Packet vs. Internet

The experimenting Radio-Amateurs and their organizations nowadays prefer to use the Internet as their primary medium for exchanging data. Why is it that they are willing to pay for the Internet services, while a wireless digital network is available for free? Despite our ever-lasting efforts we must admit the lack of modernization on our packet network. It doesn't appear to be the slowness - even on the Internet we find an acronym for WWW: WorldWide Waiting - but the complexity (with regard to configuration and operation) of the legacy communication software that's preventing comfortable use.

One can hardly compare the features and user friendliness of Netscape and Internet Explorer with those of our Packet radio software. Just think of it: automatic mail delivery with the possibility to attach schematics or pictures with simple drag&drop, transferring large files without the need of using 7plus, automatic routing instead of hopping from node to node and hoping no intermediate link will fail, etc. We could enjoy all this if we would use the same communication protocols as on the Internet, TCP/IP that is. The reason why we're not doing this at this moment is due to the proprietary command sets of the various TNC-types. The only standard that has ever been adhered to for TNCs is the KISS-protocol, which moves the Packet-Radio 'intelligence' from the TNC back to the PC. This forces us to write device drivers for our favorite operating systems (except for Linux, none of them know AX.25 nor KISS). As you might have guessed, this is not a simple job to do. It has been done on Windows '95 a few times, but their main problems are stability. One shouldn't expect such drivers to be developed for OS/2, Mac, etc. That's why we tried to make the TNC compatible with the OS instead of the other way around.

Dialing on to your TNC

These thoughts were the base for the TNC-project we started in Sint-Truiden years ago. The installation of our TNC and the connection to our Packet network should be comparably easy as for a regular telephone modem and setting up an Internet connection, respectively.

We programmed the TNC to act as an ordinary telephone modem (using the standard Hayes AT command set) towards the PC (and the Operating System). We added a few commands that are typically needed on Packet Radio networks: call sign, TX-delay, persistence, etc (which can easily be entered once as the modem-init string). Finally, we dial to our Internet Service Provider using a call sign instead of a telephone number.

Once the computer thinks it's connected to the ISP (as the TNC confirmed this), both switch to the well-known SLIP-protocol. Our TNC will replace the SLIP-specifics on all data originating from the PC by Packet Radio (AX.25) specifics and will then put these packets on the air. Captured packets will be filtered against the user's call sign and translated to SLIP-frames. So the TNC actually acts like the ISP itself, but fully transparent for the Operating System and the user. This enables Radio-Amateurs to use services like E-mail, file transfer, Netnews, WWW, IRC, etc with the very same software they're using on the Internet. The services themselves are provided by servers like on0baf.baf.be.ampr.org (44.144.181.1), just as on the Internet. The 'on the air' communication is fully compatible to the TCP/IP over AX25 standard, set up by KA9Q, so WNOS, TNOS, JNOS or WAMPES stations can be accessed with this setup based on a MCB152 and a standard Internet ready installed PC.

The full description for setting up such a server based on a Linux platform as also a detailed explanation of the TCP/IP coordination in Belgium can be found at www.qsl.net/on1blu, on0baf Internet website.

We expect this will cause the development of new equipment, which will increase our transmission speeds. These developments could reverse the decreasing interest to use or experiment with the Packet Radio network. Information on new developments can also be made available more easily. Using the well-known standard TCP/IP should also avoid the long times to develop new services or new techniques from scratch.
TNC = μC with communication software

Generally spoken, a TNC is a micro controller with TNC software (the Packet Radio protocol, for example) included on its EPROM and equipped with special hardware to control a transceiver. Typical TNCs can not be used for anything but Packet Radio (or other digital protocols for that matter).

To allow the usage of our board for almost any project one could imagine, our EPROM only includes the firmware necessary for being able to receive the ultimate software from a PC (so the firmware is comparable in function to the BIOS in a PC). Additionally, the firmware allows for debugging, disassembling or starting the software. Software is downloaded into our MCB-152 using the well known Intel-Hex format (text-files).

The transceiver-control hardware is on a separate board, which is plugged onto the MCB-152 (compare the MCB-152 with a motherboard in a PC, where additional boards can be plugged in). Currently, the MCB-152 is delivered with a plug-in board that cross-connects to a USCC-type modem - available from the BayCom group. The DK9RR modem, for example, allows for 1200bd AFSK, 4800bd and 9600bd G3RUH operation. The DF9IC modem can be used for 76K8bd or up to 615Kbd. Depending on the software you downloaded into the MCB-152, you can use your old-style Packet Radio software (with MCB-152 running as a KISS-TNC) or the newer fancy Internet software (SLIP-TNC).

The board is called MCB-152 since it is a micro controller board based on Intel's 80c152jb universal communication controller (an 8051 with internal communication facilities, capable of doing 680kb/sec with the current clock frequency - extendable to 2Mb/sec). We equipped the MCB-152 with the maximum amount of memory for an MCS51-processor (which is 128kB: 64kB data and 64kB code). All unused I/O connections are guided through extension connectors for future developments.

The firmware in the EPROM contains a library of 50+ routines, which are copied to RAM during power up. This enables novices to make an easy start on programming the micro controller. Furthermore, this copy operation is completed by turning off the EPROM, making all memory addresses available for use, unlike many other systems that hide part of their RAM behind their ROM.

MCB152, a wealth on information

We included all schematics, layouts, library references, assembly and installation guides, etc. on a CD-ROM. It also includes a complete development environment (including BASIC-interpreters, C-compilers, assemblers, debuggers, etc.), the source code of all projects we've already completed, information and utilities available from the Internet, application notes and user guides for the MCS51 series of micro-controllers.

A video file (software\intro.avi) shows the history of our project and can be used to introduce others. One can also order an assembled, functional and tested MCB-152 for an attractive price.

<table>
<thead>
<tr>
<th>Product Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCB152+MODEM+CD</td>
<td>248,00 EURO</td>
</tr>
<tr>
<td>MCB152+CD</td>
<td>163,00 EURO</td>
</tr>
<tr>
<td>CD</td>
<td>19,00 EURO</td>
</tr>
</tbody>
</table>

Account Citibank 953-0317011-81. Include your name, address and wanted articles. Delivery paid by customer.