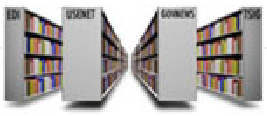

TCP/IP FAQ; Frequently Asked Questions (1999-09) Part 2 of 2

Part1 - Part2

TCP/IP FAQ; Frequently Asked Questions (1999-09) Part 2 of 2

From: tcp-ip-faq@eng.sun.com (TCP/IP FAQ Maintainer)
Newsgroups: comp.protocols.tcp-ip
Subject: TCP/IP FAQ; Frequently Asked Questions (1999-09) Part 2 of 2
Date: 7 Sep 1999 03:38:45 GMT
Message-ID: <tcp-ip-faq-2.1999-09@eng.sun.com>
Summary: Part 2 of a 2-part informational posting that contains
responses to common questions on basic TCP/IP network
protocols and applications.

Archive-name: internet/tcp-ip/tcp-ip-faq/part2
Version: 5.15
Last-modified: 1999-09-06 20:11:43
Posting-Frequency: monthly (first Friday)
Maintainer: tcp-ip-faq@eng.sun.com (Mike Oliver)
URL: http://www.itprc.com/tcpipfaq/default.htm

TCP/IP Frequently Asked Questions

Part 2: Applications and Application Programming

This is Part 2 of the Frequently Asked Questions (FAQ) list for the comp.protocols.tcp-ip Usenet newsgroup. The FAQ provides answers to a selection of common questions on the various protocols (IP, TCP, UDP, ICMP and others) that make up the TCP/IP protocol suite. It is posted to the news.answers, comp.answers and comp.protocols.tcp-ip newsgroups on or about the first Friday of every month.

The FAQ is posted in two parts. Part 1 contains answers to general questions and questions that concern the fundamental components of the suite. Part 2 contains answers to questions concerning common applications that depend on the TCP/IP suite for their network connectivity.

Comments on this document can be emailed to the FAQ maintainer at <tcp-ip-faq@eng.sun.com>.

- -

Table of Contents

- -

What Are The Common TCP/IP Application Protocols?

1. DHCP

Dynamic Host Configuration Protocol (DHCP) allows IP addresses to be allocated to hosts on an as-needed basis. The conventional scheme of allocating a permanent fixed IP address to every host is wasteful of addresses in situations where only a relatively small number of hosts are active at any given time. DHCP lets a host 'borrow' an IP address from a pool of IP addresses; when the address is no longer needed it is recycled and made available for use by some other host. DHCP also allows a host to retrieve a variety of configuration information at the same time as it acquires an IP address.

DCHP depends on UDP to carry packets between the client and server tasks.

DHCP is defined by RFC's 2131 and 2132. A widely-used implementation of DHCP can be downloaded from <http://www.isc.org/dhcp.html>.

2. DNS

The Domain Name System (DNS) provides dynamic on-demand translation between human-readable names (like www.pizzahut.com) and the numeric addresses actually used by IP (like 192.112.170.243). The basics of DNS operation are defined in RFC's 1034, 1101, 1876, 1982 and 2065.

DNS uses both UDP and TCP. It used UDP to carry simple queries and responses but depends on TCP to guarantee the correct and orderly delivery of large amounts of bulk data (eg transfers of entire zone configurations) across the network.

DNS standards are discussed in the comp.protocols.dns.std newsgroup. A very widely-used implementation of DNS called BIND (Berkeley Internet Name Domain) is discussed in the comp.protocols.dns.bind newsgroup, and the BIND software itself can be downloaded from <http://www.isc.org/bind.html>. The operation and politics of DNS are discussed in the comp.protocols.tcp-ip.domains newsgroup.

3. FTP

File Transfer Protocol (FTP) provides a mechanism for moving data files between systems. In addition to the fundamental PUT and GET operations, FTP provides a small number of file management and user authentication facilities. The protocol is defined in RFC 959.

FTP depends on TCP to guarantee the correct and orderly delivery of data across the network.

4. HTTP

Hyper Text Transfer Protocol (HTTP) is the protocol used to move Web pages across an internet. Version 1.0 of HTTP is defined by RFC 1945. Version 1.1 makes more efficient use of TCP and is defined by RFC 2068.

HTTP depends on TCP to guarantee the correct and orderly delivery of data across the network.

5. IMAP

Interactive Mail Access Protocol (IMAP) allows clients to manipulate email messages and mailboxes that reside on some server machine. The current version of IMAP is Version 4, usually referred to as IMAP4. IMAP4 is described by RFC 2060. IMAP provides no way of sending email; client programs that use IMAP to read mail usually use SMTP to send messages. IMAP is more powerful and more complex than the other widely-used mail-reading protocol POP.

IMAP depends on TCP to guarantee the correct and orderly delivery of data across the network.

IMAP is discussed in the comp.mail.imap newsgroup.

6. NFS

Network File System (NFS) allows files stored on one machine (the "server") to be accessed by other machines (the "clients") as though the files were actually present on the client systems. NFS is defined in terms of a Remote Procedure Call (RPC) abstraction which in turn formats its packets according to a processor-independent eXternal Data Representation (XDR).

Version 2 of NFS is defined in RFC 1094 and Version 3 is defined in RFC 1813. The RPC mechanism most often used with NFS, ONC/RPC, is defined by RFC 1831. The XDR conventions used by ONC/RPC are defined by RFC 1832. The ONC/RPC binding mechanism (a minimal directory service which allows RPC clients to rendezvous with RPC servers) is defined by RFC 1833.

NFS can run over any kind of transport, but is most often used over UDP. UDP does not guarantee packet delivery or ordering, so when NFS runs over UDP the RPC implementation must provide its own guarantees of correctness. When NFS runs over TCP, the RPC layer can depend on TCP to provide this kind of correctness.

NFS is discussed in the comp.protocols.nfs newsgroup.

7. NNTP

Network News Transfer Protocol (NNTP) is used to propagate netnews postings (including Usenet postings) between systems. It is defined in RFC 977. (The format of netnews messages is defined in RFC 1036.)

NNTP depends on TCP to guarantee the correct and orderly delivery of data across the network.

NNTP is discussed in the news.software.nntp newsgroup. A very widely-used implementation of NNTP called INN (InterNet News) can be downloaded from <http://www.isc.org/inn.html>.

8. NTP

Network Time Protocol (NTP) is used to synchronise time-of-day clocks between various computer systems. The current version of NTP is Version 3, defined in RFC 1305. Earlier versions (2 and 1 respectively) of the protocol are defined in RFC's 1119 and 1059. David Mills maintains a publically-available implementation of NTP server and clients along with a comprehensive collection of NTP documentation on the web at <http://www.eecis.udel.edu/%7Entp/>.

NTP depends on UDP to carry packets between server and client tasks.

NTP is discussed in the comp.protocols.time.ntp newsgroup.

9. POP

Post Office Protocol (POP) allows clients to read and remove email from a mailbox that resides on some server machine. The current version of POP is Version 3, usually referred to as POP3. It is described by RFC 1939. POP provides no way of sending email; client programs that use POP to read mail usually use SMTP to send messages. POP is simpler and less powerful than the other widely-used mail-reading protocol IMAP.

POP depends on TCP to guarantee the correct and orderly delivery of data across the network.

POP doesn't have its own dedicated newsgroup. It is sometimes discussed in client-specific newsgroups in the comp.mail.* hierarchy.

10. Rlogin

Remote Login (rlogin) provides a network terminal or "remote login" capability. Rlogin is similar to Telnet but it adds a couple of features that make it a little more convenient than Telnet.

Rlogin is one of the so-called Berkeley r-commands, (where the "r" stands for "remote") a family of commands created at UC Berkeley during the development of BSD Unix to provide access to remote systems in ways that are more convenient than the original TCP/IP commands.

The most obvious convenience is that rlogin, like other r-commands, examines a .rhosts (pronounced "dot ar hosts") file on the server side to authenticate logins based on the client host address. The .rhosts file can be constructed to allow remote access without requiring you to enter a password. If used improperly this feature can be a security threat, but if used correctly it can actually enhance security by not requiring a password to be sent over the network where it might be read by a packet sniffer.

The r-commands depend on TCP to guarantee the correct and orderly delivery of data across the network.

11. Rsh

Remote Shell (rsh) is an r-command that provides for remote execution of arbitrary commands. It allows you to run a command on a server without having to actually log in on the server. More importantly it allows you to feed data to the remote command and retrieve the command's output without having to stage the data through temporary files on the server.

Like other Berkeley r-commands, rsh uses the .rhosts file on the server side to authenticate access based on the client's host address.

On some non-BSD systems the Remote Shell command is named remsh because by the time the command was delivered on those systems the usual rsh name had been used for a "restricted shell" application, a command line interpreter intended to boost security by preventing its users from performing certain activities.

On Unix systems most of the work of rsh is handled by the rcmd(3) library function, so if you're writing a program that needs rsh-like functionality then you might be able to use that function. However, since the rsh protocol requires the client to use a privileged port you'll only be able to use rcmd(3) if your program executes with superuser privileges. That's why the rsh executable is setuid-root on Unix machines.

If your program will not run as root then you might be able to use the rexec(3) function instead. rexec(3) does not use the server-side .rhosts file. Instead it requires the client to supply an account password which is then transmitted unencrypted over the network.

12. SMTP

Simple Mail Transfer Protocol (SMTP) is used to deliver email from one system to another. The basic SMTP is defined in RFC 821 and the format of Internet mail messages is described in RFC 822.

SMTP depends on TCP to guarantee the correct and orderly delivery of data across the network.

A very widely-used implementation of SMTP called sendmail can be downloaded from <http://www.sendmail.org/>. Other open-source SMTP implementations include qmail (available at <http://www.qmail.org/>) postfix (available at <http://www.postfix.org/>), smail (available at <ftp://ftp.planix.com/pub/Smail/>), exim (available at <http://www.exim.org/>) and smtpd (available at <http://www.obtuse.com/smtpd.html>).

13. SNMP

Simple Network Management Protocol (SNMP) provides a means of monitoring and managing systems over a network. SNMP defines a method of sending queries (the GET and GET-NEXT primitives) and commands (the SET primitive) from a management station client to an agent server running on the target system, and collecting responses and unsolicited event notifications (the TRAP primitive).

Version 1 of SNMP is defined by RFC's 1098 and 1157. SNMP Version 2 is defined by RFC's 1441, 1445, 1446, 1447 and 1901 through 1909. The various things that can be monitored and managed by SNMP, collectively called the Management Information Base (MIB) are defined in dozens of additional RFC's.

SNMP sends traffic through UDP because of its relative simplicity and low overhead.

SNMP is discussed in the comp.protocols.snmp newsgroup.

14. Ssh

Secure Shell (ssh) provides remote login and execution features similar to those of the rsh and rlogin r-commands, but ssh encrypts the data that is exchanged over the network. Encryption can protect sensitive information, and it is not uncommon for security-conscious administrators to disable plain rsh and telnet services in favour of ssh.

The SSH protocol used by the ssh command has also been used to build a secure file transfer application which can be used as an alternative to FTP for sensitive data.

Complete information on ssh and its SSH protocol can be found at <http://www.ssh.fi/>.

15. Telnet

Telnet provides a network terminal or "remote login" capability. The Telnet server accepts data from the telnet client and forwards them to the operating system in such a way that the received characters are treated as though they had been typed at a terminal keyboard. Responses generated by the server operating system are passed back to the Telnet client for display.

The Telnet protocol provides the ability to negotiate many kinds of terminal-related behaviour (local vs. remote echoing, line mode vs. character mode and others) between the client and server. The basic Telnet protocol is defined in RFC's 818 and 854 and the option negotiation mechanism is described in RFC 855.

Specific Telnet options, implementation issues and protocol quirks are discussed in several dozen RFC's dating back to 1971. (That's RFC's 97, 137, 139, 206, 215, 216, 318, 328, 340, 393, 435, 466, 495, 513, 559, 560, 562, 563, 581, 587, 595, 596, 652, 653, 654, 655, 656, 657, 658, 698, 726, 727, 728, 732, 735, 736, 748, 749, 779, 856, 857, 858, 859, 860, 861, 885, 927, 933, 946, 1041, 1043, 1053, 1073, 1079, 1091, 1096, 1097, 1143, 1184, 1205, 1372, 1408, 1411, 1412, 1416, 1571, 1572 and 2066, and that's not counting obsolete ones. A couple of these are not entirely serious.) As you might infer from this pedigree, Telnet is a widely-deployed and well-used protocol.

Telnet depends on TCP to guarantee the correct and orderly delivery of data between the client and server.

16. X Window System

The X Window System (X11R6 is the most recent incarnation) allows client programs running on one machine to control the graphic display, keyboard and mouse of some other machine or of a dedicated X display terminal.

X depends on TCP to guarantee the correct and orderly delivery of data across the network.

The X Window System is discussed in the comp.windows.x newsgroup.

- -

TCP/IP Programming

1. What are sockets?

A socket is an abstraction that represents an endpoint of communication. Most applications that consciously use TCP and UDP do so by creating a socket of the appropriate type and then performing a series of operations on that socket. The operations that can be performed on a socket include control operations (such as associating a port number with the socket, initiating or accepting a connection on the socket, or destroying the socket) data transfer operations (such as writing data through the socket to some other application, or reading data from some other application through the socket) and status operations (such as finding the IP address associated with the socket).

The complete set of operations that can be performed on a socket constitutes the Sockets API (Application Programming Interface).

If you are interested in writing programs that use TCP/IP then you'll probably need to use and understand the sockets API. Your system manuals may have a description of the API (try `man socket' if you're using a Unix system) and many books devote chapters to it. A FAQ list for sockets programming is available on the Web from its Canadian home at <http://www.ibrado.com/sock-faq/>, from a UK mirror at <http://kipper.york.ac.uk/%7Evic/sock-faq/> or by anonymous FTP from <ftp://rtfm.mit.edu/pub/usenet/news.answers/unix-faq/>.

The TLI (Transport Layer Interface) API provides an alternative programming interface to TCP/IP on some systems, notably those based on AT&T's System V Unix. The Open Group, a Unix standards body, defines a variation of TLI called XTI (X/Open Transport Interface). Note that both sockets and TLI (and XTI) are general-purpose facilities and are defined to be completely independent of TCP/IP. TCP/IP is just one of the protocol families that can be accessed through these API's.

2. How can I detect that the other end of a TCP connection has crashed? Can I use "keepalives" for this?

Detecting crashed systems over TCP/IP is difficult. TCP doesn't require any transmission over a connection if the application isn't sending anything, and many of the media over which TCP/IP is used (e.g. Ethernet) don't provide a reliable way to determine whether a particular host is up. If a server doesn't hear from a client, it could be because it has nothing to say, some network between the server and client may be down, the server or client's network interface may be disconnected, or the client may have crashed. Network failures are often temporary (a thin Ethernet will appear down while someone is adding a link to the daisy chain, and it often takes a few minutes for new routes to stabilize when a router goes down) and TCP connections shouldn't be dropped as a result.

Keepalives are a feature of the sockets API that requests that an empty packet be sent periodically over an idle connection; this should evoke an acknowledgement from the remote system if it is still up, a reset if it has rebooted, and a timeout if it is down. These are not normally sent until the connection has been idle for a few hours. The purpose isn't to detect a crash immediately, but to keep unnecessary resources from being allocated forever.

If more rapid detection of remote failures is required, this should be implemented in the application protocol. There is no standard mechanism for this, but an example is requiring clients to send a "no-op" message every minute or two. An example protocol that uses this is X Display Manager Control Protocol (XDMCP), part of the X Window System, Version 11; the XDM server managing a session periodically sends a Sync command to the display server, which should evoke an application-level response, and resets the session if it doesn't get a response (this is actually an example of a poor implementation, as a timeout can occur if another client "grabs" the server for too long).

3. Can the TCP keepalive timeouts be configured?

This varies by operating system. There is a program that works on many Unices (though not Linux or Solaris), called netconfig, that allows one to do this and documents many of the variables. It is available by anonymous FTP from <ftp://cs.ucsd.edu:/pub/csl/Netconfig/>.

In addition, Richard Stevens' TCP/IP Illustrated, Volume 1 includes a good discussion of setting the most useful variables on many platforms.

4. Are there object-oriented network programming tools?

Yes. One such system is the ADAPTIVE Communication Environment (ACE). The README file for ACE is available on the Web at <http://www.cs.wustl.edu/%7Eschmidt/ACE.html>. All software and documentation is available via both anonymous ftp and the Web.

ACE is available for anonymous ftp from <ftp://ics.uci.edu/gnu/>. That's a compressed tar archive approximately 500KB in size. This release contains contains the source code, documentation, and example test drivers for C++ wrapper libraries.

- -

Part1 - Part2

[Usenet FAQs | Web FAQs | Documents | RFC Index]

Send corrections/additions to the FAQ Maintainer:
tcp-ip-faq@eng.sun.com (TCP/IP FAQ Maintainer)

Last Update June 29 2010 @ 07:58 AM