An Introduction to Networking

Understanding TCP/IP

Quadros Systems, Inc.

www.quadros.com

Table of Contents

1 What is a Networking Protocol?

A Networking Protocol is defined as a set of rules describing how to transmit data across a network. These rules consist of

- How the data must be formatted so that sender and receiver are speaking the same language
- How much data can be carried
- If and how the sender can be sure that the recipient has actually received the data
- Any additional information that the protocol needs to send to the remote side through routers and switches

2 RFCs – Protocol Standards

These rules for networking protocols are outlined in detailed specifications called Request For Comments (RFCs). An RFC is submitted by anyone who thinks that some change or addition to the Internet protocol suite needs to happen. This proposed RFC is sent to the appropriate working group at the Internet Engineering Task Force (IETF) for comments by others who are involved with that working group. If others in the working group agree with the proposed idea, then they may ask for some changes and adopt the RFC as a "Standards Track" document. If they reject the idea, then the RFC can still be published as an "Informational" RFC. Standards Track documents describe what the protocol should do in all implementations. Informational RFCs describe what a vendor would like others to implement. There are other types of RFCs, but most are beyond the scope of this document.

RFCs can also be updated to accommodate new needs or to correct problems or limitations in the existing specification. In this case, the older RFC is obsoleted by the new RFC. In an ideal world every node would comply with all of the latest standards as laid out in the RFCs. However, the reality is that in this constant state of change there are nodes in different stages of compliance.

On a regular basis, the IETF takes a snapshot of the entire RFC database and publishes a single document that states the current set of compliant RFCs that describe "the Internet". This RFC is known as "Internet Official Protocol Standards." For a node on the Internet to be fully compliant then it must meet the latest specification of every RFC for the protocols that it supports.

3 The TCP/IP Protocol Stack

Internet Protocol (IP) is the basic communications language of the Internet. IP provides the packet delivery system. TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are the main carrier protocols that use the Internet to transmit data from one node to another.

TCP is used when a semi-permanent connection is required between two nodes and also offers the ability to confirm that sent data has been received by the peer. UDP is used typically because it is faster for sending and receiving data, but it lacks the delivery confirmation and retransmission features of TCP.

TCP is used for the majority of desktop applications that we are familiar with, such as email, the World Wide Web and file transfers. However, alongside TCP/IP connections, many other protocols use UDP as an efficient way of transferring large amounts of (non-critical) data. See sections 5, 6, and 7 for more detailed information on IP, TCP and UDP.

Most Internet communications use a client-server model. What this generally means is that one node on the Internet has data and becomes the server of that data while another node requires that data and becomes the client. A familiar example of this is the WWW, where the client is the Web browser running on your PC and the server is elsewhere, hosting a file system full of data for the client to browse.

Each node on the Internet is assigned an IP Address. This address is a unique global address that allows any one node to address data directly to another node. Both TCP and UDP use this address, but combine that address with what is known as a port number. The port number is analogous to a postal number at a street address. By combining IP address and port number, very exact data exchanges can be achieved. The port number is generally used to specify a well known protocol, such as email or WWW. The combination of IP address and port number would therefore make a protocol statement such as "I am sending an email to Node X", or "I am requesting a page from Web Server Y".

The TCP/IP and UDP/IP protocols, although they are the core of the Internet, cannot, by themselves, deliver the rich functionality we experience with the Internet. Higher layer protocols, such as HTTP (Web) and FTP (file transfer) and SMTP/POP3 (email) are an important part of the story. We will provide a brief overview of these protocols in section 9.

Stack Architecture

The TCP/IP architecture is based on the Open System Interconnection (OSI) reference model (ISO 7948). In this distributed, layered architecture, each layer provides a set of functions that can be controlled and used by the functions in the layer above it. This independent operation allows modifications or alterations to be made to one layer while limiting the effect on other layers. Each layer can communicate with its peer layer on a remote machine by using the layers below it in the local "Stack."

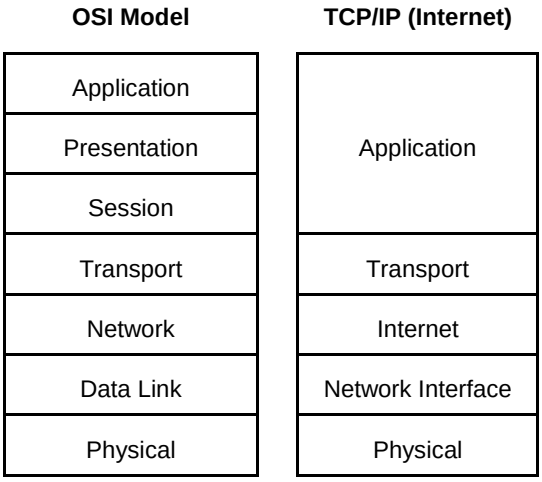

Figure 1. Stack Architecture Comparison

The five TCP/IP stack layers perform the following functions:

- **Physical Interface** Specify the mechanisms for a node on a network to interface to the transmission media. Cables, plugs, sockets, wireless, etc.
- **Network Interface** Provides error-free transmission of data between nodes on the same network.
- **Internet Protocols** Provides transmission of chunks of data (datagrams) between two nodes on the Internet. This level is more concerned with the path the packets take not the content of those packets. ***(IP)***
- **Transport Protocols** Transports data from one node to another for a specific application ***(TCP and UDP)***
- **Application Protocols** Consume and produce the data that is sent across the Internet

4 Ethernet Protocol

The Ethernet protocol is essentially the method by which two machines, physically connected on the same Local Network, can communicate with each other. For simplicity, we will group all of the most commonly used physical networks under this single heading whether wired or wireless, broadcast or not.

In the majority of situations, this means that a machine on a network can send and receive frames of data to and from the gateway machine.

The gateway machine can be seen as the only way out of the Local Network to the Internet and in most cases, the gateway acts as a protocol converter also, for example converting between Ethernet and DSL Physical Networks.

So the Ethernet Protocol can be seen as the way that one node on a local network can transfer data that is destined for the Internet to the Gateway Node.

In most cases, an Ethernet frame will have a header that contains the source and destination addresses of the sender and recipient on the local network plus a payload of data that will invariably contain an IP datagram (see next section). A size and checksum for the payload is included in the header and ensures that the data travels across the local network without error, or at least that errors will be detected.

5 IP – Internet Protocol (RFC 791)

The Internet Protocol is the basis for the universal connectivity we know as the Internet because all Internet communications use the same protocol structure. Let's take a closer look at this fundamental protocol.

Connectionless Packet Delivery Service

The Internet Protocol is essentially a packet delivery system. Technically, it is defined as *an unreliable, best effort connectionless packet delivery system*. UDP delivery is not guaranteed. The packet may be lost, duplicated, delayed, or delivered out of order. The service will not detect such conditions, nor will it inform the sender or receiver. The service is called connectionless because each packet is treated independently from all others. A sequence of packets sent from one computer to another may travel over different paths, or some may be lost while others are delivered. Best-effort delivery means that the Internet software makes an earnest attempt to deliver packets and does not discard packets arbitrarily. Unreliability arises only when resources are exhausted or underlying networks fail.

IP provides three important definitions. First, the IP protocol specifies the exact *format* of all data as it passes across the Internet. Second, IP software performs the *routing* function, choosing a path over which data will be sent. Third, in addition to the precise, formal specification of data formats and routing, IP includes a set of rules that embody the idea of unreliable packet delivery. The rules characterize how hosts and routers should process packets, how and when error messages should be generated, and the conditions under which packets can be discarded.

IP Formatting — Datagrams

The Internet calls its basic transfer unit an *Internet datagram*, sometimes referred to as an *IP datagram* or merely a *datagram*. Similar to an Ethernet frame, a datagram is divided into header and data areas. Also like a frame, the datagram header contains the source and destination addresses and a field type that identifies the contents of the datagram. The difference is that the datagram header contains IP addresses for the global Internet whereas the frame header contains physical addresses on the local network.

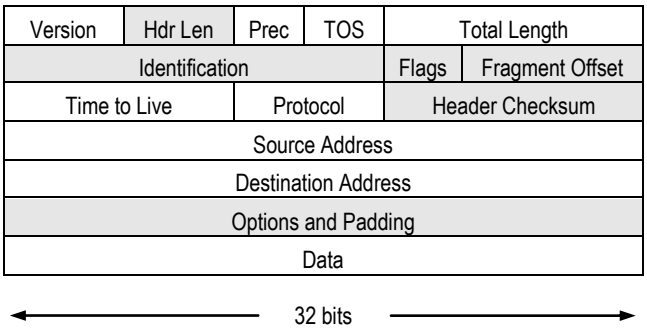

Figure 2. The IP Datagram Header

Time to Live

The Time to Live value helps ensure that undelivered IP datagrams are eventually discarded, rather than being continually forwarded across a network. Each packet is given a fixed time to live by the source machine. When a router forwards a packet it decrements that packet's TTL value by one or (often) more. When a packet's TTL reaches zero it is discarded.

IP Addresses

The 32-bit IPv4 address is grouped eight bits at a time, separated by dots, and represented to network users in decimal format (known as dotted decimal notation). Each bit in the octet has a binary weight (128, 64, 32, 16, 8, 4, 2, 1). The minimum value for an octet is 0, and the maximum value for an octet is 255.

Subnets

Each octet within the IP address can be used to further refine and locate the final destination. So an IP address of a.b.c.X can be described as node X within subnet c, which is within subnet b, which is within subnet a.

Though this notation has some more detailed control within it, it can be seen that, in most cases, node a.b.c.**X** is on the same **local** network as node a.b.c.**Y**, whereas node a.b.**c**.X is on a different local network to node a.b.**d**.X.

IP communication between the two nodes in the first example would be directly between them over the Ethernet network, whereas communication between the two nodes in the second example would go through the gateways on each of the local networks and most likely would be routed over the Internet between the two gateway machines.

IP Routing and Packet Processing

IP Process Steps

Sending data across the network using IP requires multiple steps potentially involving multiple gateways, routers and networks along the way. For each physical network the IP datagram is re-encapsulated within a frame for that network. Also, every network type has a defined maximum transmission unit (MTU) which is the largest IP datagram that it can carry in a frame. So if the packet size exceeds the MTU value, the packet is divided up into fragments. The *Identification* field in the IP header contains a unique identifier (defined by the sending host) so that the fragmented packets can be reassembled properly at the destination. Here is a simplified description of the process:

Datagram Preparation: The sending application program prepares its data and calls on its local IP software module to send that data as a datagram and passes the destination Internet address and other parameters as arguments of the call, along with the data.

Addressing and Sending: The IP module prepares a datagram header and attaches the data to it. It then determines a local network address for this destination address (in this case, the address of a gateway.) It sends the datagram and the local network address to the network layer.

Routing: The network layer creates a local network header (e.g. Ethernet), and attaches the datagram to it, then sends the result via the local network. The datagram arrives at a gateway host wrapped in the local network header, the network layer of the gateway node strips off this header, and turns the datagram over to the local IP module which determines from the internet address that the datagram is to be forwarded to another host in a second network. It determines a local net address (perhaps ATM addressing over a DSL connection) for the destination host and calls on the network layer for that network to send the datagram.

This local network interface creates a local network header and attaches the datagram, sending the result to the destination host. At this destination host the datagram is stripped of the local net header by the network layer and handed to the receiving IP module.

Hopping: The process of forwarding from one local network to the next is known as hopping. Each gateway machine will determine which local network to forward the IP datagram onto.

Although most local networks may only have one gateway node to the Internet, the architecture allows for intermediate nodes to be connected to many local networks simultaneously. These gateways that are connected to multiple local networks are referred to as routers. Within a router, for any IP datagram received, the router decides which local network is the best route for the next hop of the journey toward the final destination.

Routers are typically dedicated to that task and use all of their available processing power maintaining routing tables such that they can quickly forward an incoming IP datagram on the best route toward its destination.

Receiving: Finally, the IP datagram is delivered to the local network of the destination node and the gateway send the IP datagram to the node over the local network. The receiving IP module determines that the datagram is for this host because the destination IP address matches this node's IP address. It passes the data up the stack to the TCP or UDP layers above and most likely to final destination within an application above TCP or UDP.

6 UDP – User Datagram Protocol (RFC 768)

The User Datagram Protocol is one of the primary mechanisms that application programs use to send datagrams to other application programs. UDP messages contain both destination port numbers and source port numbers in addition to the data being sent. The combination of source and destination port numbers allows multiple simultaneous communications between the same two IP nodes.

UDP utilizes the underlying IP layer to transport information from one machine to another. By using the IP format for delivery, UDP inherits the same characteristics of an IP message. It is an unreliable, connectionless datagram service. UDP messages can be lost, duplicated, or arrive out of order. UDP packets can also be sent faster than the recipient can process them leading to lost data when any available buffering mechanisms are exhausted.

UDP, however, is connectionless. This analogous to using a delivery service such as UPS or FedEX to send packages. You divide up what you want to send to the recipient into a number of boxes and take them to the shipping company where the clerk gives each box an identity and sends it on its way. In most cases, the boxes will travel together on the same ship or plane and will arrive at their destination on the same day. However, there is nothing to prevent each of the packages taking completely different routes and arriving at the destination on different days. Eventually all boxes are likely to arrive at their destination, but not necessarily in the order they were given to the shipper. There is always a chance that one or more boxes may be lost in transit and not arrive.

Applications that use the UDP protocol are generally either able to cope with the loss of some data or unable to use the virtual-circuit method employed by TCP (see Section 7).

In essence, UDP sacrifices reliability for speed. An application using UDP accepts full responsibility for the problems of reliability, including data loss, duplication, delay, and even out-of-order delivery. However, there are applications for which UDP provides distinct advantages. In the case of voice and video applications, speed is more important than reliability. Because packets are transmitted at such a high rate of speed, we are willing to sacrifice the reliability for rate of transfer and retransmission of lost data would cause unacceptable delays in the delivery of data.

If we were using a reliable method for voice and video, we would have to wait for two machines to exchange information to ensure that the information will not be lost, sacrificing time. However, if we were to make the same connection using UDP there would be a steady steam of packets entering the application, because there is no use of acknowledgements. If two packets were to get discarded or sent out-of-order, or even lost, the result that we might see would be very minimal. If the UDP application provides simple buffering of the data received along with sequence information within the data, then the majority of problems with UDP can be resolved (other than the total loss of data). If your application can cope with loss of data then UDP is, in most cases, the best way to transmit data between two IP nodes.

7 TCP – Transmission Control Protocol (RFC 793)

The first thing to remember is that TCP is a connection-oriented protocol. This is different from UDP, as described above, where there is no delivery acknowledgement. TCP is understood to be a very reliable protocol because of this key difference.

TCP connections can be thought of as similar to a telephone call. You know the number you want to call and the called party gets the number of the caller and can decide whether to answer or not. If the party answers, you have formed a connection to that party. His or her voice coming through your earpiece provides an acknowledgement that the connection is active and information is being transferred.

The TCP Process

Reliable delivery service guarantees to deliver a stream of data from one machine to another without duplication or loss. But how can TCP provide reliable data transfer if the underlying communication system (IP) offers only unreliable packet delivery? *Positive acknowledgement with retransmission* describes the technique used by TCP to make sure all packets are received. The receiving TCP node sends an acknowledgement that it has received the sent data. If the acknowledgement is not received by the sender within a defined time period the data is resent.

Here is a description of the process:

Open Connection: On the sending node, an application issues a request to send data to a destination node. TCP creates an initial segment designed to open the connection between the sender and the receiver. In this initial contact, the two systems create a *virtual circuit* by exchanging IP addresses and port numbers. During this initial exchange,

they also setup the flow control and sequencing

Flow control: The destination node informs the receiving node of the number of bytes it is willing to accept at one time. This is known as the window size. The window is applied to the data stream and the window moves along the data stream as the sent data is confirmed as being received. This way, the source node does not send more data than the other can handle, but can slowly move through the data to be transmitted until the entire data stream is completed.

Sequencing: The first data byte in every connection is assigned a sequence number. As segments of the data are sent from the window, the sequence number is incremented by the number of bytes transmitted. This gives the receiving host the information it needs to reorder segments that arrive out of sequence. Sequencing also allows duplicate received data to be discarded.

Acknowledgement: When the sending TCP transmits a segment it does not move the window until the receiving TCP issues an acknowledgement (ACK) that it has received the data. When the segment at the beginning of the window is acknowledged, the window is able to advance along the data stream by the number of bytes acknowledged as received by the destination node. If the sending TCP node does not receive an acknowledgement, within a calculated time limit, it retransmits the data but the window does not advance.

Error detection: A checksum value in the header lets the receiving TCP test the integrity of an incoming segment. If the segment is corrupted, the receiver sends an error message to the sending TCP which then retransmits the segment.

Connection closing: When the application on the sending host is ready for the connection to be terminated the sending TCP sends a segment that tells the receiving TCP that no more data will be sent and the socket should be closed.

Transmission Delays

Because TCP is a reliable service, transmission will be delayed whenever a bit error or data loss occurs. This delay is caused by the requirement for TCP to retransmit the corrupt or lost data along with any successive data that may have already been sent. These delays may be unacceptable for certain applications such as audio and video streaming. For this reason A/V streams often use UDP transmission and deal with any packet loss in the application.

Congestion Management

In addition to handling error checking and loss, TCP is also responsible for managing the rate at which data is sent. TCP can detect network congestion and will reduce transmission rate if there are too many data losses by reducing the window of data that is sent without acknowledgement. With millions of connected nodes all using TCP, this protects the network from collapse due to too much congestion.

As a particular route across the Internet becomes congested, all TCP connections will reduce the amount of data that they send. With this mechanism, less data will be lost and less retransmissions will occur, thereby reducing the congestion. All of the TCP connections will suffer from reduced rates of transfer but retransmissions will also be reduced thereby making more efficient use of the available resources. If congestion is occurring, then generally routers will start to divert IP datagrams via alternative routes, thereby resolving the situation and allowing TCP to increase window size once more.

8 Berkeley Sockets API

The Berkeley Sockets 4.4 API (Applications Programming Interface) is a set of standard function calls made available at the application level. These functions allow programmers to include Internet communications capabilities in their products. The Berkeley Sockets API (also frequently referred to as simply ‘sockets’) was originally released with 4.2BSD in 1983. Enhancements have continued through to the current 4.4BSD system. Other sockets APIs exist, though Berkeley Sockets is generally regarded as the standard.

The socket interface allows communications between hosts (or between processes on one computer) using the concept of an Internet socket.

Sockets Overview

BSD Sockets generally relies upon client/server architecture. For TCP communications, one host listens for incoming connection requests and is regarded as the connection server. When a request arrives on that socket, the server will accept the connection, at which point data can be transferred between the hosts.

A TCP/IP connection is uniquely described using the source node’s IP address and port number combined with the Destination node’s IP address and port number.

Applications can create sockets which can then be attached to a port. Once an application has created a socket and bound it to a port, data sent to that port will be delivered to the application that is listening to that socket.

Basic Steps

- Initialize a socket
- Bind the socket to a port
- Listen (indicate readiness to receive a connection)
- Accept a connection
- Send and/or receive data

- Close a socket

9 Other Important Protocols

TCP, UDP and IP are just a part of the networking story. Other fundamental protocols play an important part in defining node addresses,

Core Networking Protocols

The following protocols work alongside IP to support assigning, finding, and resolving IP addresses:

ARP

Address Resolution Protocol. Computers wishing to communicate must know each other's physical address (or MAC address). IP addresses are an abstraction to make networks hardware-independent. The actual physical address of the destination machine must be "resolved" from its IP address before any packets can be sent. ARP handles the resolution of IP addresses into physical addresses.

An ARP request is broadcast to discover the MAC address for a machine with a specified IP network layer address. After resolving the address, the host creates a cache of IP-to-MAC mappings The ARP caches expires over time. A new ARP request is required to refresh the expired entry which then detects any changes to the topology since the last ARP.

DNS

Domain Name System. The distributed name/address mechanism used in the Internet. Translates Internet domain names or URLs (e.g. www.quadros.com) into IP addresses. Largely used because IP addresses are hard to remember but also because IP addresses may change if a server is moved from one local network to another.

DHCP

Dynamic Host Control Protocol. Dynamic, temporary assignment of IP addresses from an address pool.

A DHCP server automatically sends a new IP address when a computer is plugged into a different place in the network.

A DHCP client will broadcast a request for an IP address when it is connected to a network. The first DHCP server to respond will provide the DHCP client with an IP address. The DHCP server will also generally supply a gateway address through which to communicate outside the Local Network along with an indication of the IP addresses on the local network.

ICMP

Internet Control Message Protocol. Part of the IP protocol used to handle errors and control messages. Hosts send ICMP messages to report delivery problems to originating machines and to probe for the existence of other hosts. The ping command uses ICMP

IGMP

Internet Group Multicasting Protocol. Part of the IP multicasting protocol. IGMP allows the transmission of an IP datagram to a "host group", a set of hosts identified by a single IP destination address.

NAT

Network Address Translation. Router protocol that translates an Internet Protocol address (IP address) used within one network to a different IP address known within another network. Typically, used by a company to map its many local network addresses to one or more global Internet addresses and then unmap the global IP addresses on incoming packets back into local IP addresses. NAT helps conserve on the number of global IP addresses that a company needs and it lets the company use a single IP address in its communication with the world.

RARP

Reverse Address Resolution Protocol. Used by a diskless host to find its Internet address at startup. The host requests its IP address from a gateway server's Address Resolution Protocol (ARP) table.

Application Protocols

FTP

File Transfer Protocol. Allows for the transfer of files between two network nodes over a TCP/IP connection. FTP uses the client/server method so the FTP client initiates the procedure, sending the parameters, type and direction of the data transfer and FTP commands to the server. (See also TFTP)

HTTP

Hypertext Transfer Protocol. Protocol for moving hypertext (HTML) files across the Internet over multiple TCP/IP connections. Requires a HTTP client (typically a browser) on one end and an HTTP (web) server on the other.

TFTP

Trivial File Transfer Protocol. TFTP is a simplified alternative to FTP for transferring files over networks. Unlike FTP, TFTP uses UDP as its transport mechanism. As such, performance is low and TFTP is typically only used between nodes on the same local network. TFTP includes only a minimum number of commands and does not support comprehensive data integrity features. Data is exchanged between client and server one datagram at a time

with acknowledgement at the application layer.

10 IPv4 vs. IPv6 (RFC1883)

Because of the growth of the Internet we are running out of IP addresses to assign to machines. Many private networks handle this problem by assigning virtual addresses to machines in their network (see NAT). The new IPv6 expands the number of available IP addresses by a multiple of 10^{28} over IPv4 addresses.

IPv6 is an enhancement of IPv4 that uses a new header format to provide a more flexible network protocol with a much larger address space. Other changes which improve "plug and play" capability include stateless self-configuration, router discovery, neighbor discovery, multicast discovery, and management protocols.

IPv6 uses a simplified header structure to reduce packet processing overhead. It also improves the handling of extensions and options, allowing for more efficient packet forwarding today and making it easier to make changes in the future. One of the many other enhancements is the ability to label packets that require special handling, specifically packets that are time-sensitive or require a higher priority. This is called flow-labeling capability.

Key differences:

Function	IPv4	IPv6
ARP	Separate protocol	Embedded into the IP protocol
Address types	Unicast, multicast, broadcast	Unicast, multicast, anycast
Configuration	IP addresses and routes must be pre-configured	Self-configuring
DHCP	Dynamically obtain an IP address	Not supported
NAT	Network address translation used to assign virtual IP addresses	Not supported or necessary with IPv6
RIP	Routing Information Protocol	Not supported; uses static routes

To date, IPv6 has been deployed primarily in Asia where IP addresses are scarce.

11 RTXC Quadnet TCP/IP Networking Software

This high performance TCP/IP protocol suite, combined with the RTXC Quadros real-time operating system, gives you everything you need to deploy an Ethernet- or Wi-Fi-connected device. Serve up web pages with device status, download firmware updates, perform remote device diagnostics and maintenance, send e-mail alerts, and much more.

This suite of networking protocols is specifically designed to bring networking functionality to embedded systems with the high performance and ease-of-use you need. The RTXC Quadnet TCP/IP stack is available for IPv4, IPv6 or as a dual IPv4/IPv6 stack. Basic protocols include TCP, IP, UDP, ARP, ICMP, DNS, and DHCP/BOOTP with a Berkeley Sockets API.

Additional networking protocols such as IGMPv2, RIPv2, NAT, AutoIP and PPP are available as add-on modules. Available application level protocols include HTTP (web) server, FTP and TFTP (file transfer), Telnet, SMTP/POP3 (email), and SNMPv1, 2, and 3.

Key Features

- **High Performance.** Zero-copy design and tight coding increase processing speed and overall system efficiency. Zero copy is an efficient way of transferring packets during transmit and receive through the sockets. Instead of having to make an extra copy of the packet data to/from the application buffer to/from the stack buffer, a pointer to the data is passed to either the application, or the stack and the Device Driver.
- **Support for IPv4 and/or IPV6.** Available as IPv4 stack, IPv6 stack or dual IPv4/v6 stack. IPv6 stack is fully compliant with all TAHI conformance test suites including IPv6 Neighbor Discovery, IPv6 Path MTU Discovery, IPv6 Stateless Address Auto-Configuration (Prefix Discovery), IPv6 Robustness, IPv6 Specification.
- **Fully RFC-compliant.** Tested according to the Automated Network Validation Library (ANVL) to ensure RFC compliance. Your product will work most efficiently on the Internet if it is compliant with all of the latest standards.

  In addition to those RFCs that are regarded as essential for a node to be Internet compliant, many more RFCs are specified to improve interoperability and performance. Some RFCs directly relate to the performance of the node, while others relate to the performance of the connected network. A quality product must be efficient in itself but must also make efficient use of the attached network.

The Quadnet stack includes the implementation of several optional RFCs that specifically improve the performance of your product on the Internet.

- **Designed for Embedded Systems.** Written specifically to meet the needs of embedded applications; not repurposed from publicly available stacks. Using the available resources efficiently will ensure that less data is lost and therefore that retransmissions are reduced and communications remain at the highest rate of transfer.
- **Real-time Design**; Tightly Integrated with RTXC Quadros RTOS. Designed for real time, deterministic processing with low latency, preemption, reentrancy, timer granularity. The stack uses very small, deterministic critical sections. No function calls or loops are allowable inside a critical section, which prevents a task that is running the stack from blocking a higher priority task. The stack is also reentrant, preventing memory corruption during interrupt processing.
- **Flexible and Configurable.** Choose only the RTXC Quadnet components you need. Then use options within each component to configure RTXC Quadnet to meet the needs of your application.
- **Extensively Tested.** The Quadnet stack has undergone rigorous testing procedures, including PC-Lint testing, to ensure error free builds when using any ANSI C compiler. To ease implementation and provide a high-level of compliance, Quadnet has also been tested according to the Automated Network Validation Library (ANVL) to ensure RFC compliance and TAHI conformance testing (IPv6).
- **Auto Configuration at Runtime.** Automatic discovery of memory/resource requirements and device environment at runtime, eliminating the need to recompile the stack. The result is higher performance, less wasted memory, and easy set-up.
- **ROMable.** Allows non-variable data to be stored in ROM, reducing the usage of expensive RAM memory.
- **Large Library of device drivers.** Quadnet TCP/IP includes device drivers for the most popular Ethernet, serial and WiFi controllers.

12 More Information

For more information on RTXC Quadnet Networking Software, the RTXC Quadros RTOS, and other QSI products, including FAT and flash file systems, UPnP, USB, SDIO, CAN, GUI tools and more, please contact Quadros Systems at sales@quadros.com or call your local Quadros sales representative.