Thoughts on the Issues of Address Resolution and Routing in Amateur Packet Radio TCP/IP Networks

Bdale Garbee, N3EUA

ABSTRACT

The current KA9Q software includes a technique for automatic address resolution, but does not include automatic routing features. The difference between routing and address resolution is explained, and several concerns for on-air automatic routing algorithm implementations are mentioned.

Background:

One of the major goals of those involved in the TCP/IP project begun by Phil Karn KA9Q has been to develop a network implementation that automatically handles routing. By this, we mean that it should not be necessary for an end user to have any knowledge of the topology of the network he is using in order to perform normal "day-to-day" network activities. The user's involvement should extend only to requests of the form "get file from host < n >" or "send mail to user <n > at host <m >".

Several groups are currently working on software that implements one or another form of automatic routing. The NET/ROM product requires that the user only know the nearest NET/ROM site to both the originating and destination stations in order to establish a circuit between the two. The TEXNet system reportedly handles traffic between its endnodes in a similarly transparent fashion. But in both of these cases, the user must still be aware of some aspects of the network topology above and beyond knowing the hostname of the destination site.

What we have right now isn't *good enough!*

ARP, the Address Resolution Protocol:

In his TCP/IP package, Phil Karn has included an implementation of the ARP Address Resolution Protocol, which was originally created to allow for automatic mapping of 3%bit IP addresses into 48-bit Ethernet addresses on a local wired subnet. ARP works by having the originating station issue an "ARP Request" whenever it wishes to converse with a host whose IP address is known, but whose "physical address" is unknown. An important thing to remember about ARP is that it was designed for subnets where all hosts can hear each other directly.

In the Amateur Packet world, the physical address is equivalent to the destination station's callsign and optional sub-station identifier, or SSID. If and when the destination station hears the broadcasted request, it knows the IP address and physical address of the sending station by examining the contents of the request packet, and it responds by sending an "ARP Reply" packet directly to the originating station.

The value of ARP is that the only piece of information about a station that you need to know ahead of time in order to make contact with them is their IP address, as long as they are within RF range. ARP performs very well at the task it was designed for, which is simple mapping of logical address into physical address.

The State of Routing:

There are two simple situations that demonstrate the limitations of ARP, and will serve to distinguish address resolution from routing. Let us consider the IP Switch, and the Digipeater.

A digipeater is a device that receives a packet that has been source routed through it, and echoes that same packet on the same or a different channel, changing the packet only by marking it as "having been

digipeated by **this site".** Our interest in digipeaters stems only from their current popularity, and resulting availability. In order to make use of a digipeater with the KA9Q Internet software, both the source and destination stations need to explicitly provide the physical "digipeater string" information to the NET.EXE package, as ARP is totally unaware of the existence of digipeaters. This is not good. Several extensions to ARP **have** been proposed, but **a** better solution than extending ARP may be to manual routing of digipeaters for IP, and rely on IP Switches and/or other "level three" network. implementations to make the subnet appear to be contiguous.

An IP Switch in a certain sense performs the same function as a digipeater from the end-user viewpoint. The Switch receives packets that are destined for other hosts, and forwards them via one of perhaps several channels. For those familiar with Ethernet or other wired networks) running TCP/IP, an IP Switch in the Packet Radio environment differs from an IP Gateway primarily in that a Switch **may** want to retransmit a packet on the same channel it heard it on, while a Gateway would never want to do that. The fundamental difference between a digipeater and a Switch is that a Switch makes routing decisions, while a digipeater depends on the end user to source-route **a** connection through it.

Address resolution therefore is the mapping of a "network address", such as an IP address in the TCP/IP environment, into an actual physical station identifier, such as a TNC callsign. Routing is the set of issues and algorithms that surround passing packets through intermediate stations,,

Thoughts About What We Need:

Routing individual datagrams is currently handled by manually inserting entries into a station's "route table" that tell the software to route all packet for a given subrange of IP addresses via a specified gateway, or switch. This is already a very powerful mechanism, in that a user can specify his default routing to be via his local switch, and then add special cases to "un-default" local stations he can hit directly, and the end user problem is therefore greatly reduced.

But the IP Switches currently run the same software. The only way to build a network is to manually create routing entries at each switch. We need to develop an algorithm for automatically determining reasonable paths, that can automatically update the routing tables at each Switch site. There are two different philosophies about how to do this. One is based on the idea that the network should "already know" what to do with each packet when it arrives at **a** given switch location, the other is based on the idea that someone requesting a connection to a site that is not known locally generates by their request the initiative for the network to go figure out how to get there.

The tradeoff at the highest level is simple. The first mechanism forces the network to continuously be passing information about routes between switches, all of which is network overhead. The second mechanism generates a great deal of traffic on the first request for **a** path segment that does not currently exist in **the** routing tables, but probably **has a** lower overhead overall. It does, however, cause a much greater "turnaround time" delay for the user requesting an unusual route. What the best solution to this tradeoff between overhead and individual user performance is has not in my opinion been sufficiently considered.

A final thought for this paper is that we need to be able to handle mobile stations, not just stations that are away from their home switches, but stations that are actually in motion. This probably cannot be done by schemes that dynamically reassign IP addresses for the simple reason that a station might have a connection in progress when it crosses a switch boundary. Food for thought.

Related Issues:

There are two other issues that need to be considered. One is the mapping of mnemonic hostnames into IP addresses, both for clients like FTP and Telnet, and for Mail transfer. Secondly, there is **a** separate but similar routing problem that exists in the mail handling portion of the network software, in that a given node may have more than one mail transport system available to it (including perhaps things like uucp and Fidomail), and must make a "routing decision" as to which delivery agent should be called to process **a given message.**

The current means of handling the mapping of mnemonic hostnames into IP addresses is to maintain **a** file listing the possible translations that are known to the system, and requiring the user to specify the IP address directly when a host entry cannot be matched. A much better system would be to designate a host-name server in each local subnet, and have the client software negotiate with the name server to map

a **hostname** into an address. This reduces by far the number of locations that must maintain a full **host**-name database. **Cacheing** of some number of most recently or frequently used addresses on the local machine should minimize the impact of name-serving on the network's overall performance.

Mail routing is a fascinating topic, but one which deserves much more room than that which is available here. Suffice to say that the problem of mapping a given mail message into an appropriate delivery agent on a given mail-handling host can be as simple or as complicated as routing packets at a packet switch.

Wrong Things We Could Do:

With amazing regularity, well-meaning hams have suggested a variety of schemes for "routing" packets based on everything from grid squares to zip codes, to telephone exchange prefixes. The fundamental problem with this is that there is no relation between ***a host's name or address and what is the best path to take*** to ***reach that*** host. A classic example is the satellite "wormhole" between CA and MD, which often means that to get from the **midwest** to the east coast, the best path would be to go west, not ***east,*** to the **wormhole** end, and tunnel across country.

We need to design automatic routing mechanisms that do not rely on host names or network addresses. We can use names and addresses as hints, but cannot expect them to always work, particularly given the mobile nature of the modern ham! Routing depends on a network's topology, not the geography or political boundaries that it crosses. Any routing mechanism that we adopt must handle mobile stations as well, which is virtually impossible when the routing mechanism is based on some fixed geographical reference system.

Another interesting proposal has been to not assign fixed IP addresses to individual stations. There are some schemes for dynamically assigning addresses to small hosts in use in major universities, but I am not yet convinced that this is a good thing to do on packet, primarily because our environment poses some additional complications over those present in the wired university environment. The most obvious of which is the mobile station, which might have a connection in progress as it crosses an address allocation zone. None of the existing schemes that I am aware of allow for mobile stations. We should, however, keep an eye on progress in this direction outside of amateur radio, in case someone makes it work.

Conclusion:

The purpose of this paper has been to document some goals for those implementing routing mechanisms for packet radio, and to raise some specific concerns that should be understood and considered. Much work remains to be done. Many of the ideas discussed here are under consideration by the group of people working with the **KA9Q** Internet software, and undoubtedly we will eventually see an automatic routing implementation either as part of the package, or adopted from a third-party source's level three implementation.

References:

1. Tanenbaum, A.S., "Computer Networks", Prentice-Hall, 1981.
2. RFC826, Address Resolution Protocol.
3. **Beattie, J.G.,** "Proposal: Recommendation **AX.121NA** Numbering Plan For The Amateur Radio Network in North America", ***Fourth ARRL Amateur Radio Computer*** **Networking Conference,** 1984.
4. **Beattie,** J.G., "Proposal: Recommendation AX.121 International Numbering Plan For The Amateur Radio Network", ***Fifth ARRL Amateur Radio Computer Networking Conference,*** 1986.
5. Fox, Terry, "Amateur Network Addressing and Routing", ***Fifth ARRL Amateur Radio Computer Networking Conference,*** 1986.
6. NET/ROM Beta-Test Preliminary Documentation, Software 2000 Inc.
7. **TEXNet** Presentation at the 1987 TAPR Annual Meeting.