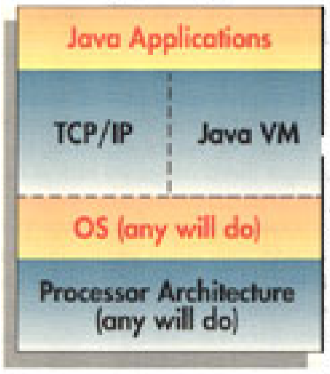
Why I'm Bullish on the Java TCP/IP Combination

In the final analysis, it won't matter if you're using Unix on RISC, Windows on Pentium, or MacOS on the PowerPC. Only the applications will matter, and they will be written in Java. Meanwhile, your server OS will use the standardized TCP/IP protocol suite. No other networking protocols will be required.

Sun, Compaq, Apple and IBM will compete in quality, performance and standards compliance. The leaders will get the newest standards implemented first and win sales. All of the companies will be ISO 9001-compliant to secure sales abroad.

Microsoft will focus strictly on content development. It controls no processor architecture, so it will not have to participate in the race to achieve the greatest performance at the lowest cost. The winners will be those that can fully exploit the synergy that naturally exists between operating system and processor architecture.

Microsoft isn't active in public domain (non-proprietary) standards development, so it won't win in that area either. More importantly, with widespread implementation of Java, Windows becomes just another GUI.



The Java Virtual Machine

Java is a C++-like programming language that can run in either interpreted or compiled mode. More precisely, it is compiled into byte codes and *then* interpreted. This means that, to run Java applications, operating systems and GUI developers only have to implement the Java Virtual Machine (VM) on top of their operating systems.

If a VM is implemented according to the public domain specifications under development by Sun, then any Java application will run on any computer.

Of course, a VM is much slower than a real machine running native code, in which case you would use a Java compiler written for your target processor. And if that still isn't fast enough, Sun is developing Java processors that directly execute the Java byte codes that run on a VM.

Once Java is fully implemented on all platforms, software developers can write just one version of an application program and it will be executable in all environments. This includes networking applications, since Java and TCP/IP operate together.

Fundamental Objections

It seems pretty obvious to me that this is the direction in which we are moving. Sun made NFS completely available to the industry (Sun profits only as technology innovator) and is taking the same approach with Java. And, needless to say, TCP/IP is already fully standardized and available to everyone in the industry. Sun's critics, however, have three fundamental objections to this approach:

1. They say TCP/IP and the Internet are not regulated, and the proprietary fingers of innovators are already creeping in (as I noted in last month's column). Soon Windows will only network with Windows. This is fine, since most networked machines run Windows. They also go so far as to claim that soon *all* networked machines will run Windows.

BRAD HARRISON

If a VM is implemented to the public domain specs, any Java app will run anywhere.

2. They testify that Java can only address a subset of operating system calls and GUI libraries. The devil is in the details, so you need the whole operating system and GUI available to applications. Java is necessarily too general to provide a complete solution.

3. Finally, critics point out that networking is really more of an art than a science, and that interoperability on multivendor networks doesn't work well. Thus, system administrators are moving toward single solutions from single companies. Of course, what they are really saying is that companies are moving to NT-only networking because it's easier.

The first two are good points and I have no firm rebuttal. Internet standardization is at a standstill. And Java does simplify, maybe too much.

The third criticism, however, I will refute. Digital Equipment had a completely scalable networking solution with Vax VMS — "from the office to the factory floor." Vax VMS and DECnet, if you could afford them, worked great, regardless of how many machines you required. But Vax VMS was a technological triumph that fell flat on its face because of cheap PCs and fast Unix workstations.

In retrospect, it was the almighty dollar (lots of computing cycles for just a little money) and the desire for freedom from a single vendor that prevented Vax VMS from succeeding. This is why I don't see the world standardizing on NT. I think people will be more likely to standardize on TCP/IP and Java, not on products from a single vendor. ▼

Brad Harrison is an engineer at Microtec Research, a subsidiary of Mentor Graphics (Wilsonville, Ore.). harrison@cardinal.com.