of the minds of the designers who invented host:user and host!user.)

On the other hand, I claim that user@host is easier for humans to read, since people tend to start reading from the left and quit as soon as they recognize the login name of the person. Also, a mail program that prints a table of headers may have to truncate the sender's address to make it fit in a fixed number of columns, and it's probably more useful to read ''mark@d.osg.a'' than ''ucbvax!sdcsv''.

These are pretty minor issues, after all, humans can adapt to skip to the end of an address, and programs can truncate on the left. But the real problem is that if the world contains BOTH left-to-right and right-to-left syntax, you have ambiguous addresses like x!y@z to consider. This single problem turns out to be a killer, and is the best single reason to try to stamp out one in favor of the other.

3. So why are we doing this, anyway?

The current world is full of lots of interesting kinds of mail syntax. The old ARPA ''user@host'' is still used on the ARPANET by many systems. Explicit routing can sometimes by done with an address like ''user@host2@host1'' which sends the mail to host1 and lets host1 interpret ''user@host2''. Addresses with more than one @ were made illegal a few years ago, but many ARPANET hosts depended on them, and the syntax is still being used. UUCP uses ''h1!h2!h3!user'', requiring the user to route the mail. Berknets use ''host:user'' and do not allow explicit routing.

To get mail from one host to another, it had to be routed through gateways. Thus, the address ''csvax:mark@Berkeley'' from the ARPANET would send the mail to Berkeley, which would forward it to the Berknet address csvax:mark. To send mail to the ARPANET from UUCP, an address such as ''ihnp4!ucbvax!sam@foo-unix'' would route it through ihnp4 to ucbvax, which would interpret ''sam@foo-unix'' as an ARPANET address and pass it along. When the Berknet-UUCP gateway and Berknet-ARPANET gateway were on different machines, addresses such as ''csvax:ihnp4!ihnss!warren@Berkeley'' were common.

As you can see, the combination of left-to-right UUCP syntax and right-to-left ARPANET syntax makes things pretty complex. Berknets are gone now, but there are lots of gateways between UUCP and the ARPANET and ARPANET-like mail networks. Sending mail to an address for which you only know a path from the ARPANET onto UUCP is even harder − suppose the address you have is ihnp4!ihnss!warren@Berkeley, and you are on host rlgvax which uses seismo as an ARPANET gateway. You must send to seismo!ihnp4!ihnss!warren@Berkeley, which is not only pretty hard to read, but when the recipient tries to reply, it will have no idea where the break in the address between the two UUCP pieces occurs. An ARPANET site routing across the UUCP world to somebody's Ethernet using domains locally will have to send an address something like ''xxx@Berkeley.ARPA'' to get it to UUCP, then ''ihnp4!decvax!island!yyy'' to get it to the other ethernet, then ''sam@csvax.ISLAND'' to get it across their ethernet. The single address would therefore be ihnp4!decvax!island!sam@csvax.ISLAND@Berkeley.ARPA, which is too much to ask any person or mailer to understand. It's even worse: gateways have to deal with ambiguous names like ihnp4!mark@Berkeley, which can be parsed either ''(ihnp4!mark)@Berkeley'' in accordance with the ARPANET conventions, or ''ihnp4!(mark@Berkeley)'' as the old UUCP would.

Another very important reason for using domains is that your mailing address becomes absolute instead of relative. It becomes possible to put your electronic address on your business card or in your signature file without worrying about writing six different forms and fifteen hosts that know how to get to yours. It drastically simplifies the job of the reply command in your mail program, and automatic reply code in the netnews software.

4. Further Information

For further information, some of the basic ARPANET reference documents are in order. These can often be found posted to Usenet, or available nearby. They are all available on the ARPANET on host NIC via FTP with login ANONYMOUS, if you have an ARPANET login. They can also be ordered from the Network Information Center, SRI International, Menlo Park, California, 94025.

RFC819	The Domain Naming Convention for Internet User Applications
RFC821	Simple Mail Transfer Protocol
RFC822	Standard for the Format of ARPANET Text Messages
RFC881	The Domain Names Plan and Schedule

```
#
# @(#)domain.mm         2.1 smail 12/14/86
#
```