Rev	Description	Date	Author
Draft	iRouter Software Architecture	5/7/96	Mike Johnson

TITLE: iRouter Software Architecture

ORIG: Mike Johnson Date 5/7/96 DWR. NO. REV.

1. Introduction

1.1 Document Description

This document describes the software architecture of the iRouter. It shows how the various parts of the iRouter code interact with one another. The packet structure used for configuration will be shown. The general data structure will be defined as well.

1.2 Definitions, Acronyms, and Abbreviations

ARP	Address Resolution Protocol.
IP	Internet Protocol.
ICMP	Internet Control Message Protocol.
UDP	User Datagram Protocol.
TCP	Transmission Control Protocol
SNMP	Simple Network Management Protocol
DNS	Domain Name System
PPP	Point to Point Protocol
PPP Link	A physical network link running PPP
FTP	File Transport Protocol
iVMAP	iReady Virtual Address Mapping Proxy database.
Real Network	The real IP address space on the PPP side of the iRouter.
Virtual Network	The virtual IP address space on the Ethernet side of the iRouter.
Incoming Traffic	Traffic traveling from the Real Network to the Virtual Network.
Outgoing Traffic	Traffic Traveling from the Virtual Network to the Real Network.
Virtual Proxy Traffic	Incoming and Outgoing Traffic that is passed without a connection entry in the iVMAP database.

1.3 Applicable Documents

NS-1000 Product Specifications Document.
PPP-1000 Product Specification Document.

2. Software Architecture Overview

The current iRouter software will run on an x86 platform.

There will be several software modules to make up the iRouter:

- Virtual IP/Port mapper will maintain the mapper database and re-map the incoming and outgoing packets.
- DNS Server proxy handles all the Name Service requests for the internal iRouter network.
- Configuration module handles the configuration of the iRouter and will handle status and statistic reporting.

The software will be designed to run in an embedded environment. It will be developed in ASM, C and/or C++.

2.1 Software Architecture Overview

Figure 1 diagrams the iRouter software modules as they relate to the iRouter hardware. The IP/Port mapper is the core software module, using the configuration, dialer, and DNS Proxy modules for support.

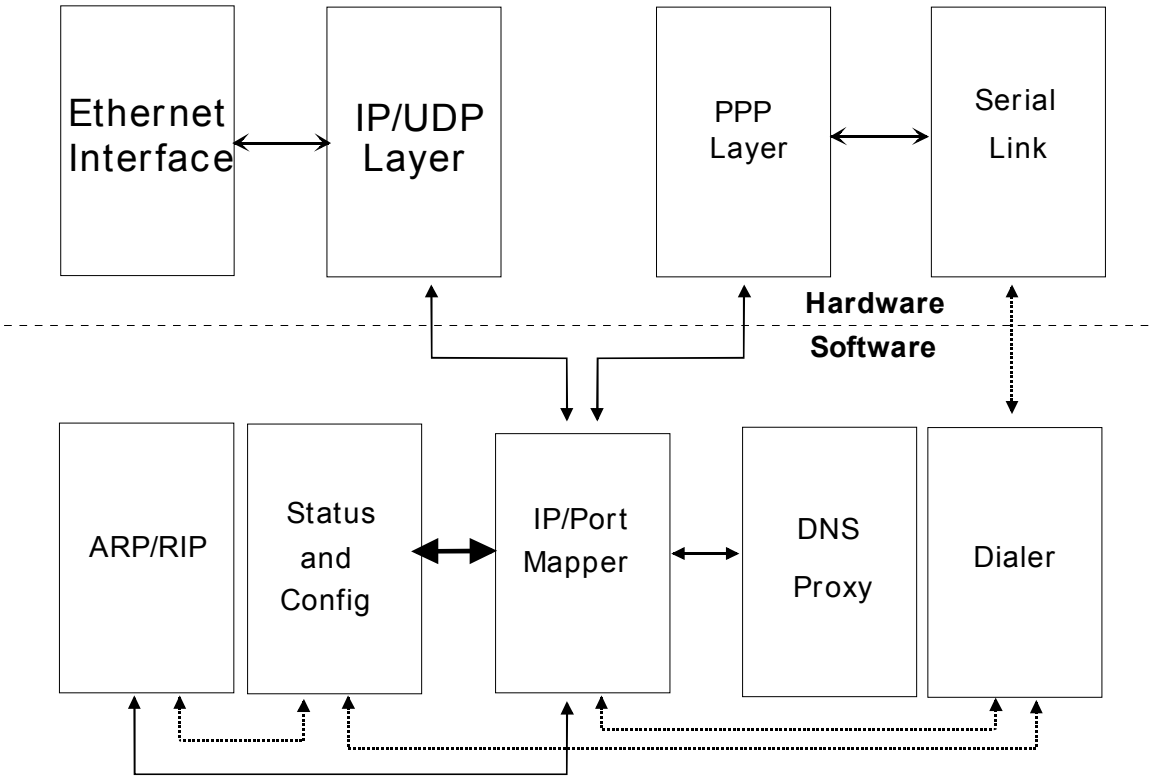

Figure 1: iRouter Software/Hardware Diagram

Status and config module has the following functionality:

- Handles requests for the iRouter's status, statistics and configurations parameters.
- Allows the configuration of the iRouter's configuration parameters (including static entries in the iVMAP database).
- Allows the initiation of a dialup session.
- In the future will offer SNMP support.

IP/Port Mapper module performs the following:

- Initiate and maintain a iVMAP database of connections from the virtual network to the real network.
- Map and De-Map packet headers using the iVMAP database.
- Direct packets addressed to the iRouter to a handler module (configuration or DNS).
- Support other transparent proxy protocols in the future (ICMP Echo, Talk…)

The Dialer module will provide the following:

- Send a dialing string to the modem and read the modems status (AT command set).
- Tell PPP to start a connection.
- Signal the state of the modem connection.

The DNS Proxy handles the following functions:

- Answer DNS requests.
- Ask a nameserver for DNS information.
- Cache DNS information.

The ARP/RIP module will provide the following:

- MAC to IP translation on the Ethernet port (ARP Requests).
- Caching of MAC to IP records.
- Provide ARP Replies.
- In the future advertise and keep track of RIP router metrics.

3.0 Software Description

All the code modules in the previous chapter will interface with each other through global routines, global variables and pointers. A single profile/configuration/parameters and statistics structure is maintained under the file name of "confstat.h".

To facilitate the debugging process, each code will declare its own internal data usage and any public variables will be declared in a single module (pubvars.asm). Since there could be a mix of assembly and C code, the C code should be compiled with structures packed in byte boundary. (However, an attempt should be made to keep 32 or 16 bit entities aligned on a word boundaries.)

3.1 Public Data Structure Types

3 types of shared (public) data structures have been defined. Each data structure has structures that are used/updated by individual code. The following shows the 3 types of shared data structures:

- Statistics. It has several data structures belonging to Status and Config, dialer, and IP/Port mapper.
- Profile. This consists of a read copy of configuration parameters and read-only paramters. It is a group consisting of different profile structures belonging to different groups.
- Configuration. These are the parameters configurable by the user-interface code.

3.1.1 Statistics Data Structure

As figure 2 indicates the statistics group consists of the number of statistics members (32 bit variable), followed by an array of 32 bit far pointers to the statistics data structure maintained by each member code, which is preceded by a 32 bit variable that has the length in byes of the statistics maintained by each member (not including the length field). The length is necessary to facilitate the resetting of all the iRouter statistics at one point. The following shows the statistics members:

1. IP/Port Mapper.
2. Dialer.
3. DNS Proxy

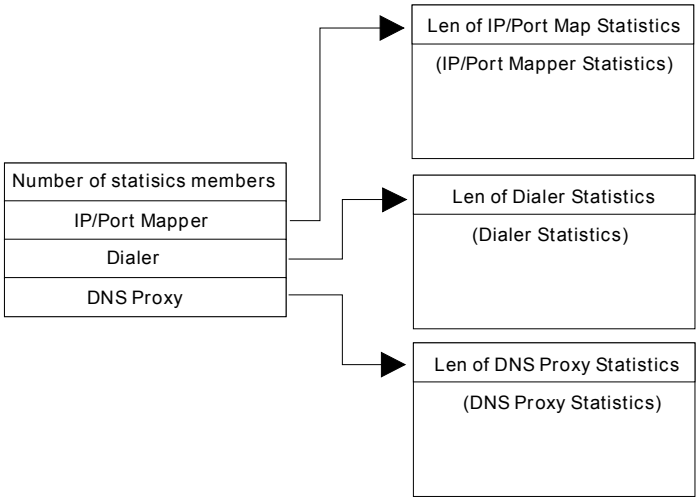

Figure 2: Statistics Data Structure.

3.1.1 Profile Data Structure

As figure 3indicates the profile group consists of the number of profile members (32 bit variable), followed by an array of 32 bit far pointers to the profile data structure used by each member code, which is preceded by a 32 bit variable that has the length in byes of the profile items used by each member (not including the length field). The following shows the profile members:

1. IP/Port Mapper structure pointer. (Incoming pass through proxy data, Local IP, Local Net Class, Base Port, Number Ports…)
2. Dialer. (Primary and Secondary numbers to dial, re-dial time out, maximum idle time, dedicated connection flag…)
3. DNS Proxy structure pointer. (Real DNS name servers, Local NS database.)

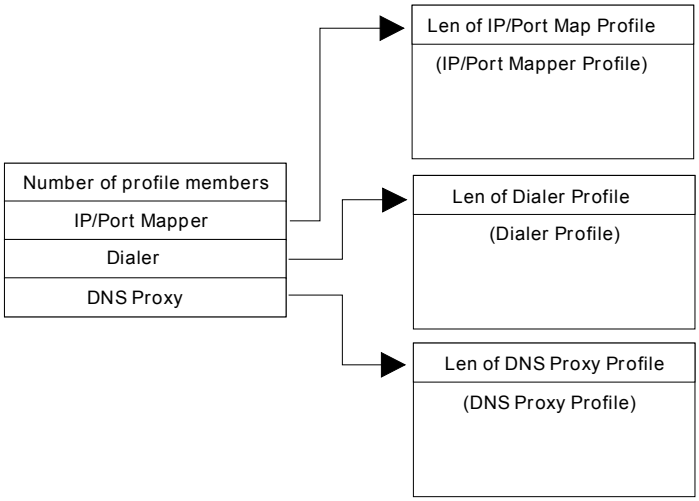

Figure 3: Profile Data Structure.

3.1.1 Configuration Data Structure

As can be seen from figure 4, the configuration structure is much the same as the profile structure. Since both the Configuration code, and in the future SNMP code, will be able to modify the configuration structure, a public byte semaphore (g_config_sem) has been defined. The code will

check and set the semaphore prior to changing the configuration parameters and clear it after calling the configuration handler routing.

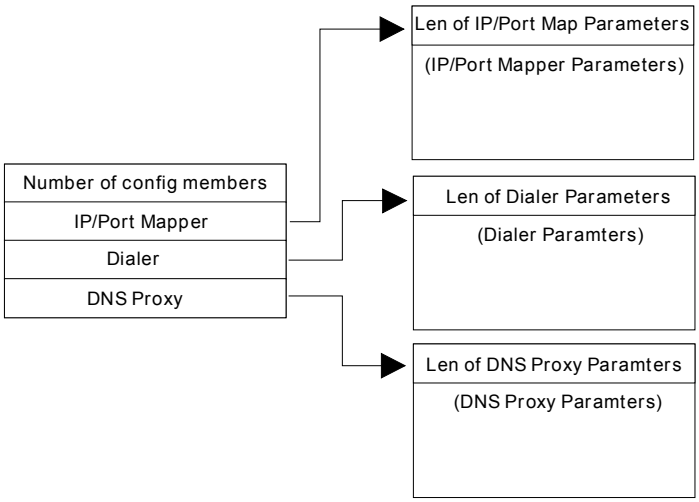

Figure 4: Configuration Data Structure.

3.2 Modules Interface

This part describes the interface between Software and Software, and Software and Hardware modules.

3.2.1 IP/Port Mapper

This code is actually 2 closely interwoven modules, a Traffic Director and the IP/Port Mapper Core.

The Traffic Director will provide the actual interface between the iRouter's software and hardware. It will directly interface to the IP/UDP and PPP hardware modules. It will direct data to and from these modules to the various software modules. It will use the iVMAP database to determine where the traffic should be sent. The traffic could be sent to the IP/Port Mapper core, or to one of the various modules like the 'Status and Config', 'ARP/RIP', 'DNS Proxy' or other current or future developed support modules.

The IP/Port Mapper Core will provide the IP/UPD/TCP header manipulation for current connections, and setup the data necessary to create a connection for unconnected data paths. Using the iVMAP database it should be possible for plug in module support for specific transparent proxy functions like FTP, ICMP Echo, Talk and other 'Well Known' applications.

The IP/Port Mapper will have two separate entry points for looking up data in the iVMAP database, an incoming (from the 'real network') and outgoing (from the 'virtual network').

An incoming packet will be looked up on the destination port only to determine what to do with the packet since all connections to the 'virtual network' have been previously setup and assigned a port number or fall into a pass through proxy arrangement by the iVMAP database.

An outgoing packet will be looked up on the source IP, Destination IP, Source Socket, and Destination Socket. The iRouter needs to determine the uniqueness of the connection and needs all the connection information listed above. Once the information has been looked up the iRouter can determine how it should handle the packet, either by creating a new connection entry, or using an existing connection entry to modify the IP/Upper Level Protocol (ICMP/TCP/UDP…).

3.2.2 Status and Config

The status and config modules function is to allow configuration and to report the status and profile of the iRouter. When you are talking directly at the iRouters IP address you are talking to the Status and Config module.

The Status and Config module will on a first draft talk an iReady proprietary configuration protocol running on the top of UDP, but could in the future support SNMP.

See the related document for the Status and Configuration packet request and reply format.

3.2.3 Dialer

The dialer module will handle any communication with a modem connected to the serial link. It will be responsible to dialup an Internet provider, check for busy signals, re-dial, and signal the PPP layer, the IP/Port Mapper and the Status and Config module its status.

3.2.4 DNS Proxy

The DNS Proxy module is an example of a 'plug in' support module that can be added to provide support or increased performance for a specific application. The DNS Proxy will act to the internal network as a DNS server, but in effect it will use other DNS servers (know only to itself) to gather data. It will also cache DNS data, limiting the load to the link to the 'real network'.

3.2.5 ARP/RIP

The ARP/RIP module will handle the physical transport to IP Layer resolution for the Ethernet connected network, in the future support for RIP may be added to support smart routing of internal 'virtual network' network packets to the 'real network' when more than one iRouter is connected to the 'real network'.

The ARP module will handle requests for the iRouter's physical Ethernet address and also request Ethernet addresses of IP addresses on the Virtual Network. It will hold a cache of looked up IP address and Ethernet addresses for further use.

4. Software State Description

This chapter describes the various states that the iRouter could be in and its action in each state.

4.1 Startup

During the software start-up the 'Status and Config' code will check for the existence of previously stored parameters, if they do not exist default values will be loaded.

If the iRouter is using default values, the IP/Port Mapper will only allow packets to move between the Virtual Network and the 'Status and Config' module.

Once previously stored parameters are loaded, or operational configuration parameters are obtained from a configuration program the iRouter goes into an idle state.

4.2 Idle State

Once in an idle state the iRouter is ready to create a connection, if the iRouter is setup in dedicated mode it will start the process of creating a PPP session. If the iRouter is in 'on demand' mode it will sit idle until an entity on the Virtual Network wishes to send traffic to the pass packet to the real world or is told to establish a PPP link by the management software. Once a PPP session has been established the iRouter goes into an active state.

4.3 Active State

The iRouter is in an active state when it has established a PPP link with the Real Network. When in this state it is ready to accept incoming and outgoing connections, and do DNS queries.

There are several sub states that the iRouter can be in when in the active state, incoming traffic processing, outgoing traffic processing, outgoing connection setup, local traffic processing, and connection maintenance.

4.3.1 Incoming Traffic Processing

The first test for incoming traffic is a protocol check, if the protocol is not UDP or TCP a quick check in the iVMAP database will determine if there is a proxy module loaded to support that protocol and redirect the packet to that module.

When an incoming TCP/UDP packet is processed the destination port number is applied to the iVMAP database, the first check is for ranges of ports to pass as Virtual Proxy Traffic. If there is a match the destination IP address is changed to the value found in the iVMAP database, the header checksum is updated, and the packet is sent out the Ethernet Port.

If the destination port doesn't register for Virtual Proxy Traffic, it is looked up in the connection table to see if there is a current connection associated with this port. If there is a match in the connection table the packet headers are fixed up with the iVMAP connection information and send out the Ethernet Port.

If the packet fails a lookup in the above two cases, the packet is discarded.

4.3.2 Outgoing Traffic Processing

When an outgoing packet is processed the first check is to see if it is an IP packet or ARP packet. ARP packets are handle in the local traffic mode. If the packet is an IP packet the protocol type is checked next, if the protocol is not UDP or TCP, the protocol type is looked up in the iVMAP database and processed by any proxy routing found. Otherwise the packet is discarded.

Then we check the iRouters IP address with the destination IP address if there is a match the iRouter enters the Local Traffic Processing Mode.

If the traffic's destination is not outside our network address space it is discarded.

The next test is to check the source IP and source port (in both UDP and TCP) against the iVMAP database to see if the packet falls under the Virtual Proxy Traffic definition, if so the source IP address is changed, header checksum recalculated and the packet sent over the PPP link.

The next check is to do a connection lookup using the source IP, destination IP, source port, and destination port. If a match is found the source IP and source port from the database is inserted in the packet, the checksums are recalculated and the packet is sent across the PPP link.

If the lookup turned up no connection one must be created, the iRouter would enter Outgoing Connection Setup mode.

4.3.3 Outgoing Connection Setup

When an outgoing connection needs to be setup a connection entry is created and entered into a hash table using the incoming traffics source and destination IP address and the source and destination port for Outgoing Traffic and linked to another hash table indexed on the newly assigned source port for Incoming Traffic. The source port and source IP addresses are replaced in the packet, the checksums are recalculated and the packet is sent across the PPP link.

4.3.4 Local Traffic Processing

Local Traffic Procession consists of looking up in the iVMAP database the protocol and destination port and seeing if there is a module associated with it (like the Status and Config module or the DNS Proxy module..) If there is a module associated with the packet the packet is passed to that module, else the packet is discarded.

4.3.5 Connection Maintenance

Periodically the iRouter enters the Connection Maintenance mode. Its basic function is to go through the connection table and look for timed out connections and terminate them.

5 iVMAP Database Structure

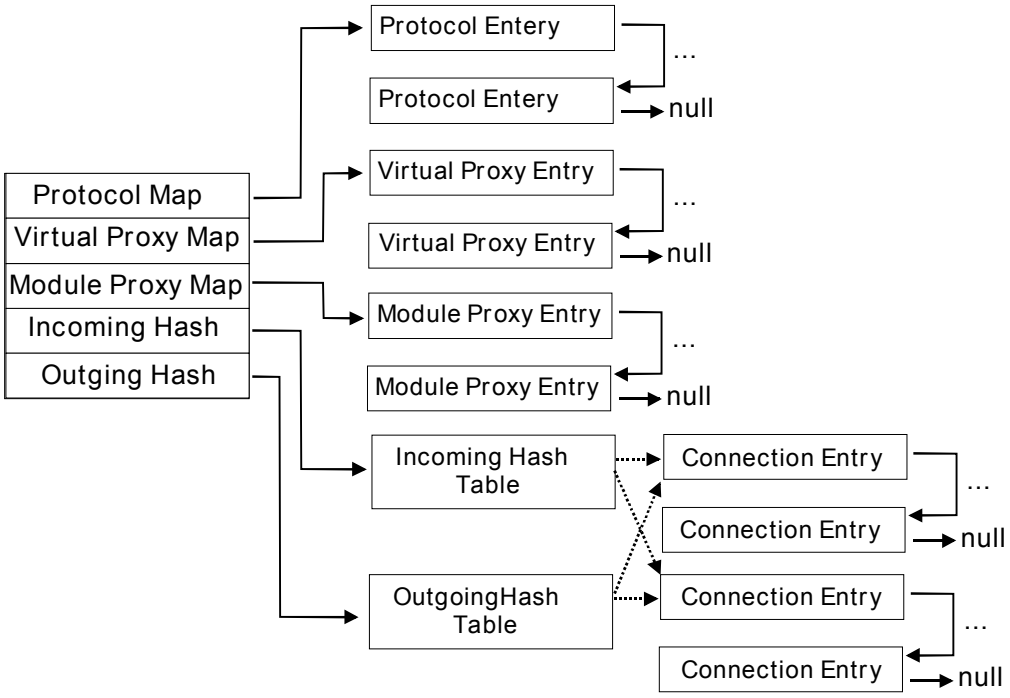

Figure 5: iVMAP Database Structure Overview

5.1 Protocol Entry

The protocol entry field will be searched first depending on the protocol field from the IP packet. This is only applicable to the Ethernet side of the iRouter. Protocols supported in the first Rev. of the iRouter should be TCP and UDP (and possibly limited ICMP.)

The following is the structure for a protocol entry:

0	1	2	3	4	5	6	7	8	9
pointer to next protocol entry				ID		pointer to protocol handler			

The top line shows the byte offset and the bottom line shows the structure items. The structures of protocol entries will make up a link list that is null terminated. The ID word (low byte only) is matched with the protocol field in the IP Header of the packet to process, and if there is a match the protocol handler specified will process the packet.

5.2 Virtual Proxy Entry

The Virtual Proxy Entry table is a list of ports/protocols (applicable to any port based transport layer like UDP or TCP) that are to be passed through without renumbering the ports, only the IP address that is on the Virtual Network side will be re-mapped, and with a fixed IP address defined in the Virtual Proxy Entry. If a packet doesn't match one of these entries the next check will be for a Module Proxy Entry.

0	1	2	3	4	5	6	7	8	9

pointer to next virtual proxy entry.	Protocol 0=UDP, 1=TCP	Port Range Start	Port Range Len

The structures of protocol entries will make up a link list that is null terminated. If the Port Range Len is 0 only one port will be covered, 1 2 ports will be converted…

5.3 Module Proxy Entry

The Module Proxy Entry table is a table of ports to run a specialized iRouter proxy module on. All ports refer to ether an Outgoing destination port or an Incoming source port. In most cases the specialized iRouter proxy module associated with the port will use information embedded inside the data portion of the packet. FTP would be a good example of a Module Proxy since the port and IP address of the data connection for FTP is passed to the server as FTP data from the FTP client. A FTP Module Proxy would look for this data and setup an inbound connection mapper for FTP data connection initiated by the FTP server on the Real Network.

0	1	2	3	4	5	6	7	8	9
pointer to next module proxy entry.				Port		pointer to module proxy handler.			

The structures of module proxy entries will make up a link list that is null terminated.

5.4 Incoming Hash Table

The Incoming Hash Table will contain a list of open connection (4 byte pointers to connection entries) pointers hashed by the XOR of the two bytes of the destination port number of an incoming TCP or UDP packet. If an incoming packet doesn't match a valid entry in the hash table it is discarded.

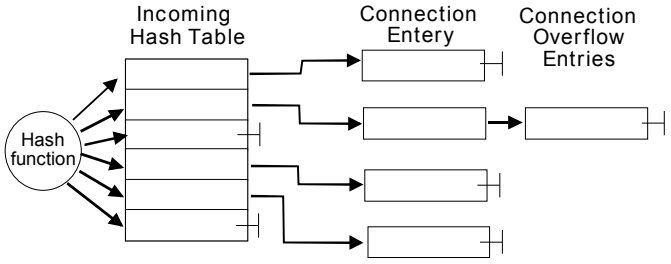

Figure 6: Incoming Hash Table

Initially for lower end products we may not want to use an Incoming Hash Table, we may just want to use an array of pointers that directly maps to a port number. Lower end products shouldn't need more than 64 or 128 connections which could be stored in a direct index table more efficiently.

See section 5.6 for the format of connection entries.

5.5 Outgoing Hash Table

The Outgoing Hash Table will contain a list of open connection (4 byte pointers to connection entries) pointers hashed into by a function that creates a hash value by XOR-ing all the bytes of

the Destination Port, Source Port, Destination IP address, and Source IP address. The Outgoing Hash Table will look exactly like Figure 6 except that the hash function will be different.

See the next section (section 5.6) for information on connection entries.

5.6 Connection Entry

The connection entry contains the mapping information for each connection from the Real Network to the Virtual Network. This information is used to modify the Incoming and Outgoing Traffic's IP/UDP/TCP headers as it passes through the iRouter. The connection entry is as follows:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Virtual IP Address				Real IP Address				Virtual Port		Real Port		Protocol (udp/tcp)		Time To Live	
Base Time		Flags		Proxy Filter Routine											

6 iRouter Management Packet Format

We could do an SNMP based management agent, develop a proprietary management format, or both.

A SNMP MIB or the proprietary management format will be determined on the next draft of this document..

APPENDIX A: FTP Proxy Module

Since FTP needs the special case of the server opening a data connection to the FTP client, to a port on the client specified in the initial control connection, we need to be able to allow a connection to a client on the inside of our network from the FTP server. Since the clients address does not exist the server will not be able to open the needed data connection under the iRouter's normal mapping scheme. The iRouter will have to be able to modify the FTP packet coming from the client and give its IP and Port information so that the FTP server can open a connection to the correct IP address, and then the iRouter can fix the IP and TCP header so that the FTP client doesn't know the difference. This is handled by a Module Proxy as follows.

During a the connection process we can look into the iVMAP database at the Module Proxy Entries and determine by checking the Real Port number (destination port number) if we are connecting to a FTP server, if so we can insert a FTP proxy filter into the connection table for the created connection. This will allow us to run some code for every packet that belongs to this connection. This will allow us to peer into the FTP packet and search for a 'PORT' command coming from the client on the inside of our network and replace it with information that will correspond to a connection entry that we will create. This will allow an incoming connection from the FTP server to use this connection entry to open it's data connection.

APPENDIX B: ICMP 'Ping' Proxy Protocol Module

ICMP is portless protocol, meaning that we only have the source and destination IP addresses (no source and destination port) to determine where the packet should travel. Instead it uses a field in the header to identify the process ID of the sending process. This means we will have to use another method to keep the correct 'Ping' (ICMP Echo) packets to move between the Virtual and Real networks. We will use the Source and Destination IP addresses in conjunction with the 'Ping'

identifier field to create a 'Ping' connection. We will also assign it an iRouter 'Ping' Connection identifier. With this information we will re-map the ICMP/'Ping' header to contain the iRouters source address and the iRouter 'Ping' connection identifier with the real identifier on Outgoing Traffic. And on Incoming Traffic we will use the iRouter 'Ping' Connection identifier passed back to us from the Real network to determine which machine on the Virtual network we should send the 'Ping' reply to (ICMP Echo Reply).

The iRouter will use the Protocol Entry of the iVMAP database for ICMP 'Ping' Proxying.

APPENDIX C: Virtual Network IP addresses:

The iRouter will be configurable to use any IP addresses (and class of IP addresses) for its Virtual Network, but when connected to the Internet it is recommended that the addresses stay RFC #1597 compliant to eliminate any routing problems resulting from using a 'Real' IP address that exists elsewhere.

According to RFC #1597 there are 3 blocks of numbers set aside for non-connected networks. These number blocks will never appear on the Internet, and are ideal for use on the iRouters Virtual Network. They are as follows:

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.16.255.255
- 192.168.0.0 to 192.168.0.0

By default the iRouter will be setup to use the class-C network 192.168.0.x where the iRouter's Ethernet port will be assigned 192.168.0.1.