# THE *HUB 5/29* IP ROUTING EXPERIMENT

by Paul Overton, GOMHD @ GB7MHD (paul @ g0mhd.demon.co.uk)
and Ian Wade, G3NRW @ GB7BIL (g3nrw @ dircon.co.uk)

**This paper summarises the disadvantages of default IP routing, which often leads to total traffic loss when attempting to forward over long distances. This is followed by a description of how to set up a hub routing scheme that overcomes these problems. An experimental scheme along these lines has been successfully implemented in Regions 5 and 29 in the UK (hence the Hub *5/29* in the title of this paper).**

## Introduction

For the purposes of allocating IP addresses, the UK is split into geographical regions based on county boundaries. IP addresses are of the form 44.13 1 .*RR.SSS*, where RR is the region number and *SSS* is a station address within the region. Regional coordinators typically issue addresses in ascending numerical order on a first-come first-served basis, with little consideration for network planning. It doesn't matter where in the region a station is located, or who its neighbours are, or how to forward traffic within the region. Each new station gets the next available address, and that's that.

With the rapid increase in TCP/IP activity over the last year or so, maintenance of IP routing tables has become a real problem. Unless users regularly update their hosts files, and keep an eye on how the network is changing around them, it soon becomes unrealistic to maintain suitable IP routing tables. As a result, reliable message forwarding and file transfer over more than two or three hops have now become virtually impossible for most people.

It thus became evident that two things were urgently needed to overcome these difficulties:

1. a means of keeping users up-to-date with network host addresses.
2. a simple algorithm for setting up the IP routing tables.

The first of these requirements has been met with the introduction of Domain Name System (DNS) servers. DNS servers hold a more-or-less complete list of AMPRnet host names and IP addresses, so that ordinary end users don't need to maintain their own *domain.txt* file. Whenever a user says **telnet zzz** or **ftp** zzz, their system interrogates the local DNS server for the IP address of station **zzz**, and thereafter uses the address provided by the server to make contact with zzz. The big advantage of this scenario is that ordinary users only need a *very* short *domain.txt* file, and they automatically track any changes to **zzz**'s IP address.

The second requirement, to provide a simple algorithm for setting up the IP routing table, is the main subject of this paper.

## The problem

In principle, IP routing is very simple. All you need to do is set up the routing table to send traffic in roughly the right direction, hoping that the next station down the line will do the same. And the next station does the same. And the station after that does the same....

Trouble is, in addition to specific routing table entries for known destinations, most people have a default catch-all routing table entry (set up by the command **route add default tnc0**, for example) to forward any traffic which doesn't match the specific routes. Because the default route doesn't include a gateway, traffic for unknown destinations will simply be forwarded to any station which happens to be in immediate radio range, and may therefore go off in completely the wrong direction. Eventually, unless you're lucky, it's probable that the traffic has nowhere else to go, and will finally evaporate in that big bit bucket in the sky!

To prevent this happening, what you need to do is route any traffic addressed to unknown destinations through specific gateways. The trick is to define an ideally small set of **route add** commands which will handle this scenario.

This is where the new hub organisation comes in. With hubs, we have a structured, well-defined network which is easy to understand, and where everyone knows exactly how to forward traffic on to the next station in the network. Routing tables are very short and simple, and assuming that everyone follows the rules it's possible to forward traffic over large distances with very little effort.

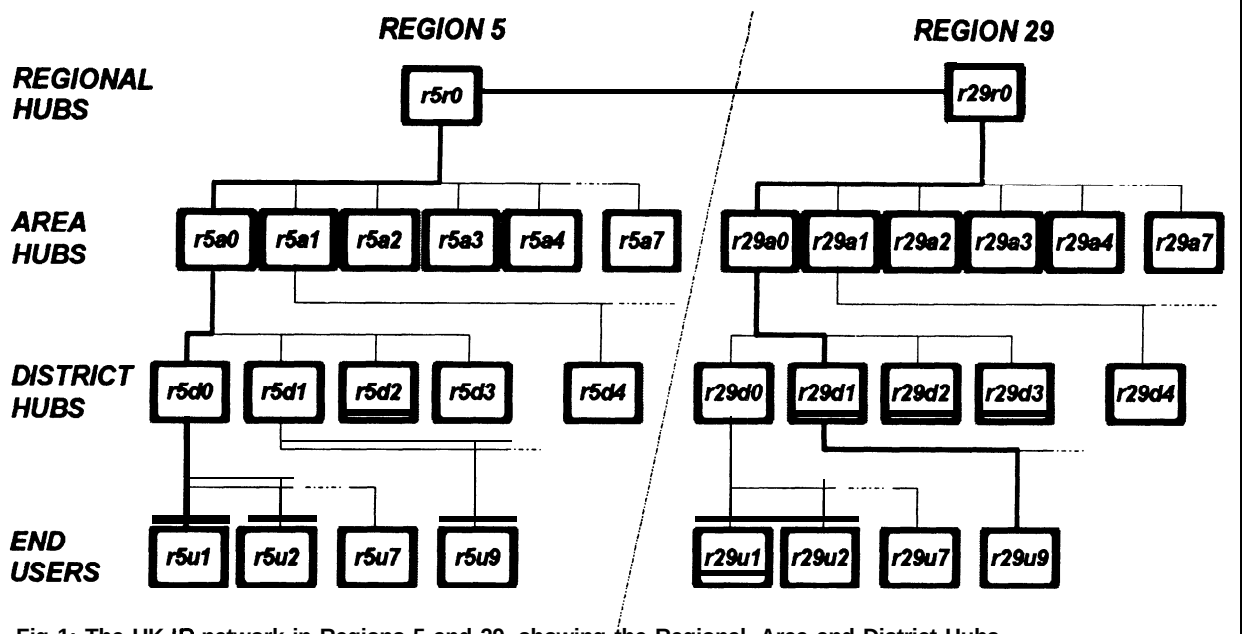


**Fig 1: The UK IP network in Regions 5 and 29, showing the Regional, Area and District Hubs. The thick black line shows the route between End User stations *r5u1* and *r29u9*.**

### How do hubs work?

In the United Kingdom, Region 5 covers the counties of Bedfordshire and Northants, and Region 29 covers Cambridgeshire (see Fig 1 above). In general terms, the Region **5/29** network looks like the following:

Starting at the top of Region 5, the Regional Hub station is called **r5r0** (i.e. region **5** regional hub 0).

Underneath the Regional Hub there are 8 Area Hubs: **r5a0** to **r5a7** (region **5** area hub 0 to region 5 area hub 7).

Underneath each Area Hub there are 4 District Hubs; for example, District Hubs **r5d0** to **r5d3** sit under Area Hub **r5a0,** District Hubs **r5d4** to **r5d7** sit under **r5a1,** and so on.

Finally, there are 7 end users per District. That is, users **r5u1** to **r5u7** are in District **r5d0,** users **r5u9** to **r5u15 are** in District **r5d1,** and so on.

A similar arrangement exists for Region 29.

### Sending packets from one region to the other

To see how the hubs work, let's follow the route that traffic takes from user **r5u1** in Region 5 to user **r29u9** in Region 29.

Starting at **r5u1,** the **traffic** passes first to its District Hub **r5d0,** then upwards to Area Hub **r5a0,** then to the Regional Hub **r5r0**.

Hub **r5r0** then forwards the **traffic** across to Region 29's Regional Hub **r29r0,** and from there it trickles down through Area Hub **r29a0** and District Hub **r29d1,** which forwards it to its final destination **r29u9**.

### The Routing Tables

Nothing remarkable so far. Now let's look at what is needed in the routing tables of all the stations traversed between **r5u1** and **r29u9**.

**At User r5u1: An easy** one here. In this model, there is only one way for *all* **traffic** to go — upwards to district hub **r5d0**. So to set up the routing table at **r5u1,** all we need is:

```
route add default tnc0 r5d0
```

(assuming the interface name is **tnc0)**

That's the complete routing table.

In other words, the routing table for every end user consists of just one entry; **a** default entry which routes all **traffic** to the end user's District Hub.

**At District Hub r5d0**: At the District Hub we have to cater for two alternatives:

1. We forward **traffic** back downwards to another end user in the same district, or
2. We forward traffic upwards to the Area Hub.

To handle the first alternative, we could have 7 separate **route add** statements, one for each end user in the district. However, by choosing IP addresses carefully, we can reduce this to just one **route add.**

For example, we can allocate the users in this district a block of addresses in the range 44.131.5.1 to 44.131.5.7. Expressing these addresses in binary:

```
        44  .  131  .  5  .  x
r5u1: 00101100.10000011.00000101.00000001
r5u2: 00101100.10000011.00000101.00000010
r5u3: 00101100.10000011.00000101.00000011
r5u4: 00101100.10000011.00000101.00000100
r5u5: 00101100.10000011.00000101.00000101
r5u6: 00101100.10000011.00000101.00000110
r5u7: 00101100.10000011.00000101.00000111
      <---------- 29 bits ---------->
```

Looking carefully at these addresses, we see that the first 29 bits are identical. It's only the last 3 bits which differ.

So to set up a routing table entry to forward downwards to these 7 users, all we need to do is set up a 29-bit routing mask:

```
route add 44.131.5.0/29 tnc0
```

This means that all traffic addressed to any station whose address matches the first 29 bits of 44.131.5.0 is forwarded through interface **tnc0**. Because there is no IP gateway included in the **route add** command, the end users have to be in direct radio range of the hub.

And for all other traffic? All we need is a second **route add** command to forward it upwards to the Area Hub **r5a0**:

```
route add default tnc0 r5a0
```

Thus each District Hub needs just **two route add** commands: one command containing a 29-bit mask to forward traffic down to other users in the same district, and a default command to send the remaining traffic up to the Area Hub.

**At Area Hub r5a0**: The story continues along the same vein. At the Area Hub, we must cater for two alternatives:

1. We forward traffic back down to another District Hub in the same area, or
2. We forward traffic upwards to the Regional Hub.

To handle forwarding back down to another District Hub, we again choose our IP addresses carefully. We've already seen that District Hub 0 (**r5d0**) handles the following end user addresses:

```
r5u1: 00101100.10000011.00000101.00000001
                        to
r5u7: 00101100.10000011.00000101.00000111
```

If we now give the District Hub **r5d0** the address 44.131.5.0, its binary equivalent is:

```
r5d0: 00101100.10000011.00000101.00000000
      <---------- 29 bits ---------->
```

That is, the first 29 bits of the District Hub address are the same as the first 29 bits of all addresses below the hub.

So, to forward from the Area Hub down to District Hub 0 and any stations 'below it, all we need is the command:

```
route add 44.131.5.0/29 tnc0 r5d0
```

Similarly, if we choose the following addresses for the remaining 3 District Hubs:

```
r5d1: 00101100.10000011.00000101.00001000
r5d2: 00101100.10000011.00000101.00010000
r5d3: 00101100.10000011.00000101.00011000
```

then all we need in the routing table for these District Hubs is:

```
route add 44.131.5.8/29 tnc0 r5d1
route add 44.131.5.16/29 tnc0 r5d2
route add 44.131.5.24/29 tnc0 r5d3
```

Once more we need a default route upwards to the Regional Hub for all other traffic:

```
route add default tnc0 r5r0
```

Thus each Area Hub needs just four **route add** commands for the District Hubs below it, and one default command to route remaining traffic up to the Regional Hub.

**At the Regional Hub r5r0:** We've reached the top of Region 5. Once again, we have to consider two alternatives:

1. We forward downwards to one of the 8 Area Hubs in the same Region, or
2. We forward across to the Regional Hub of Region 29.

By now, the pattern is clear. To forward downwards, we need 8 **route add** commands, one for each of the Area Hubs. The Area Hubs cover the following end user addresses:

```
r5a0:   00101100.10000011.00000101.00000000   (44.131.5.0) to
        00101100.10000011.00000101.00011111   (44.131.5.31)

r5a1:   00101100.10000011.00000101.00100000   (44.131.5.32) to
        00101100.10000011.00000101.00111111   (44.131.5.63)

r5a2:   00101100.10000011.00000101.01000000   (44.131.5.64) to
        00101100.10000011.00000101.01011111   (44.131.5.95)

r5a3:   00101100.10000011.00000101.01100000   (44.131.5.96) to
        00101100.10000011.00000101.01111111   (44.131.5.127)

r5a4:   00101100.10000011.00000101.10000000   (44.131.5.128) to
        00101100.10000011.00000101.10011111   (44.131.5.159)

r5a5:   00101100.10000011.00000101.10100000   (44.131.5.160) to
        00101100.10000011.00000101.10111111   (44.131.5.191)

r5a6:   00101100.10000011.00000101.11000000   (44.131.5.192) to
        00101100.10000011.00000101.11011111   (44.131.5.223)

r5a7:   00101100.10000011.00000101.11100000   (44.131.5.224) to
        00101100.10000011.00000101.11111111   (44.131.5.255)

        <---------- 27 bits ---------->
```

This time we're interested in the first 27 bits of the address. They specify the Area Hub number, so we need the following **route add** commands at the Regional Hub:

```
route add 44.131.5.0/27   tnc0 r5a0
route add 44.131.5.32/27  tnc0 r5a1
route add 44.131.5.64/27  tnc0 r5a2
route add 44.131.5.96/27  tnc0 r5a3
route add 44.131.5.128/27 tnc0 r5a4
route add 44.131.5.160/27 tnc0 r5a5
route add 44.131.5.192/27 tnc0 r5a6
route add 44.131.5.224/27 tnc0 r5a7
```

Finally, we need a **route add** command for traffic going across to Region 29:

```
route add 44.131.29.0/24 tnc0 r29r0
```

(plus, of course, similar **route add** commands for traffic going to other regions).

### The big attraction of hubs

The **route add** commands listed above are the only such commands which are necessary to forward any and all traffic within, into and out of a region. This is what makes the hub scheme so attractive.

With the hub scheme, all you need to know is how to forward to your next-door neighbour. In turn, your next-door neighbour knows how to forward to his/her next-door neighbour, and so on.

Also, because the routing tables at the hubs are very small and will only require infrequent changes, version Xl-G (or later) of **TheNet** is an ideal candidate for building the hubs — with NET/ROM forwarding disabled, of course! You don't need a computer for such a hub, just a TNC with an Xl-G EPROM and a radio.

Another big advantage of hubs is that because the network is consistently structured and all routes are known by everyone, there is no need for routing update broadcasts. This means that we no longer have RIP or RSPF traffic clogging up the network; they just aren't necessary.

so, to summarise:

- End users need *one and only one* **route add** command.
- District Hubs need *just two* **route add** commands.
- Area Hubs need *just five* **route add** commands.
- Regional Hubs need *just eight* **route add** commands for the areas below it, plus one command for each other region it forwards to.

### Don't be silly!

**Question 1**: If I can reach a station in my own district direct, is it really necessary to forward up to the District Hub and down again to my neighbour?

**Answer 1**: Of course not. There's nothing to stop you including more **route add** commands to handle

these special situations. So your complete set of **route add** commands may look something like this:

```
route add r5u2    tnc0
route add r5u7    tnc0
route add default tnc0 r5d0
```

In other words, you can add special routes for all stations which you can reach directly. The only proviso is that it's entirely your own responsibility to maintain these routes, and that the rest of the network may be unaware of them.

Most important of all, you must *always* have a default route *which points to another station.* In other words, this is all right:

```
route add default tnc0 r5d0   ☺
```

but this won't do:

```
route add default tnc0   ☹ <<< WRONG (no gateway specified)
```

**Question 2:** What do I do if I live in Region 5 and want to talk to someone in Region 29 who happens to be just a couple of miles away across the county boundary. Surely I don't have to climb all the way up to the top of the Region 5 tree and then descend all the way down the Region 29 tree to the station I want to talk to?

**Answer 2:** Same as answer 1. So you may have an additional **route add** command like this:

```
route add r29u7 tnc0
```

**Question 3:** What do I do if1 live just outside Region 29 in Region 19, but can reach a Region 29 station direct? Do I have to forward my traffic towards the Region 29 Regional Hub?

**Answer 3:** Same as answer 2. Alternatively, it may make more sense for you to trade in your Region 19 address for a Region 29 address to improve connectivity, even though you don't actually live in Region 29. The regions are defined for administrative convenience only — radio waves don't respect county boundaries, and we're trying to build a workable network here, not to satisfy the bureaucrats or the pedants!

### What you need to do

To participate in the hub scheme as an end user, this is what you need to do:

1. Get yourself a new IP address.
2. Set up your *domain. txt* file.
3. Set up your DNS client (optional).
4. Set up the IP routing table.
5. Disable routing broadcasts"
6. Set up the SMTP gateway for outgoing mail.
7. Set up the POP client to collect incoming mail.

The following sections describe these in detail. Several examples of commands are included — these examples apply specifically to PAOGRI versions of NOS, but they are almost certainly the same (or very similar) for other well-known versions as well. These commands will go in the NOS startup file (called *autoexec.nos* or similar).

Most of the examples contain the parameter <District Hub>. 'Where you see this, substitute the IP hostname-of your own District Hub. So, for example, where you see the command syntax **smtp gateway** <District Hub> and your District Hub is, say, **r5a0**, you will include the command **smtp gateway r5a0** in *autoexec.nos*.

#### 1. Getting a new IP Address

To take advantage of the the new hub routing scheme, you will need a new IP address. Contact your local IP address coordinator for this.

#### 2. Setting up *domain.txt*

Your address coordinator should be able to provide *you with* a suitable *domain.txt* file. However, you will only need this file if you can't access a DNS server.

#### 3. Setting up the DNS Client

If there is a DNS server within two or three hops range, you should set up your station as a DNS client. This means that *you only* need a minimal *domain.txt* file in your own system.

For example, your *domain.txt* could be literally as short as this:

```
$origin ampr.org
r5u1     IN A 44.131.5.1 # my own call
r5d0     IN A 44.131.5.0 # my District Hub

         IN NS r5dns
r5dns    IN A 44.131.5.95 # Region 5 DNS server

loopback IN A 127.0.0.1
```

N.B. The name server can be anywhere — it doesn't have to be in your own region.

Then, to make your DNS client interrogate the DNS server, you'll need this command in *autoexec.nos*:

```
domain addserver r5dns
```

#### 4. Setting up the IP Routing Table

To set up the IP routing table, all you need is a single **route add** command to forward all default **traffic** to your District Hub, plus possibly a handful of other **route add** commands for your immediate neighbours within direct radio range.

**Thus,** in your *autoexec.nos*:

```
route add default tnc0 <District_Hub>
route add r5u9 tnc0
route add r29u23 tnc0
```

where **r5u9** and **r29u23 are** special entries for immediate neighbours.

Note that if you include such special routing entries, your neighbours must similarly set up special entries in their routing tables to point back to you. These special entries are not part of the hub forwarding scheme, so it is up to you and your neighbours to maintain them.

REMINDER: Most important of all, for the hub scheme to work, **the route add default** command ***must*** include a gateway.

#### 5. Disabling Routing Broadcasts

Because the hub routing scheme has a welldefined and consistent addressing structure, it isn't **necessary** to broadcast **IP** routing table updates. Thus you should turn RIP and RSPF off, by commenting out **any rip** and **rspf commands** which you may have in *autoexec.nos*; i.e. precede these commands with a # character.

#### 6. Setting up the SMTP Gateway

You will probably use the Area Hubs as **SMTP** gateways to forward your outgoing mail. Thus you should include **an smtp gateway** command in *autoexec.nos*, defining your mail gateway:

```
smtp gateway <Area_Hub>
```

#### 7. Setting up the POP Client

If you don't want to leave your radio on all the time, you can ask your Area Hub sysop to hold the mail for you until you are ready for it.

To collect the mail, you'll need to set up your POP client, specifying the **popmail** server, the name of your mailbox on the server, and the POP account name/password. Thus in *autoexec.nos*:

```
pop mailhost  <popmail_server>
pop mailbox   r5u1
pop userdata  r5u1 mypassword
pop timer     3600
```

The name and password which you give in the **pop userdata** command must match the entries in the POP accounts file that the Area Hub sysop sets up in his system.

As another example, if you and the Area Hub support POP3, you can include **this** command in *autoexec.nos* (all on one line):

```
popmail addserver <popmail_server> pop3 r5u1 r5u1
                                        mypassword
```

The first **r5u1** is the name of your mailbox at the server. The second **r5u1** and **mypassword are the** POP account name/password pair.

Once this is all in place, all you have to do to collect your mail is to give a **pop kick** command from the keyboard:

```
net> pop kick <popmail_server>
```

### Contacting Region 5 and Region 29 stations from the outside

If you live outside the Regions 5 and 29, and want to communicate with somebody inside, there are three options:

1. **If you** are within direct radio range of the wanted station, set up your routing table for a direct connection. There's :no point in going up and down the hub hierarchy if you don't need to. As explained above, the wanted station should set up a similar entry to point back to you.
2. If you are reasonably close to an Area Hub (say, within two or three hops), forward all **traffic** for Region 5 and Region 29 to that hub.
3. If you live a long way from Region 5 or 29, forward your **traffic** towards the following hubs:

   **dunbbs [44.131.5.120]** for Region 5 destinations

   **mhdbbs [44.13** 1.29. **1]** for Region 29 destinations

For options 2 and 3, you'll need a couple of **route add** commands in *autoexec.nos*:

```
route add 44.131.5.0/24 tnc0 <gateway>
route add 44.131.29.0/24 tnc0 <gateway>
```

(The <gateway> is a next-door neighbour which can forward traffic towards the chosen hub).

### How it has worked in practice

Summarising our experiences in Region 29 (the story for Region 5 being broadly similar), the situation at the inauguration of the new hub routing system was that three of the four proposed Area Hubs were in place and operating. The fourth Area Hub was to have many problems with software etc, but these were fixed after some effort.

### Issuing New Addresses

The first major problem to tackle was that of issuing the new addresses and getting the routing fixed so that we could make use of the new structure. Region 29 started with about 40 listed users, all whom changed their addresses within three days of the hub system starting. The issuing of addresses was based upon the rules as laid out above, but also adding a sub-rule for other radio frequencies as well — this meant that addresses were issued on the basis of location and radio frequency.

Most users were surprised by the simplicity of the changeover. With a few exceptions, the hub system became operational within just two days of the starting date. Some of the more adventurous immediately tried to use the newly available routes, both across the region and some out of the region. Most routes worked well but some were a little slow.

The inter-region link between Regions 5 and 29 worked very well from day one.

### Starting small

The hub system started with only two layers: Area Hubs and end users. This was mainly because there weren't enough stations around or there wasn't enough equipment available to set up District and Regional Hubs. Routing for the users was exactly as set out above. However, because there was no Regional Hub for either region, the routing between the Area Hubs in Regions 5 and 29 (and to other regions as well) was a little more complex. A consequence of this is that there can be no default route for any of the Area Hubs because there are routes in all directions linking with other regions.

Since the hub system started there has been a steady growth in the number of applications for addresses, It was not long before it became necessary to start the first District Hub. In order to prevent the District Hub from overlapping its radio coverage with that of the local Area Hub, it was (decided that the main user frequency of the District Hub would be different from that of the Area Hub. The inter-hub link is on 4m and the user frequency is on 2m, 50 kHz above that of the Area Hub. This works. well, but because of the increasing level of user traffic a higher speed radio link is now being planned. There are also plans for more District Hubs.

Below is an extract from the routing table at Area Hub **mhdbbs.ampr.org**:

```
#Region 29 routes from mhdbbs.ampr.org.

route add 44.131.29.16/29 ax0

#Area Hub 1
route add 44.131.29.32/27 netrom 44.131.29.3 1
route add 44.131.29.64/28 ax0   #VHF user block

#District Hub 1
route add 44.131.29.80/29 ax4 44.131.29.80 1
#Additional 4 addresses for District Hub.
route add 44.131.29.88/30 ax4 44.131.29.80 1

route add 44.131.29.95 node     #route to DNS

#User Subnets
route add 44.131.29.96/30 ax5
route add 44.131.29.100/30 ax3 44.131.29.100 1
route add 44.131.29.104/30 ax3
route add 44.131.29.108/30 ax5 44.131.29.108 1

route add 44.131.29.112/29 ax1  #UHF user block
route add 44.131.29.120/29 ax0  #2nd VHF user block

#Area Hub 4
route add 44.131.29.128/27 netrom 44.131.29.2 1

#Area Hub 5
route add 44.131.29.192/27 netrom 44.131.29.4 1
#Route to Region 5
route add 44.131.5.0/24 ax2 44.131.5.120
```

Some of the interlink routes use NET/ROM as the carrier. This has been done because these routes use the existing network. The use of NET/ROM ensures that the IP network still functions, even if only slowly, when the existing network is busy. Also, most of the NET/ROM routes use radio frequencies which are not generally used for user access.

### Introducing the Mail Hopper

With greater distances now attainable, a new problem surfaced: how to move mail efficiently across the network. It was found that the normal system of point-to-point SMTP sessions was very slow owing to the number of hops that frames were having to go through. MX records were tried, but the disadvantage

was that an MX record was required for each user at every point throughout the network.

So a new mail forwarding technique was adopted, called the *mail hopper*. The hopper applies a route-related approach to mail forwarding, using IP addresses instead of hostnames. This required small modifications to the SMTP client code at each hub: after resolving the destination's IP address, the client now consults the routing table to discover the IP gateway through which the mail is to be forwarded. This simple change allows the mail to be forwarded entirely at the IP level to its destination, thus removing the overhead and maintenance problems of intermediate SMTP gateways. The hopper works very well in practice, and has significantly improved mail transfer times through the network.

### POP mail

Because most users don't run 24-hour hosts, it was decided to implement the POP mail retrieval system throughout Regions 5 and 29 — it had been in use for some time already by a few users, but with the introduction of the hub system it has now become the normal method of retrieving private mail for almost everyone.

### NNTP

The hub system has also presented us with an ideal mechanism for communicating with all users at all locations throughout the network, using NNTP. To do this, all the Area Hubs have now been set up as NNTP sewers and poll each other every hour. Initially the quality of the user software was not good for this function, but the situation has now been improved by the addition of third-party news readers like SNEWS and CPPNEWS. Again, NOS had to be altered to cope with this.

The NNTP service has significantly improved the quality of user support, as messages for help can now be seen and answered by all. This has turned out to be an unexpected benefit — before the introduction of easy-to-use NNTP services, the support burden for the network was shouldered by just two or three people, whereas now the level of expertise is spreading rapidly over the whole of Regions 5 and 29, and beyond.

### The Future

There are now a number of other regions successfully using the hub system in the United Kingdom and slowly we are achieving real network connectivity. We hope to continue this process so that it will be possible to cover the whole country.