Subject: TCP/IP an explanation, pt 1
Date: Thu, 10 Jan 2002 02:04:00 +0000
From: G1SOG@GB7GLO.#46.GBR.EU
To : TCPIP@GBR
de g1sog
Richard

This epic is in response to a request for an extension to an explanation I gave on how IP is transported in AX.25 packets.

I have started with a reworked section on encapsulation ("wrapping") below, to follow that a large section on protocol stacks and what encapsulation takes place where. From there it depends on what requests I get. I have outlines for sections on IP addresses, hostname to IP address mapping and routing. I may do more it depends on what is asked.

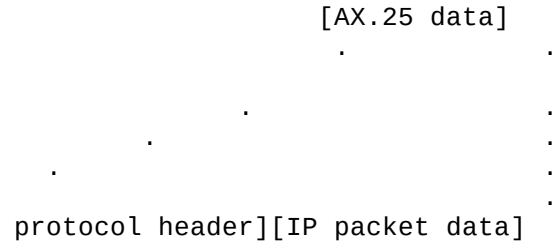
---

1) Data encapsulation ("Wrapping").

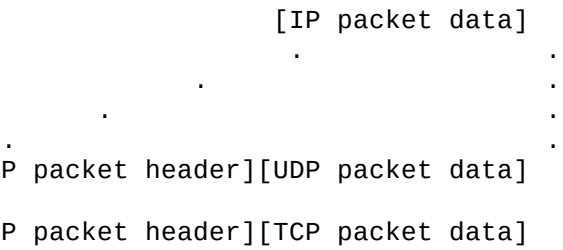
Now an AX.25 packet consists of:

[AX.25 protocol header][AX.25 data][FCS]

When we are transporting IP via AX.25 the AX.25 data is the IP packet:



Now the [IP packet data] is encapsulating a layer. With IP this could be a UDP (User Datagram Protocol) or TCP (Transmission Control Protocol) packet.



or

[TCP packet header][TCP packet data]

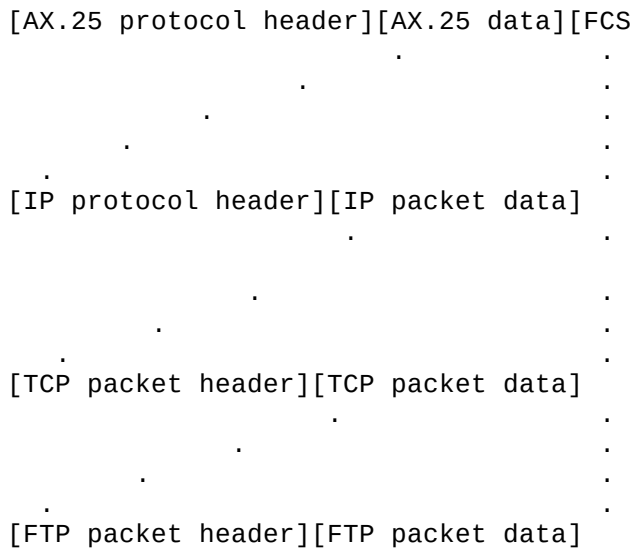
The User Datagram Protocol provides a mechanism for datagram’s to be sent from end to end. However it does not provide any form of delivery guarantee. The "user" application is responsible for ensuring data integrity.

The Transmission Control Protocol establishes a virtual circuit for the data transport and provides a guarantee of delivery, or an error message if delivery fails.

Now yet again there is an encapsulation taking place as the UDP/TCP packet payload is data from the applications running on the computer, telnet, ftp, httpd, ... most of which implement some form of protocol which "wraps" the users data in a packet. Fortunately most of the protocols have small headers because this multiple "wrapping" was considered when the protocols were designed. The shortest IP packet is about 40 bytes and carries no

user data at all but an ICMP packet which is IP protocol control information.

So for an AX.25 packet carrying part of an FTP session we might have:



From this it should be clear why IP is somewhat clunky on a 1k2 half-duplex network, you don't get much user data in a 256 AX.25 payload (about 200 bytes max).

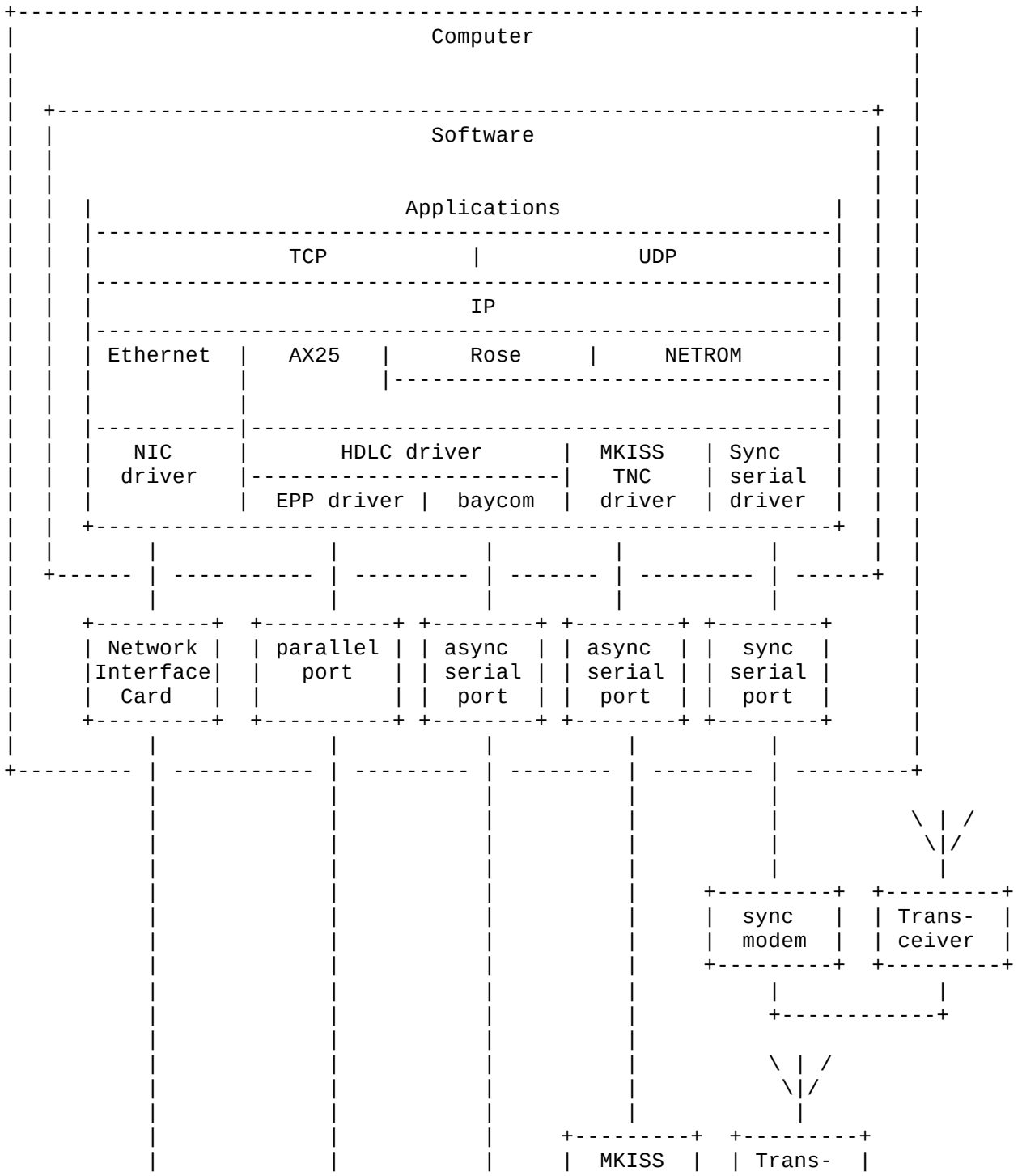
--

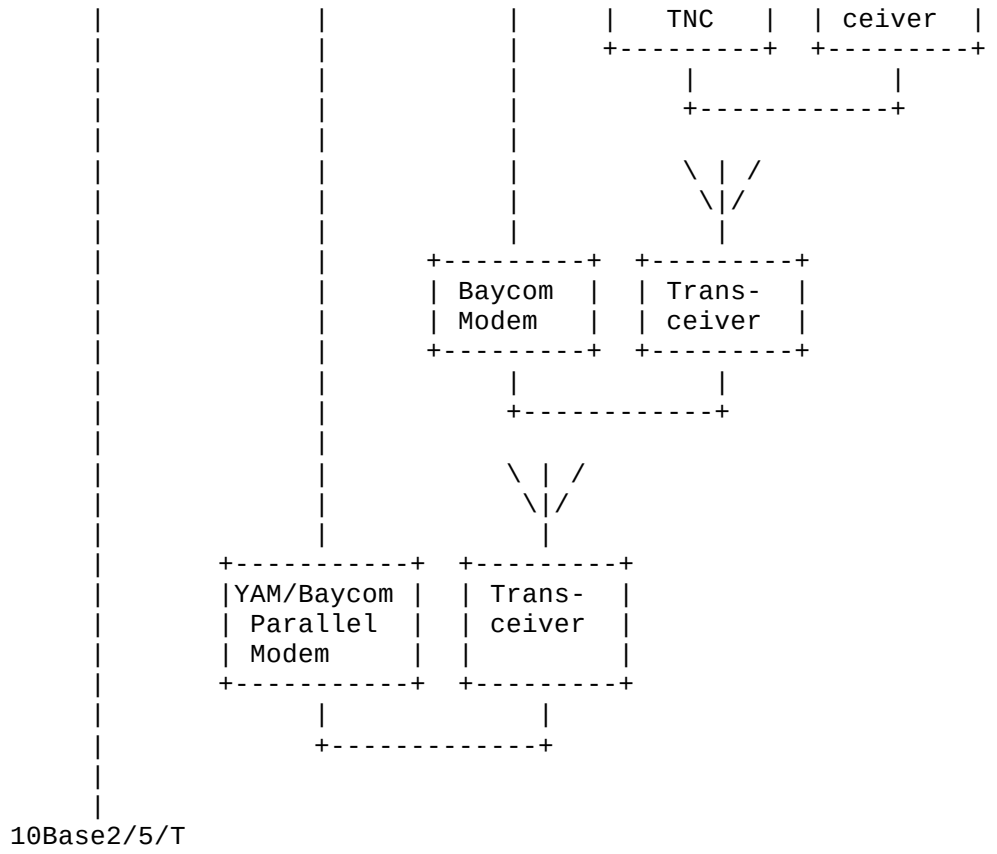
Subject: TCP/IP an explanation, pt 2

2) Software Layers.
==========

Now for the big nasty diagram:
==========

Here I am attempting to show the "layering" of software within the OS.
With most of the software "layers" come the encapsulation "layers".
I am using Linux because I am most familiar with Unix and Linux is still
(I believe) the only OS that supports the AX.25 protocol natively.





Now to put the "wrapping" and the software layers together:
==========

Applications:
- ftp - an implementation of the File Transfer Protocol
- telnet - network terminal
- sendmail - an implementation of the Simple Mail Transfer Protocol
- apache - Web server
- ntp - Network Time Protocol
- netscape - Web browser, email and usenet news client

plus many more.

All these (and more) operate by opening sockets (a file like way of connecting to the protocol stack) and then sending and receiving data via these sockets.

Each application will probably break the data it sends into sections and then send each section to the socket which in turn hands the data to the UDP/TCP protocol engine. The socket does not "wrap" the data in an envelope but it does hand the protocol engine information about the destination, the port number, the protocol to use and other detail. The protocol engine then "wraps" the data in either a UDP or TCP packet and hands the packet to the IP protocol engine with the socket information. The top part of IP protocol engine then "wraps" the UDP/TCP packet in an IP packet and hands the packet to the lower part of the IP layer. This uses the IP address and the routing table to determine which device the packet needs to go to. The packet is then handed to the protocol engine that handles the device. In the case of an AX.25 device the AX.25 protocol engine will "wrap" the packet in an AX.25 packet and deliver it to the device driver.

Each of these protocol engines are not only "wrapping" and "unwrapping" the packets as they flow up and down the protocol stack but they are keeping track of the state of the link and doing any error detection/recovery appropriate at that level.

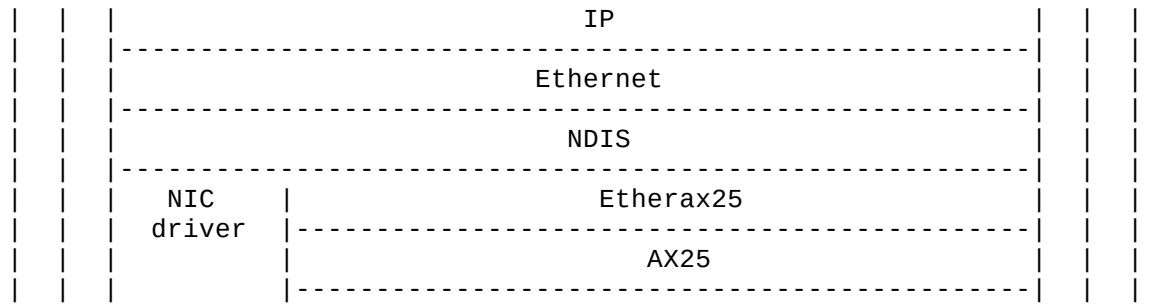
When you get to the device driver (with the exception of the HDLC driver) this is taking the data from the lowest level of the protocol stack and stuffing it into the hardware that gets the data really moving from place to place.

The HDLC driver actually takes the data from the lowest level of the protocol stack calculates the CRC, converts it into the bit pattern that is to be transmitted, does the bit-stuffing that is defined in the HDLC protocol and Manchester encoding the bits. The resultant bit pattern is then handed to the relevant driver that outputs the bits with the correct timing using hardware not designed to be used this way. Even more impressive is that works for receive as well, capturing the bit transitions, undoing the Manchester encoding, finding the framing, unstuffing the bits, checking the CRC and handing a packet up into the protocol stack.

DOS/Win95/98
============
I have a passing acquaintance with Etherax25 as I did study how it was put together. I hasten to add I have never even tried to get it to work as there is no suitable system available here to put it on.

To get the equivalent protocol and hardware stack for these OS's substitute this section into the Linux diagram.

Note: the Baycom/YAM serial/parallel modem drivers do not work under MS-Windows according to the notes.



--

Subject: TCP/IP an explanation, pt 3

3) IP addresses and Hostnames.
==========
> you have to go to this strange dotty-number ident.

Now computers like numbers, humans do not. Humans do not work well with numbers but like nice warm fuzzy things called names (in this context a callsign is a name).

There are a number of ways that the warm fuzzy names can be translated into the numbers that computers understand.

The three most common:

- The "hosts" table.
- Network Information Services (NIS). Was known as Yellow Pages originally but was renamed when BT learnt about it.
- Domain Naming Service (DNS).

In the absence of appropriate entries in a translation service the human has to resort to using the IP addresses raw.

The standard translation service of the Internet is DNS which is also used for the ampr.org domain.

NIS is most commonly used within organisations due to the security implications. However it has the ability to distribute far more than DNS, there are standard NIS tables but it is quite simple to create a new table for a specific function and have the NIS system distribute it along with all the others. NIS has it's place, normally inside a reasonably secure system.

Hostname and Domain names.
==========

A bit of definition here:

Now if I was fortunate enough to actually have a link to the UKIP network the following entry (among others) would appear in the ampr.org domain:

```
g1sog.ampr.org
\---/ \------/
  |      |
  |      now this bit is the domain name
  |
  and this bit is the hostname
```

and the whole thing is a Fully Qualified HostName (FQHN).

Now any host that is a member of the ampr.org domain can be addressed by using just the hostname, the domain name in implicit.

For example:

Assuming two stations g6kui.ampr.org & g1sog.ampr.org.

To establish a telnet session between the two stations would be as simple as running the command on the station of g6kui of:

```
  telnet g1sog
```

The translation of the hostname to the FQHN of g1sog.ampr.org and the translation of that to 44.131.204.66 would be done by DNS. The packets would then be routed to the appropriate interface and transmitted to the first router in the network.

--
Subject: TCP/IP an explanation, pt 4

4) Networks, subnet-works and super-networks.

When the IP addressing system was originally proposed the view was that there would never be a large requirement for network addresses but there would be a need for different sized address spaces.

The solution that was proposed in the Request For Comment (RFC) was for the address range to be segmented into a number of sections.
and each section allocated a letter, I have compressed the sections D onwards in to one range as they are not relevant in this discussion, if you really want to know go and read the RFC's.

```
    0.0.0.0   to 127.255.255.255    A
    128.0.0.0 to 191.255.255.255    B
    192.0.0.0 to 223.255.255.255    C
    224.0.0.0 to 255.255.255.255    D,...
```

Now for the first three classes of network a "network address mask" (commonly known as the netmask) was defined:

Class	Netmask (dotted decimal)	Netmask (number of bits)
A	255.0.0.0	8
B	255.255.0.0	16
C	255.255.255.0	24

You will see IP address quoted in one of two ways:

```
      44.131.204.66/255.0.0.0
or
      44.131.204.66/8
```

The first case is the "IP address/netmask" and the second is the "IP address/number of ones in the netmask from the left hand side"

If someone gives you an IP address without the netmask definition you should assume that the "class" netmask is to be used. As a matter of habit I give the netmask, even if it is the default "class" netmask.

The netmask is used to separate the "network" part of the address from the "host" part of the address. The reason for separating the "network" from the "host" is to allow routing (which I will come to in a later bulletin).

So for: 44.131.204.66

```
the network is : 44.0.0.0
the host is    : 0.131.204.66
```

Nitty, gritty, bitty level

```
--------------------------
Netmask  255.0.0.0     11111111 00000000 00000000 00000000
IP addr  44.131.204.66 00011100 10000011 11001100 00100010
Binary AND             -------- -------- -------- --------
Network  44.0.0.0      00011100 00000000 00000000 00000000

Netmask  255.0.0.0     11111111 00000000 00000000 00000000
ONES complement        -------- -------- -------- --------
Hostmask 0.255.255.255 00000000 11111111 11111111 11111111
IP addr  44.131.204.66 00011100 10000011 11001100 00100010
Binary AND             -------- -------- -------- --------
Host     0.131.204.66  00000000 10000011 11001100 00100010
```

Subnetworks

This, for the A and B class addresses, creates somewhat of a problem for to construct a physical network with 16777216 (2^24), for the A, or 65536 (2^16), for the B, hosts is impractical.

So by changing the netmask a network can be "partitioned" and the partitions (subnets) connected by routers.

So by using a netmask of:
255.255.255.0
my example above becomes:

the network is : 44.131.204.0
the host is : 0.0.0.66

Nitty, gritty, bitty level

```
Netmask  255.255.255.0 11111111 11111111 11111111 00000000
IP addr  44.131.204.66 00011100 10000011 11001100 00100010
Binary AND             -------- -------- -------- --------
Network  44.131.204.66 00011100 10000011 11001100 00000000

Hostmask 0.0.0.255     00000000 00000000 00000000 11111111
IP addr  44.131.204.66 00011100 10000011 11001100 00100010
Binary AND             -------- -------- -------- --------
Host     44.131.204.66 00000000 00000000 00000000 00100010
```

Super-networks

This is going the other way:

Where I have seen this used is in routers, in a companies private network, that are the links between countries where, within each country, the company has a number of sites but there is only one site in each country that links to the other countries.

Starting from the 192.168.x.x address range which by definition has a netmask of 255.255.255.0 reducing the number of bits used in the network address to make a network of more than the maximum for that class.

```
Country A
  Site 1 - 192.168.0.0/24
  Site 2 - 192.168.1.0/24
Country B
  Site 1 - 192.168.16.0/24
  Site 2 - 192.168.17.0/24
Country B
  Site 1 - 192.168.32.0/24
  Site 2 - 192.168.33.0/24
```

Nitty, gritty, bitty level

Netmask for the routers interconnecting the countries
255.255.240.0 11111111 11111111 11110000 00000000

```
192.168.0.0   11000000 10101000 00000000 00000000
192.168.1.0   11000000 10101000 00000001 00000000
192.168.64.0  11000000 10101000 00010000 00000000
192.168.64.0  11000000 10101000 00010001 00000000
192.168.128.0 11000000 10101000 00100000 00000000
192.168.128.0 11000000 10101000 00100001 00000000
```

I leave it as an exercise for the reader to extract the network and host address parts.

--
Subject: TCP/IP an explanation, pt 5a

5) Routing.
==========
In this, multi-part (sorry, you will have to put it back together yourself), ramble I am dealing only with two port routers; three port and more are only an extension of the principles of the two port case.

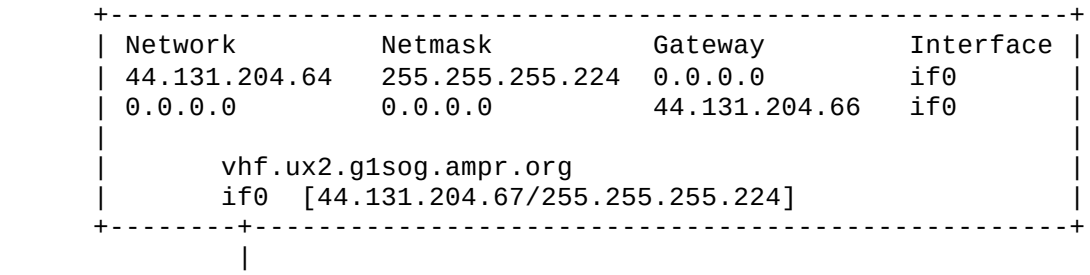
Also I am discussing here IP in a cabled system. When implementing IP in a radio network if the subnet cells are small and there are no hidden stations then the situation is identical. If there are hidden stations then steps must be taken to remove the attendant problems. One method is to imitate the "hub" of the cabled system in some way and communicate only with the hub. Those who currently run parts of the UKIP network are far better placed than I to detail what approaches are in use and how well each works.

In the diagrams:

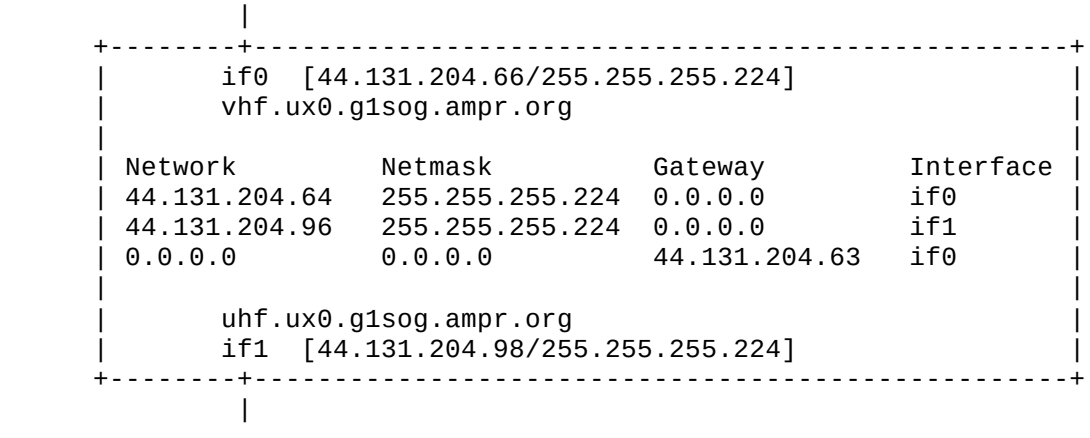
This is a network:
```
 |----------------------| 44.131.204.0/255.255.255.0
```
Best visualised as a 10base2 (Thinnet) coax LAN segment complete with

50R terminators.

This is a host:



This is a router:



The sections inside the boxes:
<some dotted name>.ampr.org
if<n> [<IP address>/<netmask>]
are the fully qualified hostname, interface name and interface IP address/netmask.

and the section headed:
"Network Netmask Gateway Interface"
is the routing table.

Both host and router have a routing table. The only real difference being that the router has two or more interfaces whereas the host has only one.

The routing table details to which interface network traffic is to be sent and to what address it goes.

The magic number 0.0.0.0 has three meanings:
in the "Network" field is the "default" route for that router, if the required network does not match any of the defined routes send it to the router defined in the "Gateway" field.
in the "Netmask" field means ignore the netmask, only found on the default route.
in the "Gateway" field means the given interface on this router.

So, on to some routing examples and more of my infamous diagrams:

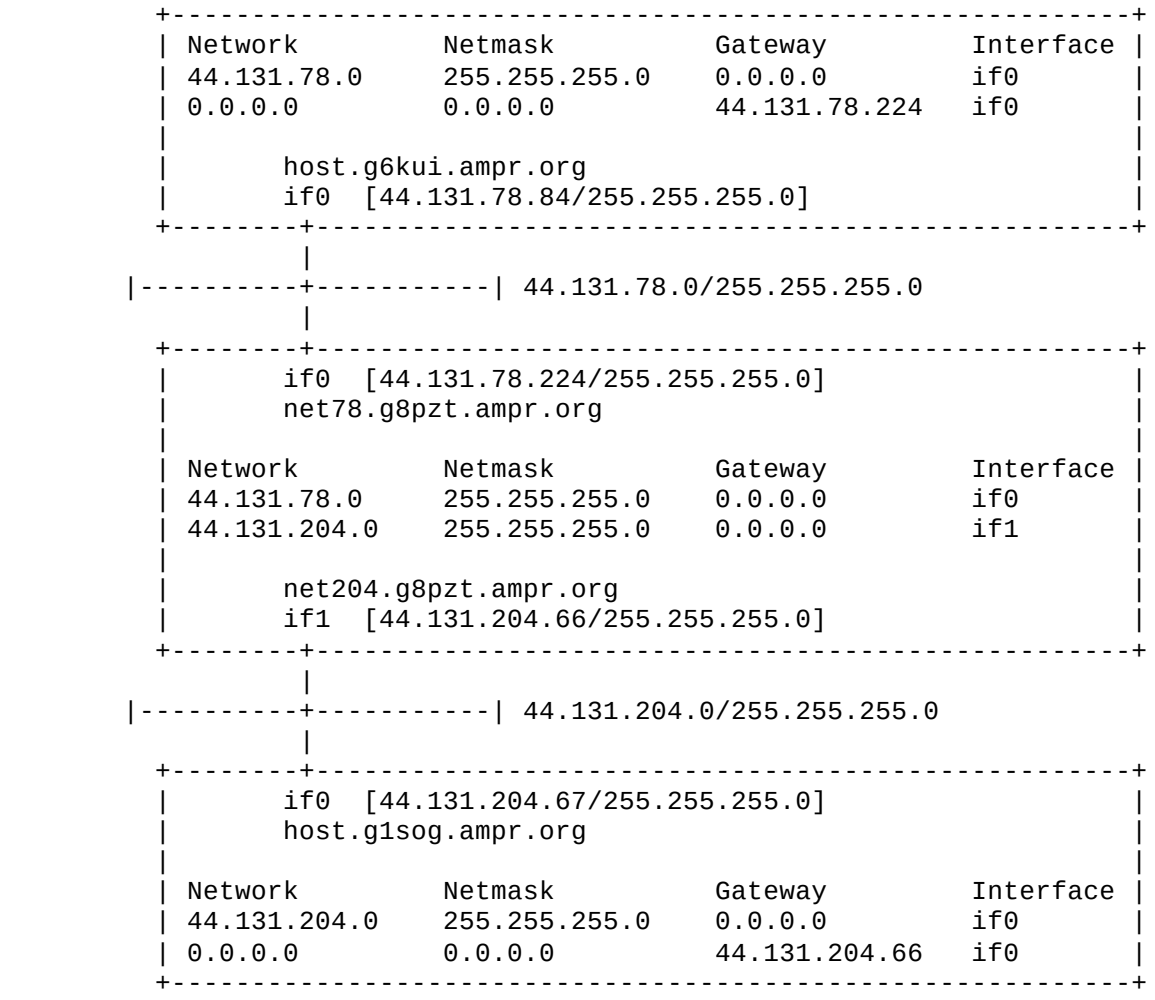
--

Subject: TCP/IP an explanation, pt 5b

==========

Diagram 1

I have taken some liberties here and for this example:
1) G6KUI has a host station host.g6kui.ampr.org with an IP address of
   44.131.78.84/255.255.255.0
2) G1SOG has a host station host.g1sog.ampr.org with an IP address of
   44.131.204.67/255.255.255.0
3) The connecting router station is operated by G8PZT and has two
   interfaces net78.g8pzt.ampr.org and net204.g8pzt.ampr.org



Now each of the hosts has a route for their own local subnet and a default route for all other traffic to their local router.

The router has an entry for each of the subnets. I have not added a default route in the router because of the limited nature of the example but it would be normal for the router to have a default route for all traffic that is not handled by the explicit entries.

If either host has a packet for another station on their own subnet it will be sent without involving the router. However if the packet is not for the local subnet it is sent to the router.

The distinction is made by which destination MAC (Media Access Control) address is used. The Address Resolution Protocol (ARP) is used to map the

IP address to the required MAC address.

For a local subnet packet an ARP request is made for the MAC address of the destination host. The packet is then sent with the MAC address and the IP address of the destination host.

If the routing table indicates that a router is involved, i.e. the IP address of the destination is not in the local subnet, then the ARP request is made for the MAC address of the appropriate router. The packet is then sent with the MAC address of the router and IP address of the destination host. The router then repeats the process by looking in it's routing table to see if the packet is in a local subnet or must be passed to another router.

So the only knowledge a host station requires is the first step in the route, once the packet is in the network the tables in the routers determine the path the packet will take.

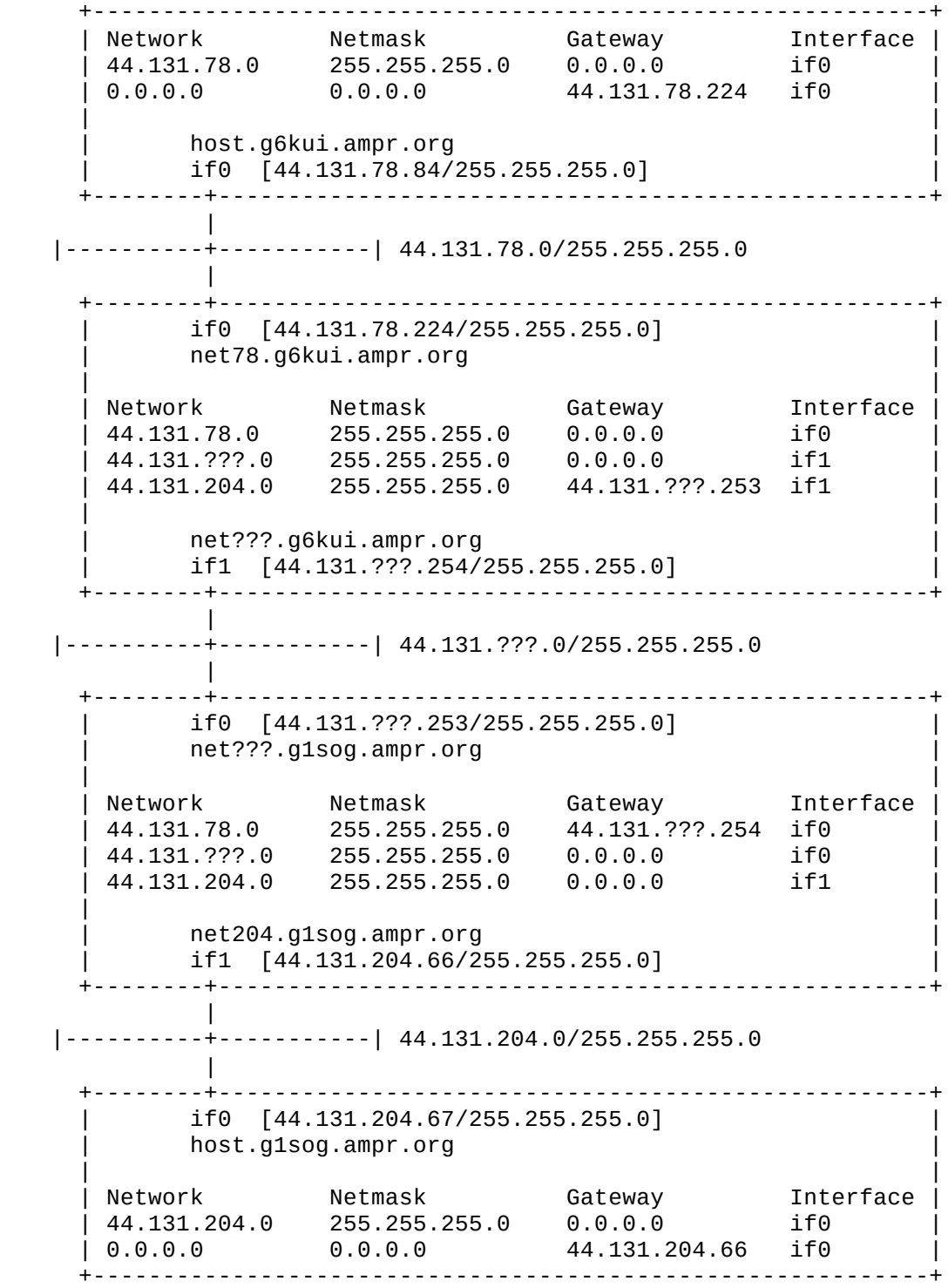
--
Subject: TCP/IP an explanation, pt 5c

==========

Diagram 2

Now I take some more liberties and have changed the assumptions to:
1) G6KUI has a host station host.g6kui.ampr.org with an IP address of
   44.131.78.84/255.255.255.0
2) G6KUI has a router station with two interfaces:
   net78.g6kui.ampr.org 44.131.78.224/255.255.255.0
   net???.g6kui.ampr.org 44.131.???.254/255.255.255.0.
3) G1SOG has a host station host.g1sog.ampr.org with an IP address of
   44.131.204.67/255.255.255.0

4) G1SOG has a router station with two interfcaes:
net204.g1sog.ampr.org 44.131.204.66/255.255.255.0
net???.g1sog.ampr.org 44.131.???.253/255.255.255.0.



Points to note here:
the routing tables of the end point hosts does not change.
due to the default route in each host routing to hosts in
44.131.???.0/255.255.255.0 is automatically present.

--
23.2.2002