



# **RTXC Quadnet TCP/IP Networking Software Technical Summary**

**Quadros Systems, Inc.**

*Real-time Operating Systems for Convergent Processing*

[www.quadros.com](http://www.quadros.com)

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>4</b>
<b>2</b>	<b>Memory Requirements .....</b>	<b>4</b>
<b>3</b>	<b>Integration with the RTXC Quadros RTOS .....</b>	<b>4</b>
3.1	Semaphores .....	4
3.2	Mutexes .....	4
3.3	Tasks .....	5
<b>4</b>	<b>TCP/IP Networking .....</b>	<b>5</b>
4.1	Protocols .....	5
4.1.1	UDP .....	5
4.1.2	RAW .....	5
4.1.3	TCP .....	5
4.1.4	ICMP .....	5
4.1.5	IGMP .....	5
4.1.6	Address Resolution Protocols .....	5
4.2	Physical Connections .....	5
4.2.1	SLIP / CSLIP .....	5
4.2.2	PPP .....	6
4.3	Packet Types .....	6
4.3.1	DIX Ethernet .....	6
4.3.2	802.2 .....	6
4.3.3	802.3 .....	6
4.4	Sockets .....	6
4.5	IP and Port Addresses .....	6
4.6	Broadcast and Multicast Addresses .....	6
4.7	Packet and Segment Sizes .....	6
4.7.1	Maximum Transfer Unit .....	6
4.7.2	Maximum Segment Size .....	6
4.8	Packet Flow .....	6
4.8.1	Input .....	7
4.8.2	Output .....	7
4.8.3	Send via UDP .....	7
4.8.4	Opening a TCP Connection .....	7
4.8.5	Closing a TCP Connection .....	8
<b>5</b>	<b>Network Applications .....</b>	<b>9</b>
5.1	FTP .....	9
5.2	TFTP .....	9
5.3	TELNET .....	9
5.4	SNMP .....	9
5.5	NFS .....	9
5.6	WEB .....	9
5.7	DHCP .....	9
5.8	POP3 .....	9
5.9	SMTP .....	10
5.10	SYSLOG .....	10
5.11	RIP .....	10
5.12	NAT .....	10
<b>6</b>	<b>Configuration .....</b>	<b>10</b>

<b>7</b>	<b>Device Driver Interface</b> .....	<b>10</b>
<b>8</b>	<b>Network Management</b> .....	<b>10</b>
8.1	Routing Information Protocol .....	10
8.2	Domain Name System .....	11
<b>9</b>	<b>Error Logging</b> .....	<b>11</b>
<b>10</b>	<b>More Information</b> .....	<b>11</b>

## 1 Introduction

RTXC Quadnet is a CPU-independent implementation of the TCP/UDP-IP network stack for use in embedded systems. It provides deterministic, configurable memory usage, a table driven device driver interface and a simple API. No disk services or external function library services are required.

RTXC Quadnet can configure itself at run-time using standard RARP, BOOTP and DHCP protocols and can support simultaneous communications over multiple interfaces. It has a simple programmer's API and supports the following protocols: UDP, TCP, ARP, RARP, BOOTP, IGMP v2 and ICMP.

The most important data structure of RTXC Quadnet is the DCU (Data Control Unit). This is a pre-allocated structure that contains link management fields and a pointer to a data area. The primary purpose of the Quadnet software is to route DCU's from the device driver's receive routine to the application layer that handles network receives and from the application layer to the device driver's send routine on sends. DCU's are routed between layers via internal queues.

## 2 Memory Requirements

The following memory resources are required. The size and number of each of these structures are configurable.

- **DCUs:** a pool of buffers used to route data between protocol layers
- **PACKETS:** a pool of packet buffers. As many as five buffers of different sizes can be allocated to optimized memory efficiency
- **IFACEs:** contain address information and optional counters
- **UDPPORTs:** contain connection management information for UDP sockets
- **TCPPORTs:** contain connection management information for TCP sockets
- **ARPCACHEs:** contain Ethernet information for IP addresses and for all output packets waiting to be sent
- **ROUTEs:** contain routing information
- **IPFRAGLISTs:** contain fragments of input packets

## 3 Integration with the RTXC Quadros RTOS

RTXC Quadnet is tightly integrated with the RTXC Quadros real-time operating system. RTXC Quadnet includes an operating system mapping layer designed to minimize the number of operating system services required.

All operating system service requests are invoked by macros. RTXC Quadnet requires access to the following kernel objects and services:

### 3.1 Semaphores

Four semaphores are required per interface. They are used for the following purposes:

- For the device driver to signal when a packet has been received or a network packet has been sent;
- For the device driver to signal when an interrupt needs processing;
- To signal when a ping echo reply has been received;
- To signal when an RARP (Reverse Address Resolution Protocol) reply has been received.

Four semaphores are required per socket. They are used for the following purposes:

- For the protocol layers (UDP/TCP) to signal the application layer when data is available;
- For the protocol layers (UDP/TCP) to signal the application layer when a send request has been completed;
- An internal semaphore is used to allow the ARP (address resolution protocol) layer to transparently block on a send if address resolution is required;
- An internal semaphore is used for implementing the *select* system call.

### 3.2 Mutexes

One Mutex is required per interface to claim exclusive access to the interface's device driver while sends, opens and statistics calls are made.

Four critical section semaphores are required. They are used for the following purposes:

- For internal use by the Quadnet list management code.
- To claim exclusive access to the UDP port list while the list is manipulated.

- To claim exclusive access to the TCP port list while the list is manipulated.
- To claim exclusive access while manipulating one of the following internal tables: ARP, routing, fragmentation, DNS host and DNS host cache.

### 3.3 Tasks

At initialization the following tasks are spawned:

- One IP task per interface. Each one of these tasks process input packets received for its interface and manages the output queue (i.e. packets queued to be transmitted).
- One timer task to perform all processing which needs to be done on a periodic basis (i.e. TCP retransmits, ARP cache timeout/retries, fragmentation table timeout, routing table timeout, etc).
- One interrupt task per interface. Each of these tasks may process interrupts at the task layer depending on the needs of each driver.

## 4 TCP/IP Networking

### 4.1 Protocols

Five types of IP packets may be sent: UDP, RAW, TCP, ICMP and IGMP.

#### 4.1.1 UDP

UDP is an unreliable transport system for transferring data between machines. When a packet is sent “successfully” it means only that the packet was sent out on the wire. Reliability needs to be done at the application layer

#### 4.1.2 RAW

RAW is similar to UDP with the following exceptions

- UDP has a fixed value. RAW sockets can bind to any protocol number
- RAW does not have a protocol header
- RAW data which is read includes an IP header; UDP read data does not include IP or UDP headers
- RAW input packets are delivered to all sockets of the same protocol type and IP address whereas UDP packets are delivered to only one connected or listener socket with matching IP, port address

#### 4.1.3 TCP

TCP is a reliable transport mechanism. When data is sent it is guaranteed the remote host will get the data. Also, TCP is a stream protocol so it does not put boundaries between sends. For example, if send is called twice to send 1000 and 2000 bytes, it may send two packets of 1500 bytes or 6 packets with 200 bytes each. This is transparent to the application.

#### 4.1.4 ICMP

ICMP is a protocol used to send control and error information between hosts. Error messages include destination unreachable, parameter problem, etc. Control messages include source quench and PING. PING messages are invaluable to determine if a host is alive.

#### 4.1.5 IGMP

IGMP is a protocol which provides multicast routers information about multicast addresses on a network. This allows the multicast router to forward multicast packets to remote networks. RTXC Quadnet is not a multicast router but does support host side IGMP. The host side IGMP sends reports when it joins a multicast group and in response to queries from a multicast router.

#### 4.1.6 Address Resolution Protocols

- **ARP** is used to retrieve Ethernet addresses from a given IP address
- **RARP** is used to retrieve system configuration information such as IP addresses, gateway information, etc.

## 4.2 Physical Connections

RTXC Quadnet supports computers connected together via Ethernet or RS232 connections. SLIP, CSLIP and PPP run over the RS232 interface only. SLIP, CSLIP and PPP perform the following:

#### 4.2.1 SLIP / CSLIP

SLIP encapsulates IP packets into SLIP packets and transmits and receives them across a serial RS232 connection. **CSLIP** (SLIP with VanJacobson Compression) is also supported. No negotiation of configuration data is done.

#### 4.2.2 PPP

PPP encapsulates IP packets into PPP packets and transmits and receives them across a serial RS232 connection. PPP also performs negotiation of configuration parameters such as compression, authentication method used, IP addresses, MTU values, etc.

### 4.3 Packet Types

#### 4.3.1 DIX Ethernet

This is the default packet format where the Ethernet header is followed by an IP header which is followed by the protocol header and data.

#### 4.3.2 802.2

Support for IEEE 802.2 standard Class I service. This includes SNAP and LLC headers. In the packet, the 802.2 header lies after the Physical Layer header (Ethernet header for example) and before the IP header.

#### 4.3.3 802.3

Supports the IEEE standard for Ethernet.

### 4.4 Sockets

Each connection on a machine (TCP, UDP or RAW) has an associated socket which is used to specify which connection an API call is referencing. It is analogous to the file pointer type when doing I/O.

Sockets can be in either blocking or non-blocking mode. In blocking mode, the *recv/send/connect/accept* socket API calls will block indefinitely until the requested action has been performed. In non-blocking mode, the *recv/send/connect/accept* functions return immediately.

### 4.5 IP and Port Addresses

Each socket has an IP and port address associated with it if it is bound. An IP address consists of a network part as specified by the network mask and a host part which distinguishes the machines on the network. The port address specifies which socket for the appropriate protocol on the machine is being referenced.

### 4.6 Broadcast and Multicast Addresses

Broadcast messages are sent for UDP and ARP protocols only and multicast messages are sent for the UDP protocol only. Multicast provides the ability to send to a group of Ethernet interfaces. Packets sent to a multicast address are sent out on the interface specified by the routing table. Packets are received on an interface if the multicast address has been set up for the interface.

### 4.7 Packet and Segment Sizes

#### 4.7.1 Maximum Transfer Unit

MTU is the maximum packet size which may be sent on the network. The packet size includes the IP header, protocol header and any data in the packet. The MTU value is set in the device table.

#### 4.7.2 Maximum Segment Size

MSS is the maximum amount of data which may be sent in a TCP packet. The MSS values are sent in the initial SYNC message where each side declares the maximum amount of data it can receive in a packet. The MSS value does not include any of the protocol headers.

### 4.8 Packet Flow

Packets are saved in DCUs. Exchanges and lists are linked lists used to save DCUs and other types of internal data structures. There is a signal associated with an exchange used to signal a layer that a packet was put on its exchange and it needs to be processed.

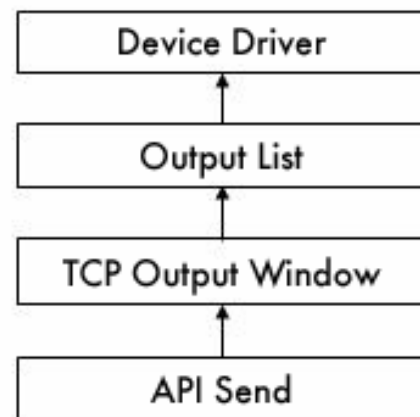


Figure 2. DCU Output Flow Summary

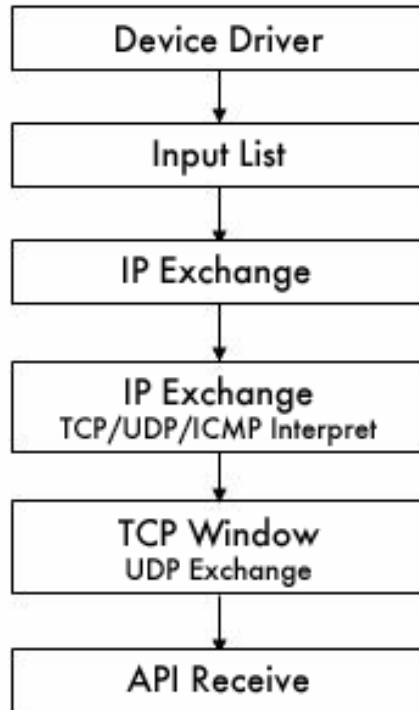


Figure 1. DCU Data Flow Summary

The following is a summary of packet flow through the various exchanges and lists.

#### 4.8.1 Input

When an input packet is received by the device driver it is sent to the input list ( a ranked list of DCUs without a signal associated with it). A function is called to send the packets from the input list to the IP exchange.

When a packet is sent to the IP exchange, the IP task is signaled. The IP task wakes up and processes the packet. The routine performs the following:

- a checksum verification on the IP header
- fragmentation processing
- checks if an ICMP source quench needs to be sent to the originator of the message to request the remote host to slow down sending packets
- calls the appropriate protocol interpret routine if a complete packet has been received (i.e. for a fragmented packet, if all the pieces of the message have been received).

If any data was in the packet it is processed as follows:

- TCP: the data is put in the input window
- UDP: the packet is put on the UDP exchange associated with the socket or port.

#### 4.8.2 Output

Sending of data is done by the API send routines, the interpret routine and the timeout processing routine.

When an API routine is called by the application to send data via TCP, the data is queued in the output window associated with the socket/port. The API routine will block until all the data is queued. The API routine starts sending the data regardless if there is any unacknowledged data in the output window. When the API routine starts sending data, it can only send the amount of data which the remote host can fit in its input window. Therefore, as window size updates are received from the remote host, the input packet processing function will continue the process of sending data. As the data is acknowledged, it is removed from the output window leaving room for the API send routine to queue more data. Every time data is removed from the output window, it signals the API send routine. The send routine wakes up and queues more data. If the data has not been acknowledged within a specified time period, the timeout processing function will resend the data.

#### 4.8.3 Send via UDP

When an API routine is called to send data via UDP, if fragmentation is enabled and the amount of data sent is larger than the MTU value for the interface, then all the data is sent in a fragmented packet. If fragmentation is disabled, as much data as can be sent in one packet, based on the MTU value for the interface (see `xn_pkt_data_max`) is sent

#### 4.8.4 Opening a TCP Connection

In order for a connection to become established, a handshake of SYNC messages and acknowledgments of the SYNC message are sent between two hosts. The connection is considered established (i.e. ready to send and receive data) when both ends have sent a SYNC message and have acknowledged the remote hosts SYNC message. A connection may be established in one of two ways:

- **Passive Connection:** a socket is allocated by calling `socket`, its local address is set up by calling `bind`, then `listen` is called. The opening handshaking is started when the remote host sends a sync message. When a sync message is received for that socket, a new socket (slave socket) is allocated to process the connection. A SYNC message and an acknowledgment of the SYNC received will be sent

to the remote host. The application needs to call *accept* after *listen*. The function *accept* will return the slave socket when the connection is established.

- o Active Connection: a socket is allocated by calling *socket*. Then *connect* is called.

**4.8.5 Closing a TCP Connection**

In order to close a connection, a handshake of FIN and acknowledgments of the FIN is done.

When there is no more data which needs to be sent, the application can do a *closesocket*. Data may still be received after *closesocket* has been called, i.e. it only shuts down data transfer in one direction. When *close-socket* is called a FIN message is sent to the remote host. The socket is not freed until the handshake is complete.

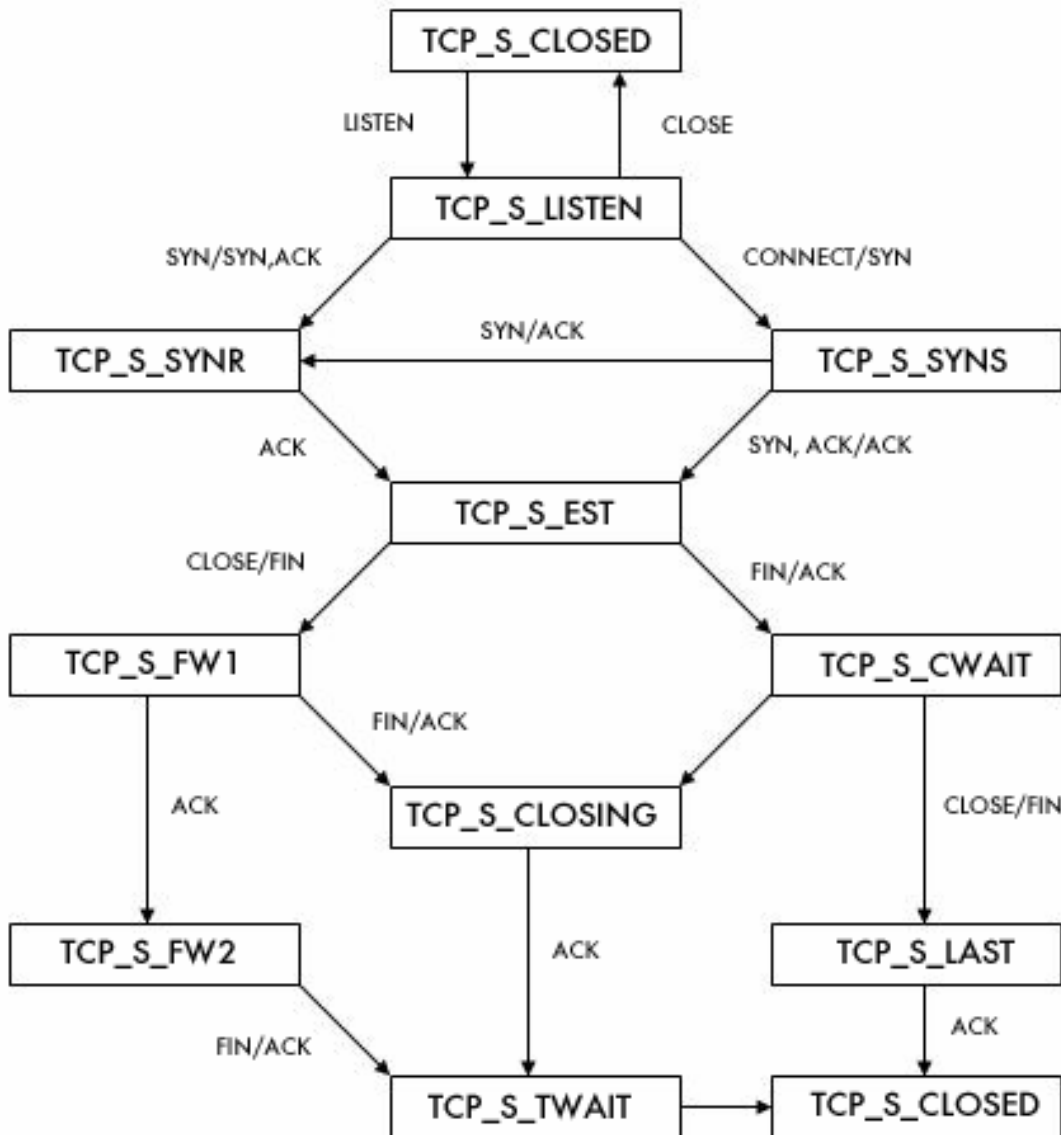


Figure 3. TCP State Transitions

## 5 Network Applications

RTXC Quadnet supports many different network applications. These application layer modules allow your embedded system to communicate with applications on other network attached devices. These application modules are packaged as options, although some are occasionally bundled together in packages.

### 5.1 FTP

File Transfer Protocol is a client/server protocol that allows a client to transfer data files to and from a server. It also provides directory listing and working directory management routines. Quadros Systems offers both client and server modules. Multiple server instances may be created if multitasking is available. The client is provided through a small set of API calls. A sample interactive client shell is provided which demonstrates the API. The underlying file system may be either the RTXCnet file system or a native file system.

### 5.2 TFTP

Trivial File Transport Protocol provides a simple file send and retrieve mechanism over UDP. It is usually used to transfer a boot image from a server to a diskless work station or black box. Quadros Systems offers both client and server modules. The client module retrieves files to a local file. This will usually be modified to load a RAM area.

### 5.3 TELNET

Terminal Emulator provides a standard socket port number and negotiation sequence to provide for remote logins over a network. Usually a client computer with a VT100 emulator program will log in to a host computer at the standard telnet port. The host computer will then bind the standard console I/O driver to the telnet driver. Quadros Systems offers a simple telnet server connection which may be modified for your application. A Telnet client module is also available.

### 5.4 SNMP

Simple Network Management Protocol is a client server protocol that allows a client (manager in SNMP parlance) to get and set variables on the server (Agent in SNMP). The agent may also send traps under some circumstances. These are unsolicited messages that convey urgent messages such as interface down. SNMP pro-

vides the transfer mechanism through an external data representation language called ASN.1. The variables are indexed through an abstract tree structure indexing mechanism. Most of the TCP-IP protocol elements have their own set of variables (MIBS). Quadros Systems offers a simple SNMPV1 agent capability that may be extended by adding elements to a table. SNMPv2 and v3 are also supported.

### 5.5 NFS

Network File System is a client server application developed by Sun Microsystems in which the server allows the client to access the server's local file system as if it were its own. Quadros Systems provides an NFS client that allows an embedded system to mount and access the file system on any host running an NFS server. Quadros Systems also offers an NFS server.

### 5.6 WEB

World Wide Web documents are distributed via the Hypertext Transport Protocol running over TCP. Clients request "web pages" from the server and the server responds with the data. The HTML (hypertext markup language) provides rich formatting information that allows creation of good looking pages with formatted text and tables. HTML also provides "forms." Forms allow the client to interactively set and retrieve variables on the server. Quadros Systems offers an HTTP server module. The server can be run in a diskless environment and special code can dispatch forms processing to 'C' functions.

### 5.7 DHCP

Dynamic Host Configuration Protocol is a client server application that allows the client to obtain an IP address lease and/or network configuration parameters. The DHCP Client automatically handles responses from multiple servers. Quadros Systems offers both client and server modules for DHCP.

### 5.8 POP3

Post Office Protocol Client provides APIs to check if there are mail messages queued on the server and to retrieve the sites, subject, sender and content of the messages. When the content of a mail message is returned, it is passed into sections by MIME type. An API is also provided to delete mail messages from the server.

## 5.9 SMTP

Simple Mail Transfer Protocol Client provides an API to send mail messages to an SMTP server. Mail messages consist of a 'from' field, a 'subject' field, a list of recipients, and an optional message body and optional attachment. The API consists of a single subroutine that returns success or failure. If the routine fails, an errno facility indicates why.

## 5.10 SYSLOG

The Syslog facility provides a very flexible mechanism for writing error report messages, diagnostics, etc. to local or remote logging facilities. The SYSLOG client can write your formatted messages to a multitude of targets simultaneously:

Any UNIX machine can act as a suitable syslog server, while excellent FREE packages for MSWindows are downloadable from the World Wide Web, enabling the ease and flexibility that comes with remote logging using a SYSLOG client.

## 5.11 RIP

Routing Information Protocol is a simple distance-vector based protocol which allows routers to share route information. RTXC Quadnet supports both versions 1 and 2 of RIP. Input packet processing is handled through a user-invoked daemon task, while output processing occurs in the timer task.

## 5.12 NAT

Network Address Translation is a router protocol which will convert local network addresses (and port numbers if NATP is on) to unique addresses on the WAN. The addresses used locally are unique to the LAN but are also used by other LANs.

# 6 Configuration

RTXC Quadnet can be preconfigured at compile time or setup to allow run-time configuration. Configuration parameters are stored in several .h files. The following protocol subsystem variables can be configured:

- o Number of sockets and interfaces to support
- o Sizes of buffers and tables
- o Timing parameters for ARP, RARP, BOOTP, TCP, etc.

Include files are used to add or exclude protocols and features from RTXC Quadnet. This will effect RAM and ROM sizes.

# 7 Device Driver Interface

Each entry in the device table defines a device that may be opened by calling either *xn\_interface\_open* or *xn\_attach*. At initialization the device table is empty; entries must be added. Each entry contains a device id (major device number) and a minor device number. These values are used to select the device. Each entry also contains default configuration parameters for the device. For ethernet devices, these default parameters are the I/O address, interrupt number, shared memory region if required and cable type if required. Multiple instances of a driver can be supported with multiple table entries.

The following device routines are supported:

- o Device Open
- o Device Send
- o Send Complete Interrupt
- o Device Statistics
- o Device Receive
- o Device Set Multicast List

RTXC Quadnet also supports UART device driver routines.

# 8 Network Management

## 8.1 Routing Information Protocol

The Routing Information Protocol (RIP) is a simple protocol for sharing route information between directly connected gateways and hosts. It is based on a distributed algorithm which iteratively calculates optimal routes between any two hosts within the RIP domain by having each gateway advertise to each neighbor all the routes its knows of. With each route, there is an associated cost metric, which may be, but is not necessarily, number of hops to the destination address. When a route advertisement (RIP response) is received, routes with lower cost metrics than those currently stored in a node's routing table are redirected through the router which claims the lower metric.

RTXC Quadnet supports both versions 1 and 2 of RIP. Version 1 describes routes based solely on destination IP address and a total route cost metric. Version 2 adds additional information such as network mask, first hop address, and a route tag, as well as providing a mechanism for authentication of RIP packets.

## 8.2 Domain Name System

The RTXC Quadnet Domain Name Server Interface contains routines to obtain IP addresses and name information from a domain name server. The DNS interface also includes other associated routines and utilities such as retrieving domain names, initializing host cache, and adding entries to the host table.

## 9 Error Logging

RTXC Quadnet includes a facility to log error information and progress information to a console, an RS232 port, or, in a DOS environment, or to a disk file (a:debug.out). This facility is very useful for debugging new system ports and applications. Four levels of logging are provided.

- **Level zero** produces no diagnostic or error report output and forces compile time exclusion of all diagnostic reporting.
- **Level one** produces output only when errors occur, such as dropped packets, check sum errors etc.
- **Level two** produces the output of level one as well as application layer diagnostic messages such as reporting when a server connection is established.
- **Level three** produces all of the output of level two plus it reports on the progress of the inner workings of the network stack.

## 10 More Information

For more information on RTXC Quadnet Networking Software, RTXC Quadros Real-time Operating System, supported processors and other products including an embedded file system, and an embedded database, please contact Quadros Systems at [sales@quadros.com](mailto:sales@quadros.com) or call your local Quadros sales representative.