news@ftp

The FTP Software® Newsletter

Volume III, Number 1 January 1990

Inside…

Item	Page
New Product News	1, 2
Masthead	2
Technical Support	3, 6, 7
Internetworking News	8

New Literature Available

Recently we have released three new pieces of literature for our PC/TCP product line.

1. Our six-page PC/TCP Plus glossy has been revised to include the new 2.04 features.

2. The Compatibility Sheet has been revised and updated. The new version describes PC/TCP Plus' level of compatibility with NetWare™, VINES®, LAN Manager and other network operating systems. It also talks about support for the four standard driver interfaces (Packet Driver, NDIS, ODI and ASI), as well as compatibility with Microsoft Windows.

3. We have a new Third Party Software sheet with the names and addresses of companies whose software runs on top of the PC/TCP kernel. It lists manufacturers of emulators for a variety of DEC®, HP®, Tektronix, IBM and other terminals, as well as companies who make X-Windows and distributed database products for PC/TCP, plus memory extenders that work with PC/TCP Plus.

Give us a call and we'll send you a copy of each.

Version 1.16 of PC/TCP® To Be Retired

This coming April we will be halting shipments of PC/TCP Version 1.16, the original software produced by FTP Software. PC/TCP Version 1.16 has evolved from the original PC/IP code developed at MIT by John Romkey, Dave Bridgham and others. FTP Software was founded to enhance this code, and out of this effort came Version 1.0 of PC/TCP.

Over time it became clear that several important structural changes were necessary for the long-term success of PC/TCP. We needed to add a Berkeley Sockets emulation, NetBIOS and NFS. To support these programs, we developed a Terminate and Stay Resident (TSR) kernel, which became the core of PC/TCP 2.0. Since Version 2.0 was introduced in 1987, most PC/TCP users have converted to it. The work on NetBIOS and InterDrive™, our implementation of NFS, couldn't have been completed without the transition to Version 2.0.

Because of our commitment to supporting our software, we have continued to maintain Version 1.16 for the past two years to provide a gradual transition for our customers. Now that most people have converted to Version 2.04, we feel that it is in our customers' best interests for us to focus on PC/TCP Version 2.04 so that they can get the best possible support from us.

news@ftp
Volume III, Number 1 ¥ January 1990

Published by FTP Software, Inc.
Editor: Nancy Connor.
Communications pertaining to this newsletter should be directed to:

news@ftp
FTP Software, Inc.
26 Princess Street
Wakefield, MA 01880-3004
Phone: (617) 246-0900
FAX: (617) 246-0901
Electronic mail: news@ftp.com

FTP Software welcomes suggestions and contributions from readers. Submissions for the next issue of the newsletter must be received no later than February 30, 1990.

© 1988, 1989, 1990 by FTP Software, Inc. Permission to use, copy, modify, and distribute this publication for any purpose and without fee is hereby granted, provided that this copyright and permission notice appear on all copies, the name of FTP Software, Inc. not be used in advertising or publicity pertaining to distribution of the material without specific prior permission, and notice be given that copying and distribution is by permission of FTP Software, Inc.

PC/TCP, LANWatch and FTP Software are registered trademarks, and InterDrive is a trademark of FTP Software, Inc.
VINES is a registered trademark of Banyan Systems, Inc.
DEC and VT are registered trademarks of Digital Equipment Corporation.
HP is a registered trademark of Hewlett-Packard Corp.
IBM and IBM Personal Computer are registered trademarks, and Micro Channel is a trademark of International Business Machines Corporation.
NetWare is a trademark of Novell, Inc.
ProNET-10 is a registered trademark of Proteon, Inc.
NFS is a trademark of Sun Microsystems, Inc.
3Com, 3+ and 3+Open are trademarks of 3Com Corporation.
Ethernet is a trademark of Xerox Corporation.

LANWatch® 2.0 Released

by James VanBokkelen

Since last spring, when we released version 1.21 of LANWatch, we ve been working on version 2.0. It s in beta test right now, and everything looks good for a release in mid January.

The most visible difference between versions 1.21 and 2.0 is the restructuring of the in-memory packet buffers. We ve eliminated the limit of 254 buffers and the array that required it. Version 2.0 allows up to 4096 packets in memory, and they re kept in a circular linked list, allowing us to do a few tricks that weren t possible before, like incremental capture and trigger filters.

A less obvious difference arises from a change in the memory model; version 1.21 (and its predecessors) were small-model programs, and there wasn t enough room in the code segment for all the filters and packet parsers we had, let alone those we wanted to add. Version 2.0 uses medium memory model (large code, small data), and as a result, it understands many more protocols out of the box.

The number of packets is now limited by available DOS memory. If you use the default capture length of 100 bytes (enough for almost all protocol headers, and usually some data), we usually wind up allocating about 3000 packets on a 640K machine. If you need to capture all 2042 bytes of data on an 802.5 token ring, the same machine has enough room for about 200 packets. If you only need to see the first 70 bytes of each packet, it can store about 4000 of them. If your Personal Computer® has less memory, we handle it gracefully.

Since we use all the available memory, DOS features that depend on the transient part of *command.com* remaining intact (like the F3 key) won t work unless you use the -m *xxx* switch when you start LANWatch to request that it not use all of DOS memory. Use of the shell escape (!) command requires this switch also, to leave some memory for the command processor.

We ve added several Examine-mode commands to make three or four thousand packets

manageable: The most important of these are forward and reverse search commands that use standard packet filters to find specific events in the data stream. Ctrl-U is a universal command multiplier, executing the next command four times, to move your viewpoint faster.

To make it easier to compare sequences of events, we ve added incremental capture mode, where we try not to overwrite previous captures with new packets for as long as possible. If you don t want this, the new R command clears all buffers and stops LANWatch when all of its packet buffers have been refilled.

Besides the search filters, there have been several other improvements in the packet filtering. We ve added one level of AND or OR to the top level filters. The new alarm filter rings the bell when a packet matches, and the new trigger filter lets you leave an unattended LANWatch running, for days if necessary, to capture a hard-to-reproduce event.

There are several new network management functions: New on-line commands activate a running display of network throughput in bytes and packets per second. Throughput output, per-protocol statistics, hardware statistics and packet length statistics can be written to a file as well as displayed on the screen. A new off-line traffic analyzer reports usage by node and protocol, while **tana** has been enhanced to detect more errors and provide better state information.

For those of you who are interested in adding your own parsers, filters or off-line analyzers to LANWatch, we have greatly improved the manual in these areas. Commenting in the source files themselves has been improved, and there is a whole new file of help routines to simplify using LANWatch data structures and code in off-line analyzers.

We re already working with many of these new features as we test in-house, and we find them quite useful. Since much of the new development reflects comments and suggestions from users of previous versions, we expect that our customers will like LANWatch 2.0 as well, and we re hoping for a good response.

NOTE: Some of the new parsers and filters were contributed by our customers; If you have done any development you feel would be of general interest, we d be happy to consider it for inclusion as a fully-supported part of this and future releases. Copyrighted code is fine as long as FTP Software and our paying customers may legally use it, and depending on your wishes, we can either give you or your organization credit in the source code, or protect your privacy.

Keyboard Remapping Revealed!

by Vicka Corey

It happens often in foreign lands. It can happen eventually to anybody. That plain old US-English ASCII table suddenly isn t enough, and it s time for a user to remap the keyboard.

You can use keyboard remapping to gain easy access to special scientific or mathematical characters, to change the placement of hard-to-reach keys such as Alt or Insert, or to customize to foreign alphabets. You can remap one key, switch two keys, or rearrange everything on your keyboard. Keyboard remapping provides access to a range of characters much greater than is apparent in the default configuration.

If you use the **tnvt** or **rloginvt** programs, you can remap both the VT®220 emulation (the default) and VT100. You can also remap the keyboard of the IBM® 3278 emulator used by the **tn3270** program. Note that if you are using a third party emulator through **tnglass**, that emulator may be remappable through a mechanism of its own. See the emulator s documentation for help.

All of the remapping information is stored in an ordinary text file located on your PC or on an NFS-mounted drive. You tell PC/TCP where this information is located by setting an environment variable. For example, the following command lets PC/TCP know that remapping information for the VT220 emulator is to be found in the file *vt220key* in the *etc* directory on the C: drive.

```
C:> set VT220_KEYMAP=c:\etc\vt220key
```

Similar commands are used to set the VT100 or 3270 remapping files. Note that the VT100_KEYMAP variable only applies to PC/TCP versions 1.16 through 2.03 the VT100 emulation disappeared as a separate Telnet executable in version 2.03 and from Rlogin in version 2.04. It is now considered a

subset of the VT220 emulator, and therefore uses the VT220_KEYMAP variable. These variables can be set from within *autoexec.bat* or other batch files for additional convenience.

Now, what goes into this mysterious file? In essence, the program needs to know what keys are to be remapped and what value each of these keys is to send. Each case of each key is handled as a separate item the remapping of lowercase g is not going to affect the handling of Ctrl-Shift-Alt g . So, each individual remapping entry contains a code to specify case, a code to identify which key is pressed, and a remapping value which tells the emulator what to send in place of the default.

Each line of the file defines a single remapping. The first code on each line refers to the case of the key in question, as described on the bottom of page 3-3 of the 2.04 *User s Guide*. This value lets PC/TCP know in which of its internal tables to find the value for the key in question. The table on page 3-3 shows all the values, from 0 for a plain character to 7 for a Ctrl-Alt-Shifted character.

The second code is a unique scan code generated by the key itself. Diagrams showing unique scan codes for different keys on a variety of keyboards are to be found in *User s Guide* Appendix D. Scan codes can also be determined by using the program **sc**, which comes included in your PC/TCP distribution. (Though it is not installed by the standard installation program, it is located on the last disk as the file *sc.exe.*) Type `sc` and then press a key to generate the equivalent scan code associated with the key being pressed, then with its release. For example:

```
C:> sc
Type ESC to exit
0x0a dn 0x0a
0x8a up 0x0a
```

This output was generated when the 9 key was pressed after starting the scan code program. Pressing the Escape key exits the program; pressing any other key generates the relevant scan codes. In the keyboard remap file, you would use only one of the codes generated the one that is unique to the key pressed. Generally, **sc** displays this unique value last. Referring to the tables in Appendix D, the scan code shown for the 9 key is 0a .

The third code is the most important one it is the replacement value. This is called the remapping table value. For characters and codes representable with standard ASCII values below 128, this third code will be the appropriate ASCII number, expressed either in decimal or in hex with the 0x prefix.

The VT220 terminal emulator also supports the DEC Multinational Character set, which contains many unusual and interesting characters. These are of great use when supporting non-English languages. These values can be found in the DEC VT220 terminal reference put out by DEC, and reprinted here. The 3270 emulator supports the IBM Extended ASCII codes for the remapping table values.

Many special terminal control keys can also be remapped. The remapping values for these keys can be found in Chapter 3, entitled Terminal Emulation , in our *User s Guide.* For example, the remapping value for the VT220 PF1 (Gold) Key is given on page 3-10, and is 0x150. (That s hex; you could convert it to decimal and substitute that value if you preferred.)

Let s step through an example remapping: Let s suppose that you wanted to remap the Alt-9 key (normally not very useful) to the VT220 Gold key (quite handy in certain VMS editors). First, you set the VT220_KEYMAP environment variable to point to a file:

```
C:> set VT220_KEYMAP=c:\etc\9togold
```

Then, using a plain text editor like **edlin**, **vi**, or **emacs**, you create a file named *9togold* on the C: drive in the directory *\etc*, and put in it the line:

```
4, 0x0a, 0x150
```

Then, you connect to a host using **tnvt** or **rloginvt** with VT220 emulation, and whenever you press Alt-9, the foreign host will respond as though you had pressed a real VT220 s PF1 key. You can check this by pressing another key which normally defaults to VT220 PF1, the PC F1 key.

This is a very simple and pedestrian example of what keyboard remapping can do, but given the level of difficulty of remapping the keyboard, it very well should be. Once all the codes and the methods of construction are understood, keyboard remapping becomes a

straightforward and extremely useful way to customize a PC keyboard for more powerful terminal emulation. For a great deal more information and detail, you can refer to the invaluable Terminal Emulation chapter of the *User s Guide* and the section on terminal emulation in the Command Reference, or call us here in Tech Support.

One last hint as far as the foreign host is concerned, a PC running a terminal emulation is just the same as a real terminal of the type that the PC is claiming to represent. In the words of James VanBokkelen, Pretend it really is a DEC VT220 (or VT100, or Big Blue 3278 Model 2) and you will be rewarded .

Notes

A few of the VT220 remapping codes were incorrectly given in version 2.04 of the *User s Guide*. The correct values, which should have appeared on page 3-11, are:

VT220 Key	Remap Value
Keypad ,	0x71d
Keypad Enter	0x71c
Keypad -	0x71e
Keypad .	0x71b

On another matter, it often causes some confusion that some terminal keys do not have associated remapping values, and that some extended PC keys are not shown as having useful scan codes in Appendix D. The reasons are simple: these terminal keys have hardware functions in the original, and the PC keys send scan code sequences that do not have convenient unique values to be remapped. For example, in remapping one of the extended AT keys, you might find you had accidentally remapped your Return key as well!

Copyright © 1983 by Digital Equipment Corporation.
All rights reserved.

Token Ring Networks: An Update

By Benjamin M. Levy

IBM has released a 16 Megabit Token ring board, and the number of manufacturers making Token Ring boards has increased.

IBM now sells 16 Megabit Token Ring boards for both Micro Channel™ Adapters and the standard PC Bus computers. With Texas Instruments release of their 16 Megabit Token Ring chipset (TMS380C16) several manufacturers (Proteon, and Ungermann-Bass among others) will be releasing 16 Megabit Token Ring boards soon. A 16 Megabit Token Ring board can run at either 4 or 16 Megabits per second. And it can be used with any existing software that uses the ASI to talk to the Token Ring board. A bridge can be used to connect a 4 Megabit Token Ring to a 16 Megabit Token Ring.

The list of Token Ring Board manufactures for the PC now includes CNet Technology, Compex, DSC Communications, Gateway Communications, Hughes LAN Systems, IBM, Inmac, Lanmaster, Lantana Technology, Madge Networks, NCR, Plexcom, Proteon, Pure Data, Racore Computer Products, Tiara Computer Systems, Thomas Conrad, Ungermann-Bass, Western Digital and 3Com™.

Wiring

IBM Token Ring boards have a DB-9 (a nine pin, D shaped connector) and use IBM Cabling System data grade media (Types 1, 2 & 6, shielded twisted pair wire) to connect to the MAU (and to connect MAUs to other MAUs). 4 Megabit IBM Token Ring boards can use RJ-11 (a four or six pin telephone type modular plug) for use with Type 3 Media, instead of the standard data connector. Some of the other Token Ring manufacturers offer both DB-9 and RJ-11 connectors and some use shielded and unshielded twisted pair cables to connect to the MAU. Also some manufacturers offer boards that can use coaxial cables.

If you are using RJ-11 modular connectors please make sure that you are connecting your Token Ring board to the Token Ring and not your telephone system.

Note that the Token Ring board will tell you if the cable is not attached to the board, but it won t tell you if the cable is not attached to the MAU. If you are having a problem with a Token Ring board that is neither sending nor receiving packets, tmake sure the far end of the cable is attached to the MAU. If it takes more than the usual ten to fifteen seconds to enter the Token Ring, then the board might be thinking that its the only board in the ring.

The MAU can be a simple device. The basic MAU will have several ports for Token Ring boards to attach them selves to the ring. It has two extra ports to attach the MAU to other MAUs to form an even larger Token Ring. When a Token Ring board enters the ring, it applies power to the MAU which will cause a relay to close which adds the board to the ring. When this happens there is usually faint click, which you can hear if you are standing near the MAU.

There are various types of MAUs. Some come with ports for two, four, eight, or sixteen Token Ring boards. Most MAUs are passive, permitting the Token Ring boards to do the work of entering and leaving Ring. Some MAUs can actively monitor the status of the Token Ring and report it to a remote computer. The remote computer can tell the MAU to remove itself from the ring if it sees a problem.

Questions

If you are going to buy a board from a dealer here are some questions that you should ask.

Does the board come with diagnostic software? This is very useful for testing to make sure that it is working and properly installed.

Does the board come with a driver equivalent to tokreui or a driver that can be used with the LAN Support Program? In order for the board to be used with software that expects an ASI, you will either need the LAN Support Program or *tokreui* (or the equivalent).

Does the board come with any other software? There are some software applications that might require a particular driver, such as Banyan VINES, Novell NetWare, and 3Com 3+™, though most just require an ASI. If you have a specific network application in mind, you might want to find out if the application comes with network drivers.

Does the board come with a cable to attach it to the MAU? If not, you'll also need to buy a cable.

How many ports does the MAU have? If you are planning to have lots of computers on the Token Ring you may need several MAUs.

Does the MAU require electrical power? This might affect where you locate the MAU.

How many Token Ring boards can their network support? If the number is too low, you might want to get a Token Ring bridge to subdivide the network.

What is the maximum length for a Token Ring cable? This will affect where you can place your MAU and computers.

Can the Token Ring board use unshielded twisted-pair wires? If so, you might be able to use existing, unused telephone wiring.

NetWare Compatibility

by James VanBokkelen

There are three ways you can make NetWare and TCP/IP usable from the same PC while using only one network interface card. Each has advantages and disadvantages; I sell one, but I ll list them in order of introduction:

1. You can modify the IPX module so that it is possible to share the hardware interface, and use a TCP/IP protocol stack side-by-side with NetWare. BICC Data networks did this first, initially with PC-IP and later with our PC/TCP product, but they didn t make their interface sharing specification public. Proteon did the same thing, but with a different, P1300-specific interface and our PC/TCP. We didn t want to write drivers for a dozen private interfaces, so John Romkey came up with the Packet Driver specification (which isn t specific to NetWare, but has been most widely used with it) and published it. Later, Excelan did the same thing using a private interface and their Ethernet board s on-board TCP/IP.

You can build your Packet Driver into *ipx.com* (as Acer, D-Link, Gateway, IMC Networks, Interlan, Schneider & Koch, Sytek, Univation and others did), in which case you can stick with the normal, illegal, NetWare encapsulation. Alternatively, you can require that your users run **econfig**, to make NetWare use a legal Bluebook packet, and build an *ipx.com* which uses a separate Packet Driver to do its network I/O (Kelly McDonald s freeware *shelldrv.obj* from BYU does it this way).

Advantages: It is faster, and there is no load on the server because the IP traffic goes direct to the other IP host. If the interface is Packet Driver and not proprietary, you can use either commercial or freeware TCP/IP stacks, as well as the freeware Packet Drivers from Clarkson. The exact TCP/IP features you get with this approach vary, depending on which TCP/IP package you use (PC/TCP, NCSA, PC-IP, WIN/PC or KA9Q at the moment), but in general, this choice gives you the widest variety of applications (3270 Telnet, Rcp, TFTP, etc.) to choose from.

Disadvantages: The LAN you re connected to has to be one on which TCP/IP is widely used (Ethernet, Starlan, ProNET-10®, 802.5). The PC has to have an IP address, and the software has to be installed on it.

Note that you get the same effect as if a Packet Driver was in use on 802.5 interfaces which support IBM s ASI software driver specification (*tokreui* or LAN Support Program). It is different in structure, but it serves the same purpose, of letting many protocol stacks use the same card at the same time. PC/TCP and IBM s DOS TCP/IP both support 802.5 via this interface.

You should also note that Banyan VINES has supported this scheme of interface sharing on Ethernet since version 2.10, with a private interface. 3Com 3+Open™ supports the same scheme via the published NDIS interface.

2. You can get the Interlan gateway product, and install it in your NetWare server. This is an NP600 intelligent Ethernet card and software which acts as a high-level translating gateway between IPX protocols on one side and TCP/IP on the other. It supports Telnet and FTP from the workstation, and incoming FTP to the server. The Telnet user interface is via a user-supplied terminal emulation program on the PC. I am told that it supports 16 connections via a single server.

Advantages: The PC can use any NetWare-supported media (Ethernet, Arcnet, Omninet, etc.) without consideration of whether or not IP is used on it. PCs don t have to

have an IP address (but some software must be installed on them).

Disadvantages: You can only use protocols the gateway understands (it can do normal Telnet, but not 3270-mode Telnet, for instance). Each active connection loads the gateway some, and this approach will probably be the slowest of the 3 even when only one person is using it.

3. Wollongong has a version of WIN/PC which uses NETBIOS datagrams to transport IP packets over NetWare. They have published an RFC describing the protocol, but I don t know of anyone else who has yet implemented it. There have also been academic developments which I understand used different mechanisms to do roughly the same thing. Either requires a separate, dedicated router to forward the IP packets onto an Ethernet or other TCP/IP media. Wollongong supplies a PC-based router which accomplishes this.

Advantages: You can run any TCP/IP application which supports the interface, and you won t care what media the PC is using. The server is not loaded.

Disadvantages: You must install software on individual PCs. A dedicated router is required, which may be a throughput bottleneck.

So, there it is, as I understand it, today, May 22, 1989.

Bit Ordering of Addresses in MSB and LSB LANs

by Roger Pfister

BICC Data Networks

Earlier versions of this paper were presented to ANSI X3T9.5 (FDDI) MAC Bridge Interest Group and to the IEEE 802.1 MAC Bridging subgroup.

A lot of confusion exists on the subject of bit ordering in the address and data fields on different LANs. This paper attempts to illustrate the similarities and differences between Most Significant Bit (MSB) first and Least Significant Bit (LSB) first LANs. This paper brings out, and tries to clarify the apparently confusing statement that some addresses on MSB LANs do not travel MSB first .

This paper does not attempt to use the language of standards, instead it uses a tutorial style.

Two types of LAN

The world is divided into two types of standardized LAN: The LSBs 802.3 and 802.4 and the MSBs 802.5 and FDDI.

In the following discussion I shall ignore the fact that FDDI does not literally have an MSB on the wire , this is because it uses symbols which are 4B5B encoded, and I shall assume a virtual bit order. Note that in FDDI networks, the symbol carrying the MSB does go first.

Definitions

On an LSB LAN the source and destination addresses are transmitted group bit first and the LLC data fields are transmitted LSB first.

On an MSB LAN the source and destination addresses are transmitted group bit first and the LLC data fields are transmitted MSB first.

I maintain the following statement is true. On an MSB LAN the on the wire bit order of the source and destination address is different than the on the wire bit order of those same addresses when carried as LLC data.

This can be restated as: On an MSB LAN the addresses when traveling as LLC data look different, they do not travel MSB first.

For completeness, the following is true of LSB. On an LSB LAN the on the wire bit order of the source and destination address is the same as the on the wire bit order of those same addresses when carried as LLC data.

The previous definitions and statements have an interesting effect when observed within real implementations. Let us consider a possible implementation of LSB and MSB MACs. In both cases a real implementation can be expected to do the following two things.

1. To deposit all the fields of a received frame in store without special treatment to any field.

2. To have LLC data fields which will be received such that any two MAC implementations produce mutually compatible data. That is, ASCII, sent as data on one LAN, arrives in

store with the same bit order as ASCII sent via the other LAN, and with the same bit order as ASCII is held internally.

It follows then, that if the data looks the same when received from both LANs then the source and destination addresses will look different. That is, the same addresses (in the destination or source field) will arrive in store in a different bit order depending on the LAN type from which it was received.

Comparing Examples

Compare the examples A and B, given at the end of this document. This is the same frame shown transmitted on both an LSB and an MSB LAN. The following points should be noted.

1. The addresses (in the source and destination MAC fields) are the same on the wire in both cases, but they will arrive in store with different bit orders. This is from the rule that puts the group bit first.

2. The group bit in the destination LSAP is transmitted first on an LSB LAN and transmitted last on an MSB LAN. (The LLC fields are all data, so of course they go a different way round depending on the LAN type).

3. The group bit in the SNAP (SubNetwork Access Protocol) organizational identifier is transmitted first on an LSB LAN and transmitted last (bit in the byte) on an MSB LAN.

4. The data, the characters OSI in ASCII, travels MSB first on an MSB LAN and LSB first on an LSB LAN.

5. On the MSB LAN the same address will have a different bit order depending on whether it travels as one of the MAC addresses or as LLC data. Contrast the SNAP and MAC addresses in the MSB example.

The SNAP is a standard method for supporting proprietary protocols within LLC and therefore within OSI. Using the three byte encoding for Dest LSAP, Srce LSAP and LLC frame type which is used in the examples together with a company specific organizational identifier ensures that any protocol so headed is unique. The organizational identifier is the same code as is issued by the IEEE to generate unique 48 bit MAC addresses.

I have used SNAP in the examples because it allows me to demonstrate an example of where a MAC (type) address is transmitted as data and hence has different encodings on the two LANs. An alternative would have been to quote a management standard and this would generate unnecessary detail.

Other Points

The main content of this paper is taken from the Draft IEEE Standard 802.1A Overview and Architecture , which at the time of writing this paper is an 802.1 working document and so not widely available.

Readers of the 802.2 LLC standard will know that the LLC standard 802.2 gives examples of LLC fields in diagrammatic binary. In this style the LLC standard puts the LSB on the left. This convention, which is the opposite of most others, may have been of use when there were only LSB LANs, but now there are LSB and MSB LANs its main effect is to confuse the issue.

No real attempt has been made to give the order of MAC particular fields other than destination and source addresses. This is because the direction of this paper is to clarify issues relating to MAC bridging. When bridging identical MACs such issues should be trivial. When bridging different MACs the only areas of concern are the data, the addresses to which it is going, and from which it has come.

A Plea for a Common Management Encoding

Having collectively caused confusion in the encoding of MAC addresses in LANS we now have a chance, before it is too late, to agree on encoding of MAC addresses across the management interface. It is necessary and reasonable that, if a station is asked what is your MAC address the station should give an answer whose meaning is independent of the type of MAC.

A History How we got here

This is a light hearted look at how we came to have the odd requirement of having to reverse addresses when we cross between MSB and LSB LANs. The main players are the Romans, the Arabs and the two now famous

tribes of Big and Little Endians. It will probably expose my own particular prejudices.

This problem goes back the best part of two millennia. The Romans decided to take up writing and decided they would start words on the left side of the paper and work towards the right. In fact they stole this idea from the Greeks who themselves had been unsure of how to do this thing called writing. For a while the Greeks tried doing one line left to right and the next line right to left!

So there we are, with the Romans writing left to right, but being very poor at numbers. Try multiplying a few M s, X s and I s and you will see what I mean. Also the Romans were in the Big Endian camp, I mean those silly long dates that have an M as the first digit the eye bumps into, that seems very Big Endian to me.

As all the merchants who were trading, and builders who were engineering, needed to calculate quickly some bright spark hit upon the idea of stealing the numbers the Arabs used. This leads us to the Arabs. The Arabs had decided to write right to left.

Have you ever wondered why, out of the four simple ways to write, left to right, right to left, up-down and down-up, that three were adopted separately by major early cultures (I don t know any down-ups). If only there had been an ISO then, imagine a meeting to standardize writing.

Back to the Arabs they are a sensible people not only did they develop positional numerals, but they were definitely Little Endians. Just think about it, if you are reading right to left and you see an Arabic number, then the first digit you reach is the least significant digit Little Endians rule. OK.

So the intellectual descendants of the Romans came to inherit a system in which the words grow in meaning and significance left to right and the numbers grow in significance right to left. From that time on we were doomed to fall into the bit reversing trap. It could all have been avoided if the first scholar to use the new counting system and who also used left to right writing had done a simple digit reversal! So 1776 would have become written as 6771, easy really.

The story now jumps to the early days of data transmission, back before the characters RS232 had ever been typeset. How to transmit data on a serial line well of course the Little Endians were very clear and the data went out LSB first. Why the numbers that form the data tended to be transmitted MSB remains, at least to me, a mystery. I suppose it is because higher layer people tend to be, first digit you see, first digit you send, hence natural Big Endians.

Then came the early days of standard LANs and two events, both of which are necessary to cause the present mess. Firstly 802, or some other equally august body, decided that the first bit of an address must always be the individual/group bit. This advice or ruling must surely predate the seven layer OSI model, because the bit order and style of transmission within a MAC is a local matter to a MAC. What would the ruling have meant if the first MACs used 8 bit parallel data or even more confusingly 16 bit parallel data? I suppose the idea at the time was that it is a good thing to be able to put a scope on the wire and see the global bit go past first, perhaps if FDDI had been about with 4B5B no one would have cared.

The second event that happened, is that the sponsors of the first standardized Token Ring, which later became 802.5, chose MSB first. (Had the ancient scholar done the digit reversal then the modern western world would have been completely Little Endian, and so the idea of going MSB instead of LSB, would never have been considered by any chip blowing engineer). MSB first is a perfectly reasonable way of doing things, because as I have said, transmission between MACs is, or should be, a local matter for that MAC type.

But now the real complication occurred. When the Token Ring sponsors chose to use MSB they also, in my opinion, should have chosen to send all MAC addresses with the group bit eighth (This can be thought of as sending the complete frame, including the MAC address in the MSB style. They should have convinced the standards body to relax the apparent restriction of sending the global bit first or they should not have used MSB. To have done one without the other gives us all the complications we have today.

Finally with the arrival of FDDI they had to pick a method. Once there were two standard methods, each was equally valid and so FDDI chose MSB.

Big Endian and Little Endian are used with acknowledgment to Danny Cohen (and Johnathon Swift) in his 1981 article A Plea for Peace .

P.S. A final sad postscript. The fact that addresses need to be reversed when bridging between 802.4 and 802.5 is all that prevents the same Frame Check Sequence from being used on both LANs, the algorithm is the same but once the MAC destination and source addresses are reversed the input bit stream is different.

Example A. A frame on 802.3 (an LSB LAN)

MAC fields

DEST addr	08-00-4E-01-02-03
SRCE addr	08-00-4E-04-05-06
802.3 LEN	01 23

LLC fields (an example)

DEST LSAP	AA AA 03
SNAP organizational id	08 00 4E
SNAP sub identifier	07 08
other data e.g.	O S I

In the above example the addresses are shown in what 802 calls canonical form or illustrative hex it can be distinguished as a notation by the hyphens between the hex pairs. On an LSB LAN the following bit pattern is transmitted.

The following example has the illustrative hex on the top line followed by the on-the-wire binary pattern, in which the left most bit is transmitted first the bit adjacent to it is transmitted next and so on.

```
   0 8         0 0         4 E         0 1         0 2         0 3
0001 0000  0000 0000  0111 0010  1000 0000  0100 0000  1100 0000
```

^the individual/group (multi-cast) bit

```
   0 8         0 0         4 E         0 4         0 5         0 6
0001 0000  0000 0000  0111 0010  0010 0000  1010 0000  0110 0000
```

^the individual/group (multi-cast) bit

```
   0 1         2 3
1000 0000  1100 0100

   A A         A A         0 3
0101 0101  0101 0101  1100 0000
```

DEST LSAP SRCE LSAP LLC frame type (e.g. unnumbered info)

^the individual/group (multi-cast) bit

```
   0 8         0 0         4 E         0 7         0 8
0001 0000  0000 0000  0111 0010  1110 0000  0001 0000
```

^the individual/group (multi-cast) bit

```
    O           S           I
   4 F         5 3         4 9
1111 0010  1100 1010  1001 0010
```

^the individual/group (multi-cast) bit

Example B. The same frame on FDDI (an MSB LAN)

MAC fields

FC	50	
DEST addr	10 00 72 80 40 C0	(the same addresses
SRCE addr	10 00 72 20 A0 60	as before !!!!)

LLC fields (an example)

DEST LSAP	AA AA 03
SNAP organizational id	08 00 4E
SNAP sub identifier	07 08
other data e.g.	O S I

In the above example the addresses are shown in pseudo illustrative HEX which attempts mirror the value that an implementor would load into a frame. On an MSB LAN the following bit pattern is transmitted.

The following example has the hex on the top line followed by the on-the-wire binary pattern, in which the left most bit is transmitted first the bit adjacent to it is transmitted next and so on.

```
   5 0
0101 0000
```

(an asynchronous LLC frame with 48 bit addressing)

```
   1 0         0 0         7 2         8 0         4 0         C
0001 0000  0000 0000  0111 0010  1000 0000  0100 0000  1100 0000
```

^the individual/group (multi-cast) bit

```
   1 0         0 0         7 2         2 0         A 0         6 0
0001 0000  0000 0000  0111 0010  0010 0000  1010 0000  0110 0000
```

^the individual/group (multi-cast) bit (Different!)

```
   A A         A A         0 3
1010 1010  1010 1010  0000 0011
```

DEST LSAP SRCE LSAP LLC frame type (e.g. unnumbered info)

^the individual/group (multi-cast) bit (Different!)

```
   0 8         0 0         4 E         0 7         0 8
0000 1000  0000 0000  0100 1110  0000 0111  0000 1000
```

^the individual/group (multi-cast) bit

```
    O           S           I
   4 F         5 3         4 9
0100 1111  0101 0011  0100 1001
```

^the individual/group (multi-cast) bit (Different!)