NET/ROM NETWORKING

This chapter and the next discuss the packet servers called nodes in more detail than the introduction of Chapter 5. The focus here is the group of node types which are based on the Net/Rom networking protocol. Chapter 10 focus on several other protocols (KaNode, TexNet, FlexNet, ROSE & TCP/IP). The goal of this chapter is to provide enough information so that you can use the Net/Rom packet network effectively. It also will attempt to convey some of the delights and frustrations of using this network.

There is quite a variety of node types which fall in the Net/Rom category. Here, the discussion tries to cover general principles of how they provide connection service. Chapter 25 in Volume 2 provides reference information about specific node implementations.

9.A WHAT IS NET/ROM NETWORKING?

The term Net/Rom is derived from the protocol developed by Software 2000 (apparently no longer in business). This protocol enabled TNCs with a different program memory (ROM) to automatically construct networks, route user connections between TNCs called nodes, and provide connection services.

Now, there are many node software implementations which use this protocol. They include TheNet, G8BPQ, MSYS, & NET/NOS (including JNOS). Networks built around this protocol cover most of the United States and Canada and are in use in many other parts of the world.

So, what is NET/ROM networking?

9.B NET/ROM NODES

Net/Rom nodes do the following things which define this protocol:

9.B.1 Neighbor list: the node maintains a list of directly heard neighbors. Neighbors are recognized by the specific pattern of their beacon. The node stores the neighbor's node name, callsign, the port on which it was heard, and number of possible routes to the neighbor. If a neighbor has not been heard in a specific time interval (called obsolescence time), the entry is removed from the neighbor list. Note that this list is adaptive and changes with changes in the surrounding network. While this results in a network requiring little manual support by the node operator (or anyone else), there are some specific problems which this process produces.

9.B.2 Destination list: the node maintains a list of possible destinations (called the node list). This list includes all neighbors plus all of the destinations which are reported by neighbors and which have an adequately high route quality. The route quality for inclusion in the node list is set by each node operator.

9.B.3 Destination list beacon: nodes beacon the entries in their node lists. This beacon is heard by neighbors and is used to construct destination lists. This shared information is used to construct routes automatically (autorouting).

9.B.4 Construct autorouted circuits: when a connect request to a distant node is received by a Net/Rom node, it begins the process of constructing a circuit to the distant node. The construction of the circuit is based on route quality which is discussed in more detail later in this chapter. Once a circuit is constructed, it becomes a pipeline for packets between the node which started the circuit and the destination node.

9.B.5 Local connection service: when a user is connected to a Net/Rom node, a connection to a local station may be requested. The node will attempt to make the connection, using the user's callsign but with the user's SSID changed to 15-(userSSID).

9.B.6 Other services: the node may provide other services. Standard services include node list, route list, and user list. In addition, bulletin board service, callbook service, etc. may be supplied (see

Chapter 5).

9.C BASIC NODE LIST

Node lists are discussed in detail in several other sections of this chapter. However, in order to interpret what is happening with route quality, we need a little bit of information about how node lists work in Net/Rom-style nodes.

The following TheNet example was chosen specifically because its node lists are small. Sending *N[ode]* causes the node to return a list of known destinations. This list may not be complete, however, because most nodes just return the names of node user-ports. For nodes in which this is true, sending *N[ode] ** causes the node to return a list of all known destinations, including backbone-port names (which begin with #).

```
n
VALLEY:N7FGF-3}  Nodes:
CRS:NV7N-14        EM4:WB6LMA-4       KENO:NV7N-2        KLMT:NV7N-1
LKV:KJ6ZP-1        MFR:WA7FAB-4       OAK:W7EXH-1        RDG:WA6YNG-1
n *
VALLEY:N7FGF-3}  Nodes:
#EUG:W7EXH-12      #EUG4:W7EXH-8      #EUG5:W7EXH-9      CRS:NV7N-14
EM4:WB6LMA-4       KENO:NV7N-2        KLMT:NV7N-1        LKV:KJ6ZP-1
MFR:WA7FAB-4       OAK:W7EXH-1        RDG:WA6YNG-1
```

9.D ROUTE QUALITY

Route quality is crucial to the routing process in a Net/Rom network. It is also one of the most confusing pieces of information available from a Net/Rom-style node.

9.D.1 NEIGHBOR ROUTE QUALITY: At each node, every directly heard neighbor is given a route quality which ranges from 0 to 255 (bigger is better). It is important to recognize that the assigned route quality may not have any relationship to the actual quality of that route at any given moment. Route quality values are assigned by the operator of the node. The node operator may choose to give all stations heard on a particular port a blanket quality value. The node operator may choose to manually assign a unique quality to each and every neighbor. Or the node operator may choose to manually assign unique route qualities to a

few and a blanket value to all of the remaining. Each of these schemes is in use in various areas and it is often an issue which brings on intense debate.

The important fact to note, however, is that the operator chooses the route quality. It is not determined by how easy it is to get packets to or from that neighbor. Thus, if antenna damage, propagation changes, or anything else alters how good a particular route is, unless the node operator changes it, the same value still applies.

Lets look at a Net/Rom style node and see how neighbor route qualities are reported:

```
SALEM:AF7S-1}  Routes:
> 3 K7UYX-1 224 8
> 2 N7IFJ-9 224 2
  1 N7HZT-9 192 2
> 2 KA7AGH-8 224 151
  2 N7TLD-8 224 4
  1 K7MYU-9 192 2
> 2 KA7UPD-1 224 57
> 3 W7VTW-11 224 44
```

For the record, this is from a G8BPQ node which uses Net/Rom protocol. This is a 3-port node and the numbers preceding the callsigns indicate the port on which each of these neighbors was heard. All neighbors on port 3 of this node share a 1.25 meter cluster and have been given route qualities of 224. All neighbors of this node on port 2 share a 70cm, 9600 baud frequency and have been given route qualities of 224. All neighbors on port 1 are on the 2 meter user frequency and have been given route qualities of 192. The remaining numbers and symbols will be discussed later in this chapter.

Lets look at another node which shows some of the other route quality issues. This one is a TheNet backbone port of a two-TNC node stack of the type discussed in section 8.E.2.

```
#ALW:WA7HJV-10}  Routes:
> 1 ALW:WA7HJV-5 255 1
  0 #BOIS3:N7FYZ-6 0 0 !
  0 WB7DOW-13 0 0 !
  0 #YKMSL:KB7HDX-7 0 0 !
  0 W7SC-1 0 0 !
  0 #HOME:KB7CFD-3 0 0 !
  0 KB7DBD-8 0 0 !
  0 K7YLO-2 0 0 !
  0 #ONO2:WB7RES-6 0 0 !
```

```
> 0 #NUKE:K7OJ-7 224 51
> 0 #PDT:N7ERT-10 224 71
> 0 #BKE:W7NYW-9 224 17
  0 #JDY:N7LZM-7 224 17
  0 #SPOKN:WB7NNF-7 224 44
```

With TheNet nodes, there are two places where a neighbor may be heard. Neighbors heard over the radio are port-0 neighbors and a zero precedes their callsign. Neighbors which are other members of the same node stack are port-1 neighbors and a 1 precedes their callsigns. Normally, other members of the same node stack are given quite high route qualities; here ALW:WA7HJV-5 is given a quality of 255. Some radio neighbors are given a quality of 224. Some others are given qualities of 0. This is done in this case because they may be heard only once in a while or, even though heard, may not have an adequate path for normal communication.

9.D.2 DESTINATION ROUTE QUALITY: It is sometimes hard to imagine how destination route quality comes about. The difficulty often arises because of confusion about what comes first. In fact, networks evolve and the route quality entries are a product of that evolution.

To see how this evolution occurs and what its result is, lets consider a very simple case. This simple case involves an isolated node named BIGTWN. Then, nearby (but not too nearby), a node named PODUNK is built to link with BIGTWN. Finally, to improve things, HILTOP is added. Lets see what happens during this progression.

STEP 1 - BIGTWN on line: We will pretend that there is a single, isolated, node named BIGTWN:NA1BC-1 (ie, Bigtown). It plaintively beacons UI packets (see section 4.D.1) from NA1BC-1 addressed to NODES. The beacon says that it is a routing broadcast (with a special character first in the message), then says "my node name is BIGTWN". It says no more because it does not know anything about any other nodes.

STEP 2 - PODUNK on line & beacons: Now, suppose that a group of hams at Podunk decide that they want to talk to their packet friends in Bigtown. So, they put up a node named PODUNK:KB2XYZ-1. Soon after being turned on, PODUNK sends a beacon very much like the ones BIGTWN has been sending. This is because, at this point, it has not yet heard from any other nodes. BIGTWN hears this beacon, and if we were to ask for a node list from it, would get:

```
 n
BIGTWN:AB7BC-1}  Nodes:
PODUNK:KB2XYZ-1
```

And if the BIGTWN node operator had set the neighbor route quality to 192 (not uncommon for this kind of network), the route list from BIGTWN would read:

```
R
BIGTWN:AB1BC-1}  Routes:
  0 PODUNK:KB2XYZ-1 192 1
```

PODUNK was heard on BIGTWN's the radio port, has a quality of 192, and knows the route to 1 destination (itself). The next beacon from BIGTWN now changes because, for the first time, it has an entry in its destination list!

STEP 3 - BIGTWN beacons: BIGTWN beacons again with a UI packet from NA1BC-1 addressed to NODES. The beacon says that it is a routing broadcast (with a special character first in the message), then says "my node name is BIGTWN". But this time, it also says "I have a route to KB2XYZ-1 which is named PODUNK. The neighbor of mine which which has the best quality route to KB2XYZ-1 is KB2XYZ-1 and the route quality by way of that neighbor is 192". The latter may be a bit confusing and a bit redundant. This is, however, the standard set of information.

PODUNK, in turn, hears BIGTWN's beacon. From that beacon, it determines that BIGTWN knows about 2 destinations (BIGTWN, itself, and PODUNK). PODUNK considers the latter a trivial route because it is a route to itself, but counts it, none the less. A user at PODUNK would then see:

```
 n
PODUNK:KB2XYZ-1}  Nodes:
BIGTWN:AB7BC-1
R
PODUNK:KB2XYZ-1}  Routes:
  0 BIGTWN:AB7BC-1 192 2
```

After the next beacon from PODUNK, BIGTWN recognizes that PODUNK also has two routes. BIGTWN's route list changes accordingly.

STEP 4 - HILTOP on line & beacons: Now, after a little use (probably not very much) the users of both BIGTWN and PODUNK discover that the route is not very good. Retry rate is pretty high (which all can see by monitoring). Somebody then notices, a little belatedly, that Big Rock Hill just happens to be directly in the path between the two. After much head scratching and discussion about whether or not it is all worth it (and, of course, it is), the two groups decide to get together and install a node part way between BIGTWN and PODUNK. This new node is HILTOP:W3AZ-1 (Hilltop) After a reasonable amount of preparation and testing, it is installed and the new network looks something like what is shown in Figure 9-1.

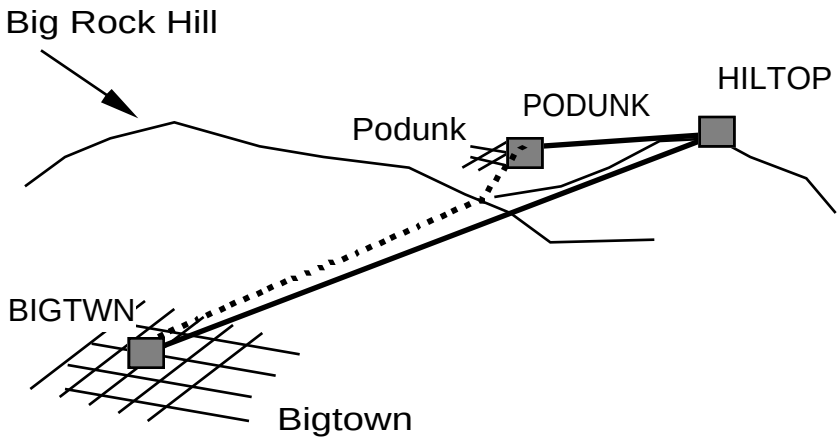

Figure 10-1
BIGTWN-HILTOP-PODUNK NETWORK

The lines in Figure 9-1 attempt to show the poor path directly from PODUNK to BIGTWN and the (expected) good paths from HILTOP to PODUNK and to BIGTWN. What happens when HILTOP goes on the air and beacons for the first time?

Assuming that HILTOP beacons before it hears either of the other two nodes, it is much like when PODUNK went on the air. The first beacon just says that HILTOP is on the air. The beacon is from W3AZ-1 and is addressed to NODES; it says that its name is HILTOP. There is nothing more because it knows nothing of the other two nodes (yet).

Suppose that users were to connect to each of these three nodes very shortly after HILTOP's first beacon. Each would see the following:

```
 n
BIGTWN:AB7BC-1}  Nodes:
HILTOP:W3AZ-1    PODUNK:KB2XYZ-1
R
BIGTWN:AB1BC-1}  Routes:
  0 PODUNK:KB2XYZ-1 192 1
  0 HILTOP:W3AZ-1   192 1

 n
PODUNK:KB2XYZ-1}  Nodes:
BIGTWN:AB1BC-1   HILTOP:W3AZ-1
R
PODUNK:KB2XYZ-1}  Routes:
  0 BIGTWN:AB1BC-1  192 1
  0 HILTOP:W3AZ-1   192 1

 n
HILTOP:W3AZ-1}  Nodes:
R
HILTOP:W3AZ-1}  Routes:
```

Note that HILTOP does not yet know anything about the other two nodes. PODUNK has now heard BIGTWN & HILTOP. BIGTWN has now heard PODUNK & HILTOP. But that is only part of the story.

STEP 5 - PODUNK beacons: If PODUNK happens to beacon next, it now has something to say. Again, the beacon is from KB2XYZ-1 and is addressed to NODES. It says that its name is PODUNK. Since it has heard from both BIGTWN and HILTOP, it says (in effect) "I know the way to BIGTWN which has the call AB1BC-1; the neighbor with the best route to it is AB1BC-1 and that route has a quality of 196. I know the way to HILTOP which has the call W3AZ-1; the neighbor with the best route to it is W3AZ-1 and that route has a quality of 196."

When BIGTWN hears this beacon, things change quite dramatically even though one will not see the effect in the node or route list. Now, BIGTWN has heard HILTOP directly and it has heard that PODUNK knows the way to HILTOP. PODUNK said that its route quality to HILTOP is 192. BIGTWN discounts this route quality according to the quality of the link between itself and PODUNK. Since it gives all neighbors a quality of 192 and 256 is perfect, it determines that the route to HILTOP through PODUNK has a quality of 192 * (192/256) = 144; the first 192 is the quality reported by PODUNK's beacon and the 192/256 is the reduction due to the quality of the BIGTWN-PODUNK link. If a BIGTWN user were now to request information about routes to HILTOP, BIGTWN would return:

```
N IHLTOP
BIGTWN:AB1BC-1}  Routes to HILTOP:W3AZ-1
  192 5 0 HILTOP
  144 6 0 PODUNK
```

Note that it shows a route quality of 192 in the direct link to HILTOP and a route quality of 144 by way of PODUNK. The significance of this difference should become obvious during the discussion of autorouting.

STEP 6 - BIGTWN beacons: If BIGTWN beacons next, it again says that it is AB1BC-1 and is named BIGTWN. It says that it has a destination W3AZ-1 which is named HILTOP, the neighbor which has the best quality route to it is (also) W3AZ-1 and the route qaulity via that neighbor is 192 (only the one highest quality route is reported). It also says that it has a destination KB2XYZ-1 which is named PODUNK, the neighbor which has the best quality route is KB2XYZ-1 and the route quality via that neighbor is 192.

STEP 7 - HILTOP beacons again: When HILTOP beacons a second time, it has heard the beacons from both PODUNK and BIGTWN. Its node list and route list are then filled:

```
 n
HILTOP:W3AZ-1}  Nodes:
BIGTWN:AB1BC-1   PODUNK:KB2XYZ-1
R
HILTOP:W3AZ-1}  Routes:
  0 BIGTWN:AB1BC-1   192 2
  0 PODUNK:KB2XYZ-1  192 2
```

The beacon is now very much like the other two beacons.

STEP 8 - FARWAY comes on line and beacons: To add a little realism to our little fable, lets now suppose that a node named FARWAY:A8ZZ-5 comes on line. It is so far away that only HILTOP can hear it. When it beacons the first time, it, like the others, simply announces it presence. HILTOP hears it and HILTOP's node and route list become:

```
 n
HILTOP:W3AZ-1}  Nodes:
BIGTWN:AB1BC-1   FARWAY:A8ZZ-5    PODUNK:KB2XYZ-1
R
HILTOP:W3AZ-1}  Routes:
  0 BIGTWN:AB1BC-1   192 2
  0 PODUNK:KB2XYZ-1  192 2
  0 FARWAY:A8ZZ-5    192 1
```

But, up to the time when HILTOP beacons next, neither BIGTWN or PODUNK know anything of FARWAYs presence. During this interval, HILTOP is the only node which knows about FARWAY.

STEP 9 - HILTOP beacons: The next beacon from HILTOP begins like all of the others. But there is now a new destination which it reports as A8ZZ-5 (called FARWAY) and the best route to it is the neighbor A8ZZ-1 by which the route is 192. When this beacon is heard by BIGTWN and PODUNK, their node lists change but the neighbor (route) list does not. For example, BIGTWN would show:

```
  n
BIGTWN:AB7BC-1}  Nodes:
HILTOP:W3AZ-1    FARWAY:W8ZZ-5   PODUNK:KB2XYZ-1
R
BIGTWN:AB1BC-1}  Routes:
  0 PODUNK:KB2XYZ-1 192 1
  0 HILTOP:W3AZ-1   192 1
```

Notice that, for the first time, there is an entry in the node list which is not also present in the route list. Suppose that, like one of the earlier steps, a user were to ask the route to FARWAY:

```
R  FARWAY
BIGTWN:AB1BC-1}  Routes to FARWAY:W8ZZ-5
  144 5 0 HILTOP
```

BIGTWN reports a route quality of 144 because it heard that HILTOP had a route quality of 192. BIGTWN reduced the route quality to that distant node then by 192/256 for the same reason as in step 5. And if asked, PODUNK would show the same route quality and neighbor which knows the way.

STEP 10 - BIGTWN and PODUNK beacon: The last step we will observe in this evolutionary odyssey is after both BIGTWN and PODUNK have beaconed. Their beacons both include this information: there is a destination W8ZZ-5 named FARWAY; the neighbor with the best route is W3AZ-1 and the route quality via that neighbor is 144. After this pair of beacons, none of the node lists change but request for the route to FARWAY from BIGTWN now shows:

```
R  FARWAY
BIGTWN:AB1BC-1}  Routes to FARWAY:W8ZZ-5
  144 5 0 HILTOP
  108 3 0 PODUNK
```

Note that BIGTWN now thinks there are two routes to FARWAY. One is via the neighbor HILTOP. The other, and lower quality one, is via PODUNK (from which the route would next to to HILTOP, also). This latter route has been further devalued to 108 because PODUNK reported that the route quality there was 144. BIGTWN then reduced that by 192/256 because of the 192 route quality between PODUNK and BIGTWN.

We might continue to track how routes and their qualities evolve as more are added to the network and as the architecture of the network evolves from this single-frequency system into a backbone system. This is enough, however, because all of the principles have been discussed.

9.E. AUTOROUTING

Suppose that a user (at BIGTWN) in our preceding fantasy were to ask to be connected to PODUNK. The routing server in the node is activated. It asks "which neighbor has the highest quality to PODUNK?". To find the answer, it scans its destination table and finds that there are two routes to PODUNK. The highest quality is directly to PODUNK with a quality of 192. The next best one is via HILTOP with a quality of 144.

The router tries to make the direct connection first. Lets suppose that it is not successful, since Big Rock Hill is in the way. There are so many retries that it fails before making the connection. Then, it tries the next best route, via HILTOP. Because the route is much better, the link to HILTOP is made very readily. Then HILTOP attempts to continue the route to PODUNK; the route to that destination is PODUNK, itself. The link is good, and the connection is readily made. As soon as all of this is complete, the originating node then notifies the user of the successful connection.

If the direct route from BIGTWN to PODUNK had not been so bad, BIGTWN might have been successful with the direct request. But what happens if conditions worsen and the link is no longer good enough to sustain operation? The Net/Rom protocol is designed to seek out an alternate route. In this case, it would attempt to remake the connection via HILTOP. Observation of this process, however, suggests that it does not work very well!

9.F. MANAGED ROUTE QUALITY

In section 9.E, we saw the consequences of route quality which does not reflect the real-world quality of routes. Some node operators take the view (with some justification) that a moderate degree of route quality management can improve the operation of some networks.

In the preceding example, proponents of managed route quality would have manually set the quality of the link from (from either PODUN or BIGTWN) to HILTOP to 192 while setting the quality of the direct path between them to, say, 120. When BIGTWN is asked the route to PODUNK (which is the same thing the router would do), it would return:

```
R  PODUNK
BIGTWN:AB1BC-1}  Routes to PODUNK:KB2XYZ-1
  144 5 0 HILTOP
  120 3 0 PODUNK !
```

Note that the exclamation mark indicates that the PODUNK route has been manually set while routes involving HILTOP take on the route quality designated for all other routes.

With this change, the route to PODUNK via HILTOP would have the highest quality and the router would always try it the first time. The poor quality direct route would only be attempted if the first one were to fail. If HILTOP were to go off the air, there would still be the direct path. There being no alternate routes, it would always be attempted as it was before HILTOP came into service.

This is a fairly modest level of management. Some would argue, however, that Net/Rom works best when it is allowed to function without human tinkering. This debate, and its variations, has no real answer and you may see versions of it arise in various situations.

9.G. THE LOST NODE PROBLEM

Suppose that one of the nodes in the previous example suddenly disappeared. Perhaps power failed; maybe it was a transmitter or TNC.

What happens to the autorouting?

There is a timeout on all the route table entries (called obsolescence counter). If no word is heard from a node in a specific time, it is removed. The problem arises before that time has run out on all the nodes. Prior to that time, a node may broadcast its node list. That list will contain the entry for the lost node because its time has not run out. That broadcast will reset the timers in other nodes which hear that broadcast, at least as far a routing THROUGH those nodes is concerned. Thus, it may take quite a while for a node entry to finally disappear.

The interim time is handled by a network parameter known as time-to-live. When a connect request is send out by a node, the time-to-live counter (contained in the packet header) is set to a predetermined value. Each time the packet passes through a node, the counter is reduced by 1. When the counter reaches 0, the node where it is simply discards it! This keeps packets from forever circulating through a network, looking for a node which cannot be reached.

To see how this works, suppose that HILTOP were to experience a power failure. Then, further suppose that shortly after this power failure, a user at BIGTWN requests a connection to FARWAY. This node is still in the destination lists of both BIGTWN and PODUNK. BIGTWN attempt to make the connection via HILTOP, without success. It finds from its destination list that the next best route is via PODUNK. So, with some difficulty, it connects to PODUNK. The PODUNK destination list shows that the best route to FARWAY is via HILTOP and it, also, tries in vain. At PODUNK, the next best route is via BIGTWN! Connections keep getting made back and forth until the time-to-live count in the connect request reaches zero and is discarded.

Suppose, now, that PODUNK makes a route beacon before the obsolescence counter of HILTOP times out. It broadcasts the fact that it has a route to HILTOP and FARWAY. At BIGTWN, the neighbor entry of HILTOP will time out but the beacon from PODUNK is newer and that one still says that there is a way to FARWAY. Thus, this entry may persist longer than HILTOP's!

9.H. WHY AUTOROUTING DOESN'T ALWAYS WORK

The previous section suggested several reasons why a connect request

to a distant node might not work. There is, however, another reason and it is quite subtle.

Suppose that you request a connection to a node which is more distant than the time-to-live number of your node? Perhaps you are in Ellensburg, WA and want a connect to SNOW (Salt Lake City, UT). There are about 9 intermediate nodes between the two. If time-to-live is set at 6, the connect request will get lost and never acknowledged! It may take a long time, if there are alternate routes (as there are in this example), for the failure message to appear. There are definite strategies to deal with this situation and they are discussed in the section 9.K.

9.I. NODE NAMES

A word may be in order concerning node names. In the U.S. West, the historical practice was to use airport/weather bureau designations. Thus, there were node names like SEA (Seattle), PDX (Portland), BOI (Boise) & SFO (San Francisco). That practice still continues. A problem arose as the number of nodes grew: there weren't enough of these names. Thus, we have names associated with geographical features (HEBO = Mt. Hebo, near Tillamook, OR and KING = King Mtn, near Wolf Creek, OR), names associated with callsigns (RLIMB = W0RLI's MailBox) and full town names (BOISE). In Alberta, A common practice is to use the prefix AB on node names (as in ABHAT, Medicine Hat, Alberta).

Another group of node names are associated with TCP/IP nodes (see Chapter 10 for more information). In much of the Western U.S., a common practice is to use a name which is the hex version of the last three sections of the stations IP address. Thus, for the node 74002C, the last part of its IP address is .116.0.44. In British Columbia & Alberta (and many other parts of the U.S.), the TCP/IP practice seems to be to use names like IPxxx or xxxIP.

9.J. ROUTE LISTS:

The route quality aspect of route lists has already been discussed in detail. Lets look now at some of the other information contained in a Net/Rom route list.

Route lists gives a list of a node's immediate neighbors. In this case, immediate neighbor means a node which it has heard first hand without intervention by any other station. An example for a G8BPQ node looks like this:

```
SALEM:AF7S-1}  Routes:
> 2 KA7AGH-8 224 109
> 3 N7DXT-10 224 30
  3 W7XI-13 224 26
  3 W7BH-11 224 12
  3 W7VTW-11 224 21
  3 KA7UPD-1 224 26
  2 N7KOJ-1 224 11
  3 K7UYX-1 224 30
  3 KA7KAJ-11 224 18
> 2 N7IFJ-9 224 2
```

The style for this list varies a fair amount with node types. See Chapter 25 of Volume 2 for details. But in this case, the neighbors are listed by callsign (some show only name, others show name and callsign). The number following the call is the route quality number. The last number is essentially the number of destinations in that neighbors destination list (all entries in the neighbor's node list, plus itself).

There is a > symbol in the first column for some neighbors. This indicates that the link to that node is in use or has been in use very recently.

The first number column has two meanings, depending on node type. If that column contains only 0's and 1's, a 1 means that the node is connected to the one you are on with a wire; a 0 indicates that it was heard over a radio. If, as in this case, there are numbers like 1,2,3, the numbers refer to port numbers and the number tells on which port the node was heard.

MSYS nodes give a route list which looks like the following. It provides headings for each column and indicates the time in hours and minutes since a route beacon was last heard from each neighbor. There is no link activity indicator.

```
r
#SRA:N7IFJ-9}  Routes:
Port   Neighbor Node Call  Quality Dests Heard
Digipeater(s)
   0      SALEM:AF7S-1      192    136   02:42
   0     WASHCO:N7TLD-8     192    130   03:26
Node cmd?
```

JNOS nodes have a route list which is like this. The style is much like MSYS with some changes, of course. There are link activity indicators. JNOS uses named ports rather than numbered ones. There are two additional pieces of information. One is whether or not the neighbor is implements G8BPQ additions to the Net/Rom protocol as shown by the (BPQ) symbol. It also shows an obsolescence count. This count is commonly set to 6 each time a routing beacon is heard and is periodically reduced; the smaller this value is, the older the information.

```
nr
Routes :
   Neighbour              Port  Qual Obs Dest
  JNOS40:WG7J-11   (BPQ) 2m     192  5    1
>    CRV:K7UYX-1   (BPQ) 2m     128  5  108
  CRVBBS:WG7J-1          2m     192  5    1
  1A0008:KB7IRS          2m     192  6    1
Area: ka7ehk >
```

9.K. WORKING OVER DISTANCES

If you can't readily connect to a node a long ways away, how can you carry on conversations over a real distance? It isn't easy, but it can work. You need persistence and a little knowledge about the network (which hopefully was provided in the earlier sections of this chapter).

You do need to be prepared for the following difficulties: unexpected disconnects, slow turn-around, networks which change, and unpredictability. Now that's an amateur radio challenge if ever there was one!

How do you do it? The first secret is to connect to nodes at intervals of 2-4 hops. This gets past the time-to-live dilemma described in the section 9.H.

The second secret is go get, or make your own, map of the part of the network you are interested in. Earlier parts of this chapter have described how you can determine which nodes are neighbors to which. Most nodes respond to an *I* command; this returns node information. Nodeops have been much better in the last few years about putting useful information here. In particular, you need to look for location information if you don't already know. You may have to rely on deduction and guesswork based on the node name if you don't get clear

help.

The third hint is to do your longer distance work when the network is being lightly used. Late at night seems to be a good choice. So do early Saturday and Sunday mornings. Best seems to be after midnight, Sunday night.

One hint which reduces the activity on your behalf in nodes is this: when you encounter NetRom style nodes with names beginning with #, they are usually backbone ports. If you are using that node as an intermediate point just passing through, connect to it instead of the user port. There is a possible trap in this strategy. One of the versions of TheNet permits the nodeop to disallow connect requests FROM the backbone port of the node to another backbone port. Fortunately, this option is rarely chosen. Unfortunately, the node does not tell you that it is refusing the request!

The author and others in western Oregon have been able to carry on conversations with packeteers throughout the Northwest using these techniques. Join us!