TCP/IP and radio amateurism
A UBA−RST TCP/IP TaskForce project

Ever since Packet Radio saw its daylight during the eighties − with speeds up to 300 and 1200 baud's − it appears not to have evolved to better operation. Sure, there are many initiatives, 9600 baud G3RUH being the most persistent, but none of them is sufficiently wide spread. It's almost as if Packet Radio is the most noticeable sign of what's currently happening to radio amateurism in general (in Belgium at least): a fade out. Many discussions have been held on the causes, Internet and GSM being the most frequently referenced. In my opinion, there are 2 causes to this:

- Even though we have a wide spanning Packet Radio network (and the Packet Radio network currently is the only way to connect ALL HAMs on a 'PERMANENT' basis), the pace of its evolution now results in an arcane technology that's hard to work with. This has a negative effect on sharing technical knowledge and experience, and therefor prohibits large based experiments.
- Since radio amateurism hasn't evolved as the Internet did, for example, there's no need to experiment with new technologies. Transceivers are already very small, further improving them seems obsolete: such improvements won't let us talk faster :−), nor let us type faster (while working with our interactive BBS systems, to name just one).

The UBA−RST team concluded this particular Packet Radio domain should undergo a major evolutionary step (or should we call it a revolution, looking to the time span we had for realizing this?). From the very beginning, we had two goals in mind:

- HAM−operators, which are not interested in computers or Packet Radio itself − but want to use these TOOLs to exchange their experiences, should be able to do this in a very simple way (without having to learn various software packages).
- HAMs that feel to perform technical experiments, should be provided a platform where they can learn things that are relevant to them. Let me illustrate: the Packet Radio protocol − AX.25 − is specifically designed for Amateur Radio. If technicians are forced to develop directly upon this AX.25 platform, they can't use their experience anywhere else. Ultimately, this means they must start producing their own goals, often not providing any contribution to radio amateurism in general.

So, now we come to the point why we choose for TCP/IP specifically:

- TCP/IP is the protocol that's currently being used on the Internet. If we superpose this protocol set on top of the bare AX.25 link protocol, we enable HAMs to use their favorite and accustomed Internet software (like NetScape Communicator, Microsoft Internet Explorer, Microsoft Outlook, etc). This would greatly simplify the usage of Packet Radio: not only because of the related software that is user−friendly itself, but also because it eliminates the need for learning additional software packages (you simply use the ones you already know).
- TCP/IP is not only used on the Internet: even the smallest companies are switching to TCP/IP for use on their local networks (LAN). This means TCP/IP technology, TCP/IP knowledge and TCP/IP experience is as relevant as you can get! If you learned something on the AX.25 + TCP/IP network, you can apply this knowledge equally well in your professional life.

Especially the latter is very important, since it revives the "raison d'être" of radio amateurism. If one wants to perform real−life experiments on server technologies, he can only do this on the Amateur Radio network! Internet providers, companies and even universities WILL NOT allow you to experiment on their network (you could cause major damage − financially, for example − when you suffocate their network and − as a consequence − potentially parts of the Internet as well). So, you could create an isolated network, but you can hardly call that a real−life test, can you?

What the project is all about

While we noticed many people were already experimenting with TCP/IP and high−speed Packet Radio, we never encountered a full−covering approach. So, next to some 'pure development' efforts (like the MCB−152), we generally collected many bits and pieces and tried to construct guidelines for SysOps to boost new life into their nodes.

This was a step by step process as we'll illustrate shortly, but on realizing each individual step we considered one aspect thoroughly: LEGACY. None of our realizations would ever be accepted if they ruled out a group of users. So we must be compatible with existing Packet Radio systems (as far as modulation and coding is concerned), allow Packet Radio users to pass the system, allow mails and bulletins to be exchanged between TCP/IP users and BBS users, etc.

For the remainder of this text, we'll talk about AX.25 users and TCP/IP users. Of course, TCP/IP users also run AX.25, but they use the AX.25 protocol **implicitly** (they typically don't even realize they're using AX.25).

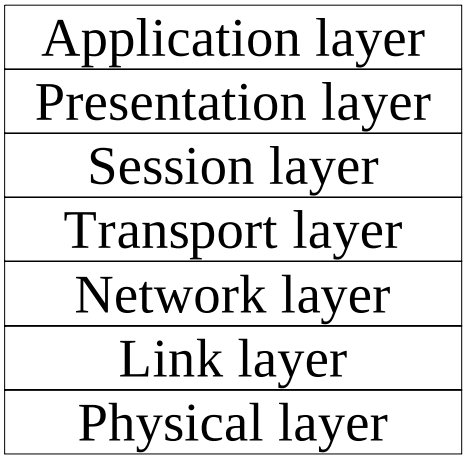

Illustration 1The OSI reference model

Generally, Packet Radio was never meant to be the ultimate communication platform itself. If you take a look at the Digital Communications section of the well−known ARRL handbook (even for an eighties−edition), you'll notice the OSI layered communication model (Illustration 1). While TCP/IP consists of lesser layers, we'll use the OSI layers bottom−up to tell our story step−by−step or layer−by−layer.

Raw Communication

Let's start off with the physical layer. The physical layer is the grouping of communication media and the collection of equipment that's necessary for modulating and demodulating raw data on a specific medium. For sound, the medium is the propagating vibration of molecules in a liquid or gas environment and human equipment are the vocal chords and ears. We, radio amateurs, are interested in the electromagnetic medium and the accompanying transceiver equipment, of course.

By adding an additional TCP/IP load to AX.25, we hope to create an essential need to improve digital transmission performance: a need for better transceivers and modems.

Packet Radio

At the link layer, we find our AX.25 protocol.

The link layer is actually a collection of software that provides courtesy protocols (keep silent while others are talking, for example).

For coding/decoding digital (AX.25) data, we developed our MCB−152. These were our design goals:

- it should support high speed communication evolution, without needing to buy entirely new equipment,
- for initial appreciation it must be compatible with current TNCs (1200 and 9k6 G3RUH−compatible) and current software (KISS firmware), and
- it should require minimal configuration effort.

The first two goals are realized by separating the controller hardware and the modem itself through a connection header that's pin−compatible with Baycom's USCC−type modems and by

developing KISS firmware.

The third goal was realized by developing our famous SLIP firmware. After firmware download, the MCB−152 will act as a typical telephony modem (interpreting the highly standardized AT Hayes command set). The user dials as ’telephone number’ the call−sign of his local IP router and as soon as the dial is complete, the MCB−152 will turn into a SLIP−to−AX.25 translator (and vice versa). So, the user only needs to setup an additional Internet account, using one of the generic modem drivers that are provided with Operating Systems like Windows ’95/’98/NT, etc. The MCB−152 also auto−detects the baud rate used on its serial port.

One last word on our MCB−152: we decided not to limit the MCB−152 for TNC usage. Instead, we developed a complete development library for programming the Intel 80c152 chip and collected all available development environments and documentation on a single CD. This enables the MCB−152 to be used for education, satellite tracking, you name it!

TCP/IP

It’s only at the network layer that TCP/IP appears for the first time. You read this right: without AX.25, TCP/IP simply can not operate on our radio frequencies.

There’s currently no development required here: the TCP/IP protocol set is available on nearly all computer architectures and operating systems (and allows for interoperation between them).

For completeness:

- The network layer is the collection of software that handles routing. It’s the network layer software that reduces our personal "network knowledge" to identifying our local IP router and identifying our destinations (no longer bothering about the local node of our destination and possibly several intermediate routes). The network layer software should also redirect data appropriately (and transparently!) when intermediate nodes fail.
  When using the TCP/IP protocol set, IP (the Internet Protocol) is the routing mechanism used. It’s a hop−to−hop routing protocol, meaning that routes are not calculated by the IP software on your machine. Instead, each node (including your machine) decides − for each individual packet − what neighboring node the packet will be forwarded to.
- The transport layer is responsible for transporting information. Until now, we only talked about packets of data; transport layer software will maintain an information stream on top of an unreliable network. Two kinds of transports are currently common: simple, unacknowledged datagrams (UDP protocol) and acknowledged in−order no−duplicate information transfer (TCP protocol).

In Belgium, we had to do some preparation, however. IP addresses were distributed at a too coarse level (provinces), and those provinces were assigned decimal coded identifiers. For as soon as a second TCP/IP server appeared in a single province, problems arose. Within a province, IP addresses were distributed sequentially. When several servers appeared, users typically operated through their nearest TCP/IP server. However, the IP address distribution didn’t allow for any reasonable separation between the users of different servers. Result: within a province, explicit host routes had to be created on all servers within that province, for all users in that province.

Our TCP/IP TaskForce submitted a proposal for introducing TCP/IP regions (corresponding to IP router availability). This allows for very simple network route entries. These regions are authoritative now, meaning that each region coordinator can distribute IP addresses freely within his assigned range of IP addresses. For now, region coordinators forward their assignments to the national IP coordinator, such that the latter can synchronize towards the worldwide ampr.org file. In a near future, we hope NameServer entries may be added to this worldwide ampr.org file, thereby fully enabling the authority of the regions. This would enable us to experiment with technologies such as DHCP/DDNS.

Sessions and presentation

Currently, the TCP/IP layer model doesn't provide explicit session (some applications use cookies to simulate session management) and presentation layers (some applications implement MIME extensions), so we'll skip it for now.

Applications and services

So, we finally reached the application layer. The application layer is the collection of all protocol API libraries (HTTP, FTP, SMTP, etc.) and server software.

User applications (like Netscape communicator, Microsoft Chat, etc.) use these API libraries for information exchange across the network. What this all means is that a programmer doesn't need to program a script engine for automating routes to distant BBSs, downloading and uploading mails and bulletins, or any other network issue for each application he wishes to create. He can now concentrate on one thing: providing a user−friendly interface.

These applications, however, communicate with service providers (Microsoft Outlook, for example, communicates with an SMTP server and a POP server). This brings us to our final effort: setting up TCP/IP servers (which is far more complex matter, but the user isn't exposed to that).

We currently have 2 options:

- a Windws NT based server and
- a UNIX based server (like Linux)

The first isn't quite an alternative, especially in the NT4 era. NT4 is a desktop operating system with many networking tools. Next to the fact that NT servers are generally too expensive for amateur radio servers, many services need to be bought separately as well (all of them being equally expensive). Furthermore, a driver should be developed to let NT communicate with our TNCs, somewhat like SV2AGW and FlexNet32 do for Windows '95/'98 (SLIP can't be used on servers, since that's a point−to−point protocol for dedicated links).

Windows '95/'98 is not feasible for providing services. Windows 2000, however, is a far better environment, since it really is a network operating system and it finally complies to standards. Most Internet services are included now too, but it's still expensive and requires driver development.

We choose for Linux since

- it's free,
- it supports AX.25 in its kernel NOW and
- it comes with all − currently well−known − services, all for free as well.

The Linux server

Our project specifically collected configuration information for ALL those services. We succeeded in providing our users with a wealth of services:

- E−mail service, through the sendmail SMTP−server and the IMAP/POP3 server from the University of Washington
- News bulletin service, through the INND NNTP−server
- Web browser service through the Apache web (HTTP) server. We also allow our users to put their personal home pages on our webserver

- FTP file service, NFS file service, SMB file service (c.f. Network neighborhood in Windows operating systems), etc.
- Remote login (telnet) service
- Chat/conference service through Undernet IRC server
- Domain name resolving through the BIND DNS server

Most of the server software mentioned is also very popular on the Internet for use by ISPs, universities, etc. Each of these services will facilitate information exchange enormously as schematics and source code can be mailed using drag&drop, project status can be published through web pages, etc.

Remember our permanent concern for legacy:

- We allow AX.25 users to participate in the IRC network.
- We also provide an Internet−type DX−cluster service that can be connected from both AX.25 and TCP/IP users (and which links up to the existing DX−cluster network).
- And, extremely important, we provide an SMTP/NNTP <−> BBS gateway. Currently, we run a JNOS−box on our Linux server. This JNOS−box's TCP/IP stack is connected to the native Linux TCP/IP stack through a virtual SLIP link, allowing JNOS's SMTP and NNTP servers to exchange data with Linux's SMTP and NNTP servers. For forwarding, JNOS taps Linux's AX.25 interface. Soon, we will start using the mailgw and lnxforward packages to do this. Mailgw is actually an MTA (Mail Transport Agent) and we will configure sendmail and INND to exchange BBS−destined or BBS−source messages with an FBB−compatible system. Lnxforward is an FBB−compatible forwarding−only module (also providing a web interface for reading mails and bulletins). This allows for transparent exchange of mails and bulletins between BBS and TCP/IP users.

Instead of using the lnxforward package, we could also install yet another full−fledged FBB system on our server, but we prefer not to. By installing another classical BBS system, you don't send a consistent message that users would benefit from switching to TCP/IP. Instead, we closed a deal with a neighboring BBS SysOp by which we have a forward feed to our gateway and our local AX.25 users have a home BBS through our local node step up.

These efforts resulted in seminars and a step−by−step guide for setting up a brand new server.

Future vision

By introducing TCP/IP on our amateur frequencies, we brought radio amateurism back up−to−speed with industrial development/environment. As we noticed that major SQL (database) server builders like Sybase and Oracle make their latest versions freely available on Linux for development use and as we noticed free application servers − supporting Java and XML, which only reached hype stage in industry so far − appear for Linux, we are convinced radio amateurism can reconquer its pioneering role that characterized radio amateurism until a few decades ago.

To continue the illustration: for demonstrating TCP/IP at 76kbps, we issued voice−over−IP, which is still cutting−edge in industry (not that we have actual plans for applying this technology, but we did it!). We also noticed the appearance of APRS, even before all telecom operators implemented their GPRS. Combined with TCP/IP, this should further boost amateur radio spirit.

Finally, since network traffic performance becomes more and more important, we planned brand new experiments that could deliver an alternative for the current − expensive − region interlinking, allowing for significant higher throughput.

References and authors

Members of the UBA−RST TCP/IP TaskForce:

- Gert Leunen (ON1BLU)
  TCP/IP server setup
  Belgian IP/DNS coordinator − Network 44.144.0.0 (0xffff0000) − Zone be.ampr.org
  AMPR−mail: on1blu@on0baf.baf.be.ampr.org
  Internet E−mail: on1blu@qsl.net
- Joachim Elen (ON1DDS)
  Firmware development
  AMPR−mail: on1dds@on0baf.baf.be.ampr.org
  Internet E−mail: Joachim.Elen@village.uunet.be
- Walter Machiels (ON4AWM)
  Hardware development
  AMPR−mail: on4awm@on0baf.baf.be.ampr.org
  Internet E−mail: on4awm@wol.be

Project web pages:

- IP coordination & Linux server setup: http://www.qsl.net/on1blu
- Hardware developments: http://home.worldonline.be/~vda10786
- Firmware developments: http://gallery.uunet.be/Joachim.Elen
- Product distribution: http://www.caseconsole.com/mcb152