NOTE: This Technical Note has been retired. Please see the Technical Notes page for current documentation.

Technical Note NW550

MacTCP Q&As

CONTENTS

This Technote contains a collection of archived Q&As relating to a specific topic--questions sent to Developer Technical Support along with answers from the highly-trained DTS engineers. Current Q&A's can be found on the Macintosh Technical Q&A's web site.

[Oct 01 1990]

OK to abort a MacTCP name resolver call

Date Written: 9-Mar-1993

Last reviewed: 5-Jun-1998

Is it possible to abort an outstanding MacTCP name resolver call before it completes? The `CloseResolver` call description says that "[b]efore the application exits, the `CloseResolver` call must be made to release memory structures and terminate all outstanding domain name server calls." The very next sentence says that "`CloseResolver` must not be called until all outstanding resolver calls have been completed." Which of these statements is correct? If it's the latter one, then does that mean that if I make a `StrToAddr` call and it returns cache fault that I have to wait for the result proc to be called before my application can quit? How long is that likely to be?

There is no way to cancel an asynchronous name resolver request. You must not call `CloseResolver` until all outstanding name resolver calls are complete. You must call `CloseResolver` before quitting your application. Therefore, if you have an outstanding DNR request and the user selects *Quit* , your application must delay quitting until the name resolver request has completed.

You can see this behavior in many existing MacTCP applications, which put up a modeless dialog that says "Waiting for DNR requests to complete before quitting" if you attempt to quit them while there are outstanding name resolver requests.

The timeout for name resolver calls is hard-coded to 37 seconds (MacTCP) or about 2 minutes (Open Transport MacTCP backward compatibility); if no response is received after that period, the call will complete with an error. You should not rely on these timeout values in your application, since they are subject to change.

You can circumvent this restriction by using the Open Transport Internet services interface, which allows you to cancel asynchronous requests by simply calling `OTCloseProvider` on the Internet services provider.

One other thing to be aware of is that, although you can perform multiple name resolver lookups simultaneously, MacTCP has a hard, global limit of 4 simultaneous outstanding name resolver requests. In general, DTS recommends that your application not issue more than one outstanding name resolver request; otherwise, you may prevent other applications from using the name resolver. Furthermore, if a name resolver request returns the `outOfMemory` error code, you may want to retry the request at some later date.

Again, these restrictions do not apply to the Open Transport Internet services interface.

Finally, the interrupt-safe nature of the MacTCP name resolver is documented in Technote 1104: Interrupt-Safe Routines.

Maintaining a listening server socket on MacTCP

Date Written: 1-Feb-1993

Last reviewed: 5-Jun-1998

I need to consistently listen on the same port for incoming connections. This behavior is analogous to the Listen and Accept functionality provided by the industry-standard socket interface for TCP on UNIX systems. That is, I listen on a port/socket for incoming connections; when a remote system attempts to connect, a new socket is created representing that connection, and the port/socket performing the listen continues to listen for other connection requests.

When using the `TCPPassiveOpen` operation under MacTCP, the listen operation doesn't persist once an incoming connection is established. That is, the stream used to listen for an incoming request becomes the connection with the remote system. Once that connection is established, there's no stream listening for subsequent connection requests. Is there any way I can keep a listener active at all times? My application needs to have a listener constantly, or at least with as short a window of unavailability as possible.

You're right that MacTCP's non-socket interface has no way to maintain a persistent listener using MacTCP. To emulate this functionality, you'll have to keep more than one listener active at all times. You can do this by queuing multiple `TCPPassiveOpen` commands asynchronously to the MacTCP driver. How many to queue depends on how fast you expect connections to come in and how quickly you can re-queue new opens after connections are made. Three listeners should probably be enough if you expect a relatively small number of connections and you can re-queue new parameter blocks as you receive connections.

`PBControlAsync` is safe to call at interrupt time in this case. Any asynchronous calls to MacTCP will queue up and wait, so it's safe to post them from an interrupt or an ASR. However, since there isn't an ASR event for Notify, I agree that using an ASR isn't a solution for you in this case.

There is some sample code you might find useful. On the Toolchest Developer CD, under the path ":Sample Code:Snippets:Networking:TCP Server:" you'll find a sample server that uses TCP.

MacTCP over ARA requires MacIP gateway

Date Written: 10-Dec-1992

Last reviewed: 5-Jun-1998

When I dial into my AppleTalk network none of my MacTCP-based software works. I have MacTCP configured with an IP number on the remote machine, and Telnet (a MacTCP-based application) runs fine, but I can't connect to any of the hosts on the network I'm dialed into. What could be causing this problem?

The problem may occur because you don't have a MacIP gateway on the dial-in net to act as a proxy IP encapsulation service. When you select the AppleTalk icon (such as LocalTalk, EtherTalk, or Dial-In) in the MacTCP control panel, you're using a MDEV (IP Link layer) that encapsulates the IP data into an AppleTalk packet and sends it out. You therefore need to have something on the other side that will break out the IP data from the AppleTalk packet and send it out as an IP frame on the network you're dialed into.

There are several third-party solutions to this problem, the Shiva FastPath and Cayman Gator Boxes being the most common. The feature is officially called MacIP, but it is also known as IP Encapsulation or KIP. If you use MacTCP this way, you must select the zone the MacIP gateway is in via the pop-up menu in the MacTCP control panel. You must also configure the MacIP gateway to give out IP numbers, and set the remote Macintosh to "Server" IP addressing in the MacTCP control panel.

Note: Two Mac OS computers using MacIP should be able to communicate peer to peer, but they won't be able to contact non-MacIP devices.

Definition: MacIP: The act of taking IP data frames (packets) and placing them inside an AppleTalk packet for transport over an AppleTalk-only media (also known as LocalTalk and ARA). MacIP is also known as KIP or KSTAR, which are both names left over from the Kinetics FastPath. The protocol specification is available online.

MacTCP ARP limits are hard-coded

Date Written: 22-Sep-1992

Last reviewed: 5-Jun-1998

Is it possible to increase ARP cache size? It appears that MacTCP has memory to cache 35 to 40 entries. If my application sends out IP requests periodically to about 40 unique IP stations, MacTCP drops some entries, and makes an ARP request. Is there any way to program the cache size to a higher limit?

Unfortunately, the number of ARP table entries is hard-coded to 20 in the MacTCP source code, and the ARP time-out for aging entries out of the ARP cache is also hard-coded to 10 minutes. There's no way to change these values, since they are

in the code as #defines and used throughout.

Under Open Transport, these values are tunable using the OT Advance Tuner.

How to cancel transactions under MacTCP

Date Written: 7-Dec-1990

Last reviewed: 5-Jun-1998

How do I cancel transactions (specifically connect requests) under MacTCP? I know that I can specify a timeout; however, I would like the user to be able to hit "Command-." if the transaction's tired of waiting. I tried `KillIO` but that seems to crash.

MacTCP does not support the `KillIO` call to abort a transaction. You should use `TCPRelease`. `TCPRelease` aborts the current activity and releases the stream. As an alternative, a `TCPPassiveOpen` can be canceled by issuing an `TCPActiveOpen` from the same machine on the listening port.

Note: There is no need to call both `TCPAbort` and `TCPRelease`. `TCPRelease` performs a `TCPAbort` before releasing the stream.

Note: You should call `TCPRelease` once--and only once--for each stream you create using `TCPCreate`.

Porting a TCP/IP application to Mac OS

Date Written: 19-Mar-1991

Last reviewed: 5-Jun-1998

We need the following to port our mainframe front-end application to the Mac OS:

- a C++ compiler compatible with AT&T 2.0 (and ANSI C)
- a TCP/IP library with an Application Programming Interface (API) similar to sockets
- an Ethernet card

Apple's Mac OS developer tools, including a C++ compiler, are available online. There are also a number of third party C++ compilers for Mac OS.

Sockets is the general UNIX (Berkeley) API for TCP/IP programming. Apple currently offers some libraries that allow you to work with MacTCP but not APIs similar to sockets. There are, fortunately, a number of third-party socket libraries for MacTCP.

Ethernet cards are available from a number of sources, including Apple.

MacTCP gateway address and subnet mask fields

Date Written: 7-May-1991

Last reviewed: 5-Jun-1998

What are the MacTCP Subnet Mask and Gateway Address fields used for?

In the TCP/IP protocol, there are a couple of provisions for simplifying the addressing scheme for smaller local networks. One of these is the subnet mask. It allows the sender to identify the destination machine by a shorter address, which is long enough to distinguish all the machines on the local network. This mask is made up of four octets, and is used as a bitwise mask to show how many bits are being used to specify the network number, subnet number, and node. The gateway address field specifies the IP address of the local network's router. Packets destined for another network are sent to this router, which then transmits the packets to other networks.

For more details on the TCP/IP protocol suite, see *Internetworking with TCP/IP, Volume 1, Principles, Protocols and Architecture* , 3rd ed., Douglas E Comer, Prentice Hall, 1995, ISBN 0-13-216987-8.

Communicating different processes on the same Mac with MacTCP

Date Written: 5-Jun-1991

Last reviewed: 5-Jun-1998

Is it possible to have both the source and the destination application on the same machine if they are using MacTCP to

communicate? If so, what should I do to make sure both applications share the processor so this works properly?

It is possible to use MacTCP to communicate to different processes on the same Mac OS computer. Unlike PPC, MacTCP does not bypass the network for same-machine connections, so there may be some "network traffic" involved, depending on your selected link-layer.

As far as "sharing" the processor goes, this is *very* important in what you are doing. Here's a brief outline of what you need to do:

- Make asynchronous MacTCP driver calls;
- If you want to use polled I/O, you can wait for `ioResult` < 1 to tell when the call completes. Otherwise, you may want to have a completion routine which queues completed calls for processing at event time; or
- Call `WaitNextEvent` or `EventAvail` while waiting for driver calls to complete (this gives time to the other MacTCP applications).

For a good reference on using "idle procs" to give time to background applications when using MacTCP, refer to the article, "MacTCP Cookbook" in *develop,* Issue 6 and get John Norstad's NewsWatcher source code.

How to tell how many of `StrToAddr`'s network addresses are valid

Date Written: 30-Aug-1991

Last reviewed: 5-Jun-1998

Using `StrToAddr`, you can receive up to four IP network addresses returned by the service for the host requested, but the MacTCP Programmer's Guide does not say how to determine the number of valid addresses. According to the name resolver code, MacTCP zeros out 4 long words before the `StrToAddr` lookup and then fills in the addresses it finds. This means that the array contains *at most* 4 addresses, and is terminated by a zero address (0.0.0.0) if fewer than four addresses are returned.

MacTCP resolver code is in domain name resolver

Date Written: 17-Sep-1991

Last reviewed: 5-Jun-1998

Is the MacTCP resolver code contained in the "MacTCP DNR" file?

The resolver code is contained in the "MacTCP DNR" file, as well as in the MacTCP control panel. The DNR code is stored as a resource file in the System Folder and is read into memory as necessary in order to convert host names into addresses. It is used in conjunction with a domain name server or with the local hosts file.

MacTCP Hosts file location and function

Date Written: 17-Sep-1991

Last reviewed: 5-Jun-1998

Where should the MacTCP Hosts file be located, and who looks for that file?

According to the MacTCP 1.0 Documentation Kit (APDA #M0217LL/A), the Hosts file should be located in the user's System Folder. The file maps machine names to Internet addresses, providing the same service as the domain name server. The Hosts file can be used if you have no domain server on your network. It is also suggested that this file be used for frequently used name-to-address mappings. Instructions on setting up the file are included in the MacTCP documentation.

MacTCP "obtain address" options

Date Written: 24-Sep-1991

Last reviewed: 5-Jun-1998

Under MacTCP, what is the significance of the "obtain address" options, with respect to (1) manually, (2) server, and (3) dynamically?

According to the MacTCP 1.0 Documentation Kit (APDA #M0217LL/A), "obtain address" means the following:

1. Manually -- You will need to fill in some or all of the fields in the IP Address box;
2. Server -- The Internet Protocol (IP) address for the user Macintosh is automatically obtained from a server the first time MacTCP is opened after a restart. This option requires:

    - Reverse Address Resolution Protocol (RARP) or Bootstrap Protocol (BootP) server on an Ethernet, or
    - MacIP gateway on an AppleTalk network that's compatible with the MacIP protocol as defined online.
3. Dynamically -- The node portion of the IP address for the user Macintosh is set dynamically the first time MacTCP is opened after a restart. MacTCP chooses a random address and tries to confirm whether another machine is using that address. If it is, it chooses another random address. The process repeats until an unused address is found. With this option you still need to set some of the fields in the IP Address Box.

All these processes are described in further detail in the MacTCP documentation.

Note: Most non-Mac OS systems label what MacTCP calls Server addressing as "dynamic" addressing, and do not support MacTCP's Dynamic addressing method.

MacTCP and SLIP

Date Written: 6-Nov-1991

Last reviewed: 5-Jun-1998

Does MacTCP actually support SLIP? If not, does Apple plan to provide SLIP support in the future. Does anyone know of any company that does a SLIP for the Mac OS?

SLIP is an asynchronous, serial line protocol developed for running TCP/IP over serial communications lines in a point-to-point configuration. SLIP was developed to transmit IP packets over low-speed, sometimes noisy, asynchronous communications lines where error recovery and an efficient line protocol are needed. The SLIP protocol is now being replaced with a new serial line protocol named "PPP," which uses a more efficient means of establishing a point-to-point IP connection.

MacTCP includes hooks that let you write different link-layer modules. This makes possible the development of interfaces to SLIP, PPP, and to any other link layer that someone may need, like broadband, X.25, and FDDI. Apple does not currently provide support in MacTCP for SLIP or any other serial line protocol. There are, however, several third-party SLIP and PPP extensions available for use with MacTCP, and Apple now bundles an Open Transport-compatible PPP implementation with Mac OS.

MacTCP `StrToAddr` and header's `hostInfo` record structure

Date Written: 12-Dec-1991

Last reviewed: 5-Jun-1998

I'm using MacTCP 1.1 with Think C, and when calling `StrToAddr` with hostname = "90.25.3.240", the returned addr field has the following:

```
addr [0] = 0X00005A19
```

The correct answer should be addr `[0] = 0X5A1903F0`. What's the problem?

The `StrToAddr` problem you are experiencing is related to a shortcoming in the header files. The structure for the `hostInfo` record you are accessing (from "AddressXlation.h") is as follows:

```
typedef struct hostInfo {
    int rtnCode;
    char cname[255];
    unsigned long addr[NUM_ALT_ADDRS];
```

The problem with the way the structure is defined is that `rtnCode` is defined as an "int." The size of an int is dependent on your development environment. In MPW, ints are 4 bytes, while Think C uses 2 byte ints as its default. If you're using Think C, you will see the addresses shifted forward in the storage by 2 bytes, since the `rtnCode` structure causes the rest of the struct to be shifted forward in memory.

If you're using Think C 5.0 or later, you can either check the "Use 4 byte ints" check-box in the preferences, or you can change the header files to use "long" in place of "int" (a Think C long is the same size as an MPW int). Alternatively, you should upgrade to Universal Interfaces, which contains MacTCP headers that are compatible with most Mac OS C compilers, including Think C.

MacTCP EnumCache data enumeration sequence

Date Written: 12/12/91

Last reviewed: 5-Jun-1998

Does MacTCP's EnumCache name resolver call return the data randomly or in a particular order?

Since the EnumCache documentation doesn't specify any particular sequence for name resolver cache enumeration, it's best to play it safe and not assume any particular order for the data returned by the call. Since the internals of the name resolver may change in the future (for example, Open Transport MacTCP backward compatibility), you probably don't want to rely on any undocumented data ordering assumptions. If you want the entries separated, the best way probably would be to have a dynamically sized array for each of the entry types (such as addresses, HInfo, and name servers) and fill them in the EnumCache callback.

MacTCP addressing modes and Shiva Fastpath

Date Written: 12-Dec-1991

Last reviewed: 5-Jun-1998

When I attempt to connect to a LocalTalk net using MacTCP 1.1 after opening the MacTCP driver, this error occurs: "Error opening driver / Error in getting address (-23004)". I'm running System 7.0 on a Macintosh IIci with Shiva Fastpath 4.0, and I've configured TCP Admin as follows:

1. Indicate Resident Zone
2. Click More
3. Dynamic addressing
4. Range 1 to 65534
5. Enter Gateway Address (address of Fastpath for resident zone)
6. Enter Domain Name & Address

The Fastpath won't work with MacTCP configured to use dynamic addressing. You need to set MacTCP to use server-based addressing. If you change this setting, and the FastPath is configured correctly, you should be up and running.

MacTCP 1.1 and sending urgent data per RFC 1122 or 793

Date Written: 7-Apr-1992

Last reviewed: 5-Jun-1998

MacTCP 1.1 packets containing "urgent data" have the so-called "urgent pointer" set according to RFC 1122. All our TCP/IP hosts expect a behavior corresponding to RFC 793, however. Is there a possibility to configure MacTCP to this effect?

MacTCP can send urgent data according to either RFC 1122 or RFC 793; however, it is not a setting in the driver that determines this. When a programmer makes the TCPSend call, they set a flag in the parameter block that indicates whether or not to use urgent data, and which version of urgent data to use. Therefore, you would have to modify the program that you are using to set up the parameter blocks appropriately.

To send urgent data using the non-compliant RFC 793 method, set the urgentFlag field of the TCPSend parameter block to 2. Any other non-zero value of urgentFlag indicates the RFC 1122 method should be used.

MacTCP 1.1 ICMP echo request documentation bug

Date Written: 13-Feb-1992

Last reviewed: 5-Jun-1998

I'm trying to use the ICMP echo request function described on page 88 of the MacTCP Developer's Kit 1.1 manual to "ping" a TCP host before trying to open a connection. The problems I'm having seem centered around the ICMPEchoNotifyProc. If I declare the callback as

the routine gets called and the ICMPParamBlockPtr is valid. But after the callback returns, my computer crashes. Can you give me any suggestions as to how to make this puppy work?

You've found a bug in the manual. The `ICMPEchoNotifyProc` is actually defined as:

The definition is correct in "MacTCP.h".

Note that in this definition, the callback uses C-, not Pascal-calling conventions. In C, the caller removes the parameters from the stack, not the callee, and since your routine has already removed the parameters before returning, they are removed twice, corrupting the stack. Changing your procedure to use C-calling conventions should fix your problem.

MacTCP 1.1 & 1.0.1 compatibility

Date Written: 13-Feb-1992

Last reviewed: 5-Jun-1998

We have implemented a TCP/IP product using MacTCP Development Kit version 1.1. If MacTCP version 1.0.1 is installed, will there be any compatibility problems with our implementation.?

There are only a very few new calls in MacTCP 1.1 that aren't in the 1.0.1 driver. Among these are:

- Name resolver `HInfo` and `MXInfo` calls
- UDP multi-port sends and receives
- ICMP echo protocol support

As long as you're not using any of these features, any program developed for MacTCP 1.1 will work with 1.0.1.

MacTCP 1.1 header file incompatibilities and fixes

Date Written: 30-Sep-1991

Last reviewed: 5-Jun-1998

MacTCP 1.1, Apple's System 7-compatible version of MacTCP, fixes several problems in earlier releases, but doesn't address several header file incompatibilities. The latest MacTCP header (which have all been rolled into a single file, "MacTCP.h") are available as part of Universal Interfaces.

No Pascal headers for MacTCP

Date Written: 19-Mar-1992

Last reviewed: 5-Jun-1998

I want to access MacTCP in a Pascal program. Do you have the C parameter block definitions available as Pascal records or will I have to do my own translation?

"MacTCP.p" is available as part of Universal Interfaces. However, programming MacTCP in Pascal is still a challenge because of problems compiling the "dnr.c" file and implementing C callbacks in Pascal. Libraries and sample that show how to call MacTCP from Pascal are available from Stairways Shareware. For more hints and tips about Pascal programming on Mac OS, check out Pascal Central.

MacTCP Type of Service documentation fix

Date Written: 21-Apr-1992

Last reviewed: 5-Jun-1998

Are MacTCP's Type Of Service (Reliability, Low Delay, and Throughput) bit settings swapped in the documentation?

The bit settings in the MacTCP programmer's documentation are reversed for the Type of Service constants. The bit settings should be as follows:

Type of Service Byte (3-bit field)
Bit 0 set for low delay
Bit 1 set for high reliability
Bit 2 set for high throughput

If you wish to correct these settings they can be found on page 35 of the *MacTCP Programmer's Guide* .

MacTCP ULP timeout documentation fix

Date Written: 5-May-1992

Last reviewed: 5-Jun-1998

In the MacTCP Developer's Kit (Version 1.1) there is a ULP timeout action field which is used in several PowerBook calls to MacTCP. For some calls, zero indicates an abort and nonzero indicates a report. `TCPSend`, `TCPClose` and `TCPStatus` are this way. For other calls, the reverse is documented (0 indicates report and 1 indicates abort). `TCPActiveOpen` and `TCPPassiveOpen` are documented this way. Can I believe the documentation or are there errors in it? If there are errors, what is the correct way it should be ?

According to the MacTCP source code, you're right, the documentation is incorrect. For each of the calls with a ULP timeout action, the correct values are:

Value	Action
1	abort
0	report only

Mac OS Communications Toolbox MacTCP tools

Date Written: 6/10/92

Last reviewed: 5-Jun-1998

My application uses the Mac OS Communications Toolbox and ADSP to connect to network services. What software do I need to get to make the same connections using TCP/IP? MacTCP or other recommendations?

Apple only makes one MacTCP Communications Toolbox tool that works specifically with MacX and TCP/IP (the TCP Tool). This tool is not supported for use by third party developers. For CTB/MacTCP connection tools which are supported, you need to get a third-party CTB MacTCP tool. These are available from several vendors. For example, TCPack from ASC (Advanced Software Concepts) is a set of Connection Tools designed to allow simultaneous TCP/IP connections using Apple's MacTCP driver. All Apple's terminal tools (asc3270, asc5250, etc.), the Apple standard terminal tools (TTY, VT100, VT320), as well as several third-party terminal tools can be used to enter terminal sessions with TCPack. TCP/Tools from InterCon Systems is another package which allows you to use existing CTB applications to log onto Unix workstations using your LocalTalk or EtherTalk network.

Back to top

Downloadables

 Acrobat version of this Note (K) Download

Back to top
