Test TFTP server and client for the Q68 UDP driver

To check the correct functioning of the UDP driver, I programmed small (and primitive) TFTP server / client programs, under SMSQE and Java. These are by no means polished applications, rather a rough draft of how this could be implemented. The SMSQE programs are simple uncompiled (but QSAVEd) basic programs – the actual file transfer is made via a special m/c extension, however.

I very strongly recommend that you read the UDP driver manual and this manual before using these programs.

Table of Contents

1 TFTP

TFTP is a very primitive file transfer protocol.

The idea of this is that you have a (remote) server to which clients upload files and from which clients download files. So the clients contact the server and

- either tell it the name of a file that the server is to save on its local disk (file uploaded to the server by the client)
- or tell it the name of the file they want to obtain (file downloaded from the server)

In both cases, the name of the file must be the entire name (including any device or directory). The java program is there so that you can test this between a PC/Mac and a Q68 (they must both be on the same LAN). Alternatively you could use SMSQmulator (preview version – please contact me if you need this for testing).

Note that the TFTP protocol only provides for these two operations, you can't, for example, get the directory of a disk on the server or client. Note that the conventional TFTP port on which a TFTP server listens to incoming connections is port **69**. I have found, however, that on some PCs that port is already occupied, which is why I chose port 55555 as the preset port in the UI.

If you use a PC as a server/client, do not forget to open a port in the firewall!!!

2 Installation

I've tried to keep the SBasic and Java applications similar in appearance (within the limits of the respective window managers). Whilst the Java program is a single file, the SBasic part comes in different files.

2.1 Installing the SMSQE programs

2.1.1 Using the supplied TFTP.WIN container file

The supplied container file **TFTP.WIN** is a "win" disk which can be used on the Q68 and emulators. It contains everything you need to run and use the programs, it also contains the sources for the SMSQE programs (they're QSAVEd SBasic programs) and the UDP driver (asm_ip directory).

You should probably rename this file to whatever you name your normal qxl.win type file for win1_ (so that you can boot right off it – there is a supplied boot program), then simply copy it to the Q68 SD card.

If used on an emulator, proceed similarly.

2.1.2 Installing the SBasic programs manually

These come as five “QSAVEd” SBasic programs (but the _bas version is also supplied):

- The main server (server_sav).
- The alternative main server (server2_sav).
- The specific server instance (servinst_sav)
- The client (client_sav).
- The user interface (UI_sav)

As you can see, a formidable amount of thought went into the names for these programs…

Copy the programs to wherever you want them. The only proviso is that they all must be in the same directory. I’d suggest you put them on a ram disk, but this is just so that they load a little bit faster.

The UI_bas is a pointer environment program, the others aren’t.

Before using the programs, the following extensions must be installed:

- the driver (udp_q68_bin)
- the menu extensions (menu_rext)
- “qptr” for the UI (qptr_bin)
- “outptr”, also for the UI (outptr_bin)
- “tftp” for the actual file ops (tftp_bin)

The names in parentheses refer to the names of the files in the TFTP.WIN disk (“extn” directory).

Note: the SBasic programs can also be used in what will be the next version of SMSQmulator (and probably QPC). There, in addition to the extensions mentioned above, you will also need a file called “trap3_bin” (which is also in the TFTP.WIN container file and loaded directly if used from an emulator). It contains some extensions used by the programs and which are normally part of the Q68 UDP driver.

2.2 Installing the Java program

Copy the program (Aserver.jar) into the directory of your choice. The program combines the server and the client in a single file. It is configured to run on Java 8 and higher.

3 Starting the programs

3.1 SMSQE

3.1.1 Using the supplied TFTP.WIN container file

It is best to set up the Q68 so that it boots right off the container file.

When started for the first time, it launches immediately into MenuConfig, so that you can configure the driver. If you use this only for the LAN, I’d suggest that you leave all options as they are, except for the IP address : if you do not use DHCP, change the corresponding item and then you MUST enter the IP address the Q68 should have. Without IP address, the programs will not work. Please see the UDP driver manual for all of the configuration

options. The default options are: Initialize the driver automatically, don't get IP from DHCP, use full duplex, no default IP address.

Once configured, you should make sure you save the configuration options as proposed by MenuConfig : if you do that you will not be bothered with the configuration next time you start the program.

The UI_bas program is then started automatically. **Please note that whenever you run the UI program, it removes a server job listening on the same port as the one set by default in the UI program (i.e. port 55555)**.

There are two test data files on the TFTP.WIN disk : win1_test_l (Large, more than 2MiB) and win1_test_s (Small, some 50 KiB). You can use these as up- or download files. The name "win1_test_s" is already set in the stuffer buffer (Alt+Enter).

Do not forget to set the peer's MAC address if that can't be handled via ARP.

3.1.2 If you decide to run the SBasic UI program manually

You will need to:
1 – Load all the extensions mentioned above by LRESPRing them.
2 – Set the IP of the Q68 (if this isn't done automatically via DHCP). See the UDP driver manual for how to do that.

You start the programs by EXECing the UI_sav file. All SBasic programs MUST be in the same directory.

Do not forget to set the peer's MAC address if that can't be handled via ARP.

3.1.3 If you decide to run the other SBasic programs manually.

Please see **below**.

3.2 Java

Double-click on the Java program on a PC/Mac etc.

4 Using the programs from the UI

Remember, for this to work, you have to have set the Q68's IP address, either manually or via DHCP. You must also set the peer's (server's) MAC address, unless the peer reacts to ARP requests. To set the peer's MAC address manually, use the ARP_SET command: ARP_SET ip$,mac$ (see the UDP driver manual).

Both SMSQE and Java interfaces are similar, there is a part dedicated to the server, another dedicated to the client and a third part where the client and/or server display some status messages. The last part can be cleared (in SBasic via the Wipe item in the upper right corner). On the Q68, I'd suggest that you use screen mode 1 or 4 (4 colour QL mode).

4.1 Launching the server

If you want to start a server that listens to incoming requests, you should set the (**local!)** IP and the port of the server. You may also set a default directory. Please also read the section about options and the XTcc field (they all must be set BEFORE launching the server).

IP address
Note that the IP address of the machine is to be used as a server **MUST** be either the loopback address (127.0.0.1 – see the UDP driver manual for this) or the IP address of the machine that the server program is running on.

Port
The port may be any from 1 to 65534. This is not checked by the program, so if the port is outside this range or set to a port that can't be used or is not a number, then the server won't start. A port between 50000 and 60000 is generally OK. The default is 55555. The normal port to be used with TFTP is port 69.

Default directory
The server can use a default directory, to/from where it saves/loads files for downloads or uploads.

Click on the "Start server" button. A small confirmation text is shown, the button becomes a "Stop server" button and then nothing else seems to happen. That's normal, the server program is an independent job/thread that is just an infinite loop. Nothing more needs to be done. To stop the server, use the appropriate button.

As of now the server is waiting for a client to connect to it.

If you use a PC as a server, do not forget to open a port in the firewall!!!

4.2 Starting the client

If you want to start the client, proceed as follows:
1 - Configure the IP and the port of the server (use the same items as for the server) to which the client is supposed to connect.

2 - Edit or choose the names of the source and the destination files by clicking the corresponding buttons/icons. On the SMSQE side, a DO on the respective buttons open a Menu extensions window, a HIT let's you enter a name directly. Note: See below for how filenames are handled/expanded.

- The source file. If the operation is an upload (local file sent to the server) this is the name of the local file to be uploaded to the server (so this must be the name of a file the client can validly open). If this is a download, it is the name of the remote file which the server is to send to the client for download (so this must be the name of a file the server can validly open).

- The destination. If the operation is an upload (local file sent to the server), this is the name of the remote file under which the server is to save the file that is uploaded to it (so it must be able to save this file under that name). If the operation is a download this is the name of the local file under which the client saves the file downloaded from the server (so it must be the name of a file the client is able to save). When testing, if you send a file to the Q68, I'd recommend that the

destination be on a ramdisk, whence you can then copy it later to its final destination.

- **NB:** If you do not set the destination filename, the program will automatically use the source filename as the destination filename.

3 - Choose whether you want to upload or download the file.

4 – ONLY when downloading, you may also choose to delete any existing file with the same name on the client machine.

5 – Set the XTcc status. Please see below for more information on this.

Now click on the “Start client” button. The operation will start, you will see some information displayed in the window (from the server and/or the client). At the end, the client and/or the server instance just stops and quits, leaving behind the messages. If you upload a file that already exists on the server, the server will NOT overwrite the existing file but generate an error (unless you want to live dangerously - see below).

Sometimes when transferring a large (2 MiB) file, the Q68 may seem to hang a bit. Just give it some time….

6 – The “Swap” button just swaps the source and destination files.

7 – Please note: whenever you “buttonize” the window of the UI program, that program will stop the server!

4.3 Default directory and how filenames are handled

I tried to make this as painless as possible. On the Q68, both the server and the client can make use of the default directory.

On the SMSQE side, this is handled as follows:

- For the server: If a default directory has been set when the server was started (highly recommended, unless you want to live dangerously see below), the name of the default directory is always prepended to the filename passed to the server. This means that the server can only write to and read from files in that directory. If no default directory has been set, the filenames needs to be a valid QL filename.

- For the client, this is a bit more involved: if a default directory has been set for the client, then the client will use **that for the file it uses locally**: The “local file” is the file to be opened for reading when it is uploaded to the server, or the file to write into when data is to be downloaded from the server.
  However, in the request sent by the client to the server, the filename does NOT include the default directory – this allows the client and the server to have different default directories.

For the Java program, there must be a valid default directory, which must not be the root directory. Filenames are relative to that directory. Of course, the program needs to have read and/or write permissions for that directory and filenames need to be valid PC filenames.

Empty destination: if you do not set the destination filename, the program will automatically use the source filename as the destination filename. This will be a problem if both the server and the client run on the same machine and use the same default directory….

Note that the names must be in a format understood by the client and the server. Suppose the server lies on a PC and the client is run from the Q68. If you download a file from the server, the source name must be a PC name as the server will fetch the file from the PC it is running on. The destination name (client running on the Q68) must be a SMSQE name as the Q68 will save it on its file system.

4.4 Setting options

The options menu (accessible via F3 on the SMSQE program) lets you set

- the frequency in which the programs display a progress report. When up- or downloading, every umptieth block the programs may display that they sent or received block number xxx. When the client and server run from the same UI, you will see both. Default is 100.
- Number of retries. How often the machine tries to re-send (or re-receive) a package that wasn't transmitted correctly. Default is 10. The timeout for each try is 2 seconds on the java side and ½ second on the SMSQE side, so the server or client will try for a total of 20 seconds to remedy the bad transmission.
- Set the erase option for the server – this may be dangerous, so read this before doing so.
- Set whether the server to be used is the alternative server (see here why this could be necessary).

The options must be set BEFORE launching the server or client.

5 Using the SBasic server or client manually

You may also start the server or client manually without using the UI.

5.1 Starting the programs without the UI

The general way to launch these programs is to use **EX** (or EXEC) with the following parameters:

EX program_name [,#chan_nbr] ; command_line$

where:

- *program_name* is the entire path to the program to be EXECed – it can be the _bas or the _sav version.

- *chan_nbr* is the channel number of a channel opened **in the SBasic in which you use the** EX **command.** This can be a screen channel, or any other normal SMSQE channel. It **must** be preceded by "**#**" and will only be used if the "-c" command is also set. The program will print its reports into that channel. This channel is NOT closed when the program quits (as it was opened in the SBasic that EXECed the program).

- *command_line$* is the command line you wish your program to have, see just below for the structure of the command line.

Make sure you get the separators right : “,” (comma) before the channel number, “;” (semi-colon) before the command line.

5.2 The command line

The command line is formed of commands, with possible options. The structure is:

-command
or
-command<space> option.

A command is a single letter preceded by “-”. It is separated from the options, or the next command, by a space. The case of the letter does not matter. The order in which the commands are given is without importance – if you give the same command twice with contradictory options, only the last one will be used.

Many commands have default values which will be used if the corresponding parameter is not passed to the programs. The commands are as follows :

Cmd	Use	Option for command	Default values	Use with
-c	use the supplied report **C**hannel	none	don’t report **	S,C
-d	**D**ownload the file	none	none	C
-e	Erase existing files on server	none	don’t erase	S
-f	**F**requency of reports	0-3000	200	S,C
-i	**I**P address of server	the IP address	the machine IP *	S,C
-l	**L**og file	log file name	none **	S,C
-o	default direct**O**ry	default directory to use	none	S,C
-p	**P**ort to use	none	69	S,C
-r	number of **R**etries	number of retries	10	S,C
-s	**S**ource file name	File name	none	C
-t	**T**arget file name	File name	same as source	C
-u	**U**pload the file	none	none	C
-x	**X**Tcc is to be used	none	don’t use XTcc	S,C

* on the Q68 only, on other machines it’s set to the loopback address.

** if the command line contains commands to report to a channel AND to a (valid) log file, then the reporting will be done ONLY to the log file.

The “**Use with**” column in the above table shows for which program the option can be used : S = Server, C = Client. An option not destined for the corresponding program will simply be ignored.

For the server to function, it needs at least the IP and the port, which are the IP and port on which it listens. If none are supplied, it uses the default values

For the client to function, it needs at least the IP and the port of the server (if none are supplied, it uses the default values), and a source file name and and up- or download command. There are no default values for the source file and up- or download command. If only a source filename is supplied, source and target filenames are the same.

5.3 Living dangerously

It is highly recommended that, for the server at least, you **set the default directory,** especially if

- **THE Q68 IS CONNECTED TO THE INTERNET** and not only your LAN, and
- **YOU USE THE -e command line parameter.**

When the -e parameter is set for the server, this means that the server will erase an existing file whenever a client uploads a file with the same name. When the machine is accessible from the internet and this parameter is set for the server, someone could, for example, send a file called “win1_boot” to the server. That would overwrite your existing boot file, so that the next time you boot the Q68, is uses a foreign boot file. This is definitely a security risk.

➔ Is the risk real? Well, yes. If I can think of it, somebody else can think of it, too.
➔ Is the risk significant? Probably not (famous last words…).

5.4 Examples:

All examples suppose that you have set the machine’s IP address previously and that the programs lie in a directory ‘tftp’ on win1_.

5.4.1 Server

EX `win1_tftp_server_bas`
Starts the server listening on the default port 69 at the machine’s IP address (if Q68) or the loopback address (emulators)..

EX `win1_tftp_server_bas ; ”-i 127.0.0.1 -p 69”`
Starts the server listening on port 69 at the loopback address. This would be for testing purposes only.

```
OPEN_NEW#3,”ram1_logfile”
```
EX `win1_tftp_server_bas, #3 ; ”-i 172.16.0.2 -p 55555”`
Starts the server listening on port 55555 at IP address 172.16.0.2, (which is supposed to be the machine’s IP), reporting to ram1_logfile – except that it won’t report because you forgot the “-c” command.

EX `win1_tftp_server_bas, #2 ; ”-c -i 127.0.0.1 -x -r”`
Starts the server listening on default port 69 at the loopback address, reporting to SBasic’s channel #2., use XTcc. The “-r” is ignored as it has no options.

EX `win1_tftp_server_bas ; ″-i ″ & GET_MY_IP$ & ″ -p 55555 -l ram1_serverlog -o win1_tftp_data″`
Starts the server listening on port 55555 at the address of the machine, reporting to ram1_serverlog and using the directory “win1_tftp_data_“ as default directory.

EX `win1_tftp_server_bas, #2 ; ″-i 172.16.0.2 -p 55555 -c -r 34 -f 200 -l ram1_logfile -o win1_tftp_data -x -d -u -s mysource -t mytarget ″`
Starts the server listening on port 55555 at IP address 172.16.0.2, (which is supposed to be the machine’s IP), set report frequency to 200, set retries to 34, set the default directory to win1_tftp_data, use xtcc, download and upload the source and target files, report to ram1_logfile but also report to SBasic channel#2. Here, the download and upload commands are ignored not only because they are contradictory but also as they make no sense for the server itself – the same is true for the source and target files. Also, programs can only report to one destination: the conflicting instructions here (report to a log file and to a channel) are resolved in favour of the log file.

5.4.2 Client

EX `win1_tftp_client_bas.`
Will do nothing as no source filename is supplied **.**

EX `win1_tftp_client_bas ; ″s ~myfile″`
Will do nothing as neither “-d” nor “-u” is supplied.

EX `win1_tftp_client_bas ; ″s ~myfile -d″`
Starts the client, looking for the server at the default address (the machine’s address if Q68, 127.0.0.1 if emulators) and port (69). Will download the file “myfile” from the server and save it locally under the same name. This is the minimum information a command line to the client program needs to have. As a rule, however, the IP and port of the server to contact should also always be given.

EX `win1_tftp_client_bas ; ″-i 172.16.0.2 s ~myfile -u -d″`
Starts the client, looking for the server at address 172.16.0.2 and default port 69. Will download the file “myfile” from the server and save it locally under the same name. Note the conflicting “-u” and “-d” commands – as “d” comes last, this is used.

EX `win1_tftp_client_bas ; ″s myfile -d -t anotherfile″`
Starts the client, looking for the server at the default address (the machine’s address if Q68, 127.0.0.1 if emulators) and port (69). Will download the file called “myfile” from the server and save it locally under the name “anotherfile”.

EX win1_tftp_client_bas ; ″-i 172.16.0.2 -p 55555 -s ~myfile -u″
Starts the client, looking for the server at address 172.16.0.2 and port 55555, uploads the local file “myfile” and ask the server to save it under the same name.

EX win1_tftp_client_bas ; ″-i 172.16.0.2 -p 69 -s ~myfile -u -t remotename ″
Starts the client, looking for the server at address 172.16.0.2 and port 69, uploads the local file “myfile” and ask the server to save it under the name “remotename”.

EX win1_tftp_client_bas, #2 ; ″-i 172.16.0.2 -p 55555 -c -r 34 -f 200 -l ram1_logfile -o win1_tftp_data -x -d -s win1_subdir_mysource -t mytarget ″
Starts the client looking for the server on port 55555 at IP address 172.16.0.2, (which is supposed to be the machine’s IP), set report frequency to 200, set retries to 34, set the default directory to win1_tftp_data, use xtcc, download the remote source file called “win1_subdir_mysource” and save it locally as “mytarget” in the default directory, report to ram1_logfile but also report to SBasic channel#2. Programs can only report to one destination: the conflicting instructions here (report to a log file and to a channel) are resolved in favour of the log file.

5.4.3 Server instance

Do not start a server instance on its own. Use a client to connect to the server, which will then start the server instance.

6 What’s happening and why are there two server programs?

6.1 Using the normal server

The programs roughly work as follows:

If you use the server side of the program, then the server program just sits there in an infinite loop, waiting for a client to connect to it. When a client connects to it with a read (download) or write (upload) request, the server spawns a new job/thread, the “server instance”. Under SBasic, it does that by EXECing the “servinst_sav” program, passing it the correct command line.

The server instance then contacts the client and sends/gets the file to/from the client. As the file exchange proceeds, you will see some information displayed in the window (from the server and/or the client): The computer sending the file will display a message for every umptieth* block sent, the computer receiving the file will show a message for every umptieth* block received. At the end, the client and/or the server instance just stop and disappear.

There is nothing to stop you from starting the server on both machines, and using a client on each machine to connect to the server running on the other. Note also that you can mix the programs (a java server on a PC, the basic programs on the Q68, or a java client on a

PC, the basic server on the Q68). There is also nothing to stop you from starting the server and the client on the same machine.

Please note that all programs assume that file operations will be OK. If not, the operation will fail, and the server instance or the client may hang (for a time). Notably, if the file to be saved (on the server or the client) already exists, the operation will fail.

* See the options menu (F3 button in the SMSQE world).

6.2 Using the alternate server

The above scheme is all very nice and (IMHO) is exactly how things should work. But you will probably find that this isn't always the case: I have noted that when using a PC to connect, as a client, to the Q68 acting as server, this **doesn't** work : when the server instance spawned by the Q68's server after a request by the client sends some information back to the client on the PC, the client doesn't get that information. This problem seems to be due to the firewall in (at least my) PC and seems to find its origin in the fact that the the server instance uses a fresh random port to connect to the PC's client port.

My PC firewall stops packets sent thusly from getting through to the PC client. I can see that the packets do actually arrive at the PC, but the firewall seems to prevent the packets from reaching the PC's OS. Apparently, the firewall keeps a record of what peer port the PC client connected to (i.e. Q68's server port), and refuses to let data through to the client port when the data doesn't come in from the same port in the Q68. Now, I'm told that *this is not supposed to happen*, but here at least it does. This is confirmed by the fact that, if I switch the firewall off, everything works as expected – and it also happens when using a standard, built-in TFTP server.

Hence the need for an alternative ~~fact~~ server, ingeniously named server2_sav. This server will use the same (server) port for receiving the client's request and sending the data to the client. This has one advantage and several disadvantages:

- The advantage is that it works...
- The disadvantage is that the server cannot accept any new request while it is still serving a client request.
- In fact, if a second request arrives whilst the first is still being processed, the server will probably crash – remember, I consider this to be test software (even if it comes with this lengthy manual).

You can use the alternative server either by starting it directly via the command line, with the required parameters, or from the UI. If you use the UI,

7 Some practical examples

7.1 Testing the server and client on the same machine.

A a first step, you might want to have the client and server run on the same machine and talk to each other there.

For that, you set the IP address of the server to the "loopback" address, which is 127.0.0.1. Now you choose any port you want (from 1 to 65535). Note that the

conventional TFTP port on which a TFTP server listens to incoming connections is port 69. I have found, however, that on some PCs that port is already occupied (I'm not sure by what), which is why I sometimes chose port 55555 as the preset port.

Start the server.

Now set the names of the source and destination files. There is no need to set the server IP and server port again, as they are the same since we're on the same machine.

Hit the "Start client item" and you should see the transfer start.

7.2 Using the programs on two different machines

Suppose you have two machine on the same LAN : machine A, which has IP address 172.16.0.1 and machine B, which has address 172.16.0.55. You want to use port 56000.

You want A to be the server, the client is to run on B.

On machine A, the server, set the IP address to its own IP address, i.e. 172.16.0.1, and the port to 56000. Hit the "Start server" button.

Now on machine B, set the IP address to the server's IP address, i.e. 172.16.0.1, and the port to 56000.

Fill in the source and destination filenames, select down- or upload and hit the start client button.

8 XTcc

Some non SMSQE/QL programs, notably file zippers, or "C" files written on other systems for SMSQE or the QL, sometimes tack on an extra "field" at the end of a file if this file is meant to be an executable program. This field consists of two long words: a special marker ("XTcc") followed by the length of the data space.

When the XTcc button is selected, the SMSQE programs will try to honour that flag: the client program will tack this information on when it uploads an executable program to the server. When it downloads a file from the server it will check whether the XTcc field is there, and, if so, save the file as an executable file (in a slow and primitive manner). You can switch this behaviour on and off with the XTcc button.

The server program will also try to do the same – with one difference however. The server only uses the status that was passed to it when you started it. If you want to change the status of the XTcc item and have this reflected in the server, you must stop the server, set the XTcc item, and restart the server.

9 Troubleshooting

I have found that most problems, when communicating between the Q68 and a PC, stem from the different firewalls: the one in the router and the one in the PC. This is especially

true when trying to access the Q68 from the internet. Most notably, I can see that packets arrive at the PC, but aren't acted upon as the PC's firewall, for some reason, doesn't let the packets through.

I must add, that this is also true when using a bog standard TFTP server and client from two PCs.

Thus, try at first to use the programs from withing your LAN and then , if you're adventurous, switch off all firewalls (and/or configure your router to put your machines into the DMZ).

W. Lenerz
Dec. 2020