by MICHAEL O'BRIEN

"With Dayna's PC Card Roamer, Newton users are free to roam about an office environment without the constraint of network wires." –Dayna Web Page

"Network security included data scrambling." –Xircom Web Page. Indeed.

"Egg freckles." –Write this on a Newton MessagePad 130, then tap "Assist."

Mr. Protocol Has It All in Hand

Q: *I give up. You've had me running around outside long enough on this mobility kick. I'm giving up. You'll find me on the park bench feeding birds.*

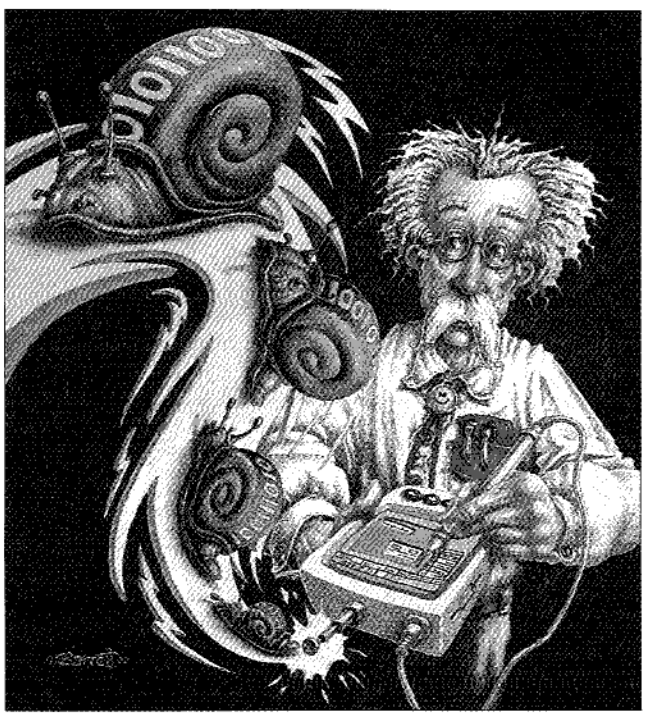
A: Good for you! It's about time someone got the message. There is more to the definition of "outside" than "the place where there's no 10base-T." But there's more to mobility in computing than wandering around in the never-never talking to satellites. It's useful indoors too. The problem is not finding really cool instruments. The problem is in conducting the orchestra.

Ask yourself, first of all, what you're trying to do when you think of mobile computing as the answer. What was the question? Either you're trying to duplicate the ability to do something you commonly do in your "home" computing environment, or you're not–i.e., you're trying to do something you don't do when you're at home, something that's purely mobile. In the former case, you're attempting to attach a long umbilical cord to an existing application. In the latter case, you're dealing with a whole new set of requirements. These are knotty problems, both of them.

Let's take a look at the umbilical cord first.

Out in the Field

If we want to do something in the field that we also do "back home" on the desktop, then we need to extend network services from the desktop to the field. Of course, we probably won't do just this. What we really want out in the field is access to one or several servers, the same servers we use from the desktop. The umbilical cord, then, represents replacing the desktop LAN connection with something less material.



The first thing we notice is that bandwidth has fallen by at least a factor of 10. Standard Ethernets run at 10 Mb/s, while the fastest wireless infrared and radio networks run at between 1 and 2 Mb/s. Typical 10 Mb/s Ethernets do not achieve anything like this rate on a sustained basis, of course. It is intrinsic to the design of Ethernet that congestion sets in at between 40% and 60% saturation. However, infrared and radio networks share this characteristic, so their effective bandwidths are cut back commensurately.

Even this factor of 10 is a best-case scenario. Wide-area wireless networks have a much worse speed differential. The best they can do is about 28.8 Kbaud, the same as a dial-up modem. This speed mismatch is so great that a simple extension of the umbilical cord is no longer possible. More thought must be given to the matter.

The first question is, why doesn't it work? What are we doing with all that bandwidth that we need it so much? We're certainly not reading the display at that rate of speed.

In most cases, the answer is housekeeping. We're using the bandwidth of Ethernet, and clamoring for more, because the network protocols we're using have not been tuned for the specific applications we're running. And this is a good thing. If every application needed its own specially tailored network protocol, we'd be in sorry shape. Protocol standards would be worthless, and interoperability would be nil. As it is, we can define a low-level

TOM BARRETT

protocol that just provides access to (for instance) a file system, using a set of generic calls, and suddenly we get to put all our disk drives on a nice fast server and use the network to move file system data around. This is the principle behind NFS.

Other low-level protocols, less obvious ones, are similarly wandering around, monitoring the network and keeping all the clocks set. High-level protocols include HTTP, which is used to move around all the information on the Web, and FTP, which allows people to write Web browsers like Netscape that can do the same file transfers as the FTP command, without boodles of special-purpose code.

The price we pay for all of this is overhead. The packets used by these protocols carry a lot of dead weight information that most applications don't need...although the "deadwood" of one application may be important to another. The point is that most of the stuff flying around on the wire is there because it's easier to write applications that use existing protocols than it is to design new ones each time, and it makes interoperation easier too.

Once we're in the regime of tight bandwidth, though, design constraints become more rigorous, and the cheap off-the-shelf protocol solution may no longer work.

Let's take an example. Consider an inventory application. In its desktop incarnation, a few query strings are transformed into a network query to a back-end database server. This might be split up in one of several ways. The application might be rolled into a single unit: Front-end interface, back-end database and application "glue" code present as a single program. In this case, the network traffic amounts to file system access traffic via NFS, AFS, UFS, Novell Inc. NetWare, or what-have-you. Or the application might represent a front-end GUI bound with a query processor that spits CORBA database queries, or the equivalent, at a back-end database engine managing local disk storage.

Now, what we want to do is to put this application onto laptops in salespeople's cars.

The first thing to do is to look not only at the channel capacity of the wireless WAN service we're thinking of using, but at the protocol or protocols that it supports. Some services are choosing to join the TCP/IP camp, not only because many applications are currently being written to use TCP/IP, but also because the protocols are public. Just by putting effort into the Internet Engineering Task Force arena, you can propose a standard for carrying IP over your XYZ native wireless network and stand a fair chance of having it adopted, assuming your XYZ network isn't too heavily proprietary.

Most of the stuff flying around on the wire is there because it's easier to write applications that use existing protocols than it is to design new ones each time, and it makes interoperation easier too.

Examples of this kind of TCP/IP wireless WAN network are the Metricom Ricochet service mentioned in previous columns, and the technology being pushed by the cellular phone companies to make use of their own unused channel capacity, known as Cellular Digital Packet Data, or CDPD. The Ricochet service runs at between 14.4 and 28.8 Kb/s. The native speed of CDPD is 19.2 Kb/s, but the error recovery protocol used cuts that down to an effective 9.6 Kb/s.

At these rates, the all-in-one architecture for our inventory application is a no-hoper. Take that sucker out and bury it, because it's not going to play in Peoria, nossir. Disk access protocols are far too verbose to be carried over links as slow as 28.8 Kb/s unless something dire is going on, and "dire" had better mean "extremely temporary" because no human with the average life span is going to be willing to wait for that system to return an answer. Other activities, such as eventually marrying and having a family, will intervene.

So why should you believe Mr. Protocol about this? He's glad you asked. Even though the thought of his ever producing a family is one of the more shuddersome thoughts to come under this roof, he does have better things to do with his time than to wait for a protocol as flatulent as these to bleed over a slow link.

Sniffing Out the Competition

To test his hypothesis, he invites you to beg, borrow or liberate in the name of the people a LAN sniffer of some description, and watch these protocols in action. If you can, restrict the sniffer to observing traffic between a single node and the file server, and then perform an `ls` on the node, or open a file in an editor. A jaw-dropping amount of activity results, including things like access authentication. Over a slow link, you want to authenticate the connection, not every single file access. But that's only the beginning. Watch what happens during a directory lookup. Make sure your sniffer has lots of buffer space.

This does not mean there's something wrong with these protocols. Not at all. It's just that they're designed for a much higher-speed regime than is supported on current wireless WANs. If our application uses such protocols, then it will have to be rewritten; it cannot use CDPD.

But what if our application uses CORBA? In that case, a traffic analysis will have to be performed to determine how verbose the application is in practice. Find out how much traffic actually moves in servicing a query, then multiply it by the speed differential and determine if the answer will come back before old age removes any interest in seeing it. There's likely to be more to it, of course. If the protocol has timeouts, for instance, they may have to be changed. But this architecture has at least a chance of working without major architectural reengineering.

What do we do if the application does have to be reworked? The most likely road to success involves analyzing the flow of information between the major segments of the application.

The graphic front end might be a good place to start. It's likely that the user's query involves only a few small strings. The answer would be a somewhat larger bundle of strings plus perhaps a compressed GIF image or two. This might be a good point to divide the application. Put the front end into the field, and leave the rest of the application on a server back at the office, running as a proxy. The GUI talks to the proxy over the air, and the proxy communicates with the database and sends the answer (only) back over the air to the GUI running on the laptop in the field. The downside is that a protocol, however small and simple, will have to be designed to carry the queries and answers back and forth.

Which Protocol?

And here we begin to see our stumbling block: What protocols are supported by the wireless LAN or WAN solution we choose? CDPD and Ricochet support TCP and IP. But most LANs have various other protocols swarming around. On the desktop, packets come and go in Ethernet format, and ARP, the Address Resolution Protocol, supports the binding between IP addresses and Ethernet MAC addresses.

With Ricochet and CDPD, though, the IP address is pre-assigned to the modem, and the wireless host is insulated, and isolated, from access to the actual transport layer. Instead, the host must use SLIP or PPP to communicate with the modem and from thence, to the rest of the network.

Similarly, the graphical front end in our example must be made smart enough to open a TCP/IP connection back to its proxy. This won't be too hard on a UNIX platform, though there are precious few of those running on laptops, but separate third-party software would have to be bought for a DOS or Windows machine, though not for Windows 95 or Windows NT. A Mac Powerbook would also probably have the necessary TCP/IP drivers. In any event, the GUI would have to be coded to use whatever API is presented by the TCP/IP stack on its machine, and to use its own special "inventory application" protocol on top of that to move the queries and answers around.

Back in the office, where bandwidths are higher, a similar question arises. Consider two home pages, chosen more or less at random. The first is `http://www.xircom.com`, and the second is `http://www.dayna.com`. These two companies make "compatible" wireless LAN hardware, in the sense that what one transmits, the other company can receive and make sense of. So far so good. But what does that mean?

Xircom's marketing stresses PC solutions. Therefore, its literature is full of references to Novell Inc. NetWare and various operating systems sold by Microsoft. Dayna's corporate approach emphasizes connectivity solutions between Macintosh and PC platforms. Which ones would you like to plug into a Sun? Good question.

After digging, the reader will eventually determine that the radio modems and the base stations alike are protocol-independent, and act merely to extend the wire-based LAN, whatever protocol(s) may be moving over it. Hence, a Xircom base station can be attached to a Macintosh Ethernet network and will happily squirt AppleTalk over the air...probably.

Mr. Protocol's advice is to ask for a demo first. The entire wireless arena is so new that it is not entirely clear which solutions look good on paper and which ones will actually work. Some of the wireless attachment points might act as routers, in which case they will route only the protocols they understand. Others will act as bridges, and will be able to route any protocol.

One final hurdle, not to say *the* final hurdle: What is the mobile platform? Currently, PCs own this area outright. There are some strong Mac Powerbook users out there, but Sun-style UNIX boxes don't have a strong showing in the general market, Tadpole and BriteLite offerings notwithstanding. PCs speak Novell if they speak anything, at least if they're running a Microsoft operating system they do. Novell now also offers NFS support, and other applications such as Netscape use TCP/IP, so possibly the over-the-air network is IP encapsulated in a packet radio format.

The Microsoft world just isn't too well geared yet to network operation, at least not compared with the UNIX world, so applications aside from simple email might be scarce. Email is important, but mobile email within the building is probably less important than other applications. On the road it's a different story, of course, but within a building, other things probably take precedence.

Office Mobility

Usually, being mobile inside a building means one of two things: There's a meeting going on and you want your computer environment there at that meeting. Or you're trying to find a place to hide and get some real work done. What you do in these two situations might be very different.

If you're hiding, you'd like to hide with the same environment you have on your desktop, since you want to do the same sorts of things you do at your desk, but without the interruptions. If you have a Sun on your desk, you would ideally like to have a Sun portable in your hidey-hole. Portable SPARC machines are still much more expensive than laptop PCs, though. The other choice is to run an X Window server on an ordinary PC laptop. These machines eat inordinate amounts of network bandwidth, though, because they're slamming bitmaps around.

One answer might be to run a UNIX variant on a PC laptop. This would cut down on network traffic over a PC running an X server, at least somewhat alleviating the 10-to-1 speed differential between wired and wireless, and would still be cheaper than running a SPARC laptop.

If you're at a meeting, though, it's a different matter. One solution that's rapidly increasing in popularity is to

take all the important information–everything–and dump it into a gigantic corporate Web server. Then, use a Web browser at meetings to call up whatever information is needed. This allows the use of any platform, be it Sun, Mac or PC, that can run a Web browser. For these extremely mobile applications, though, pen-based computing is becoming increasingly popular, and this is an area where the idea of integrating with a wider network has superb advantages, yet is still quite primitive. This is due to the early stage of development of pen-based computing in general.

Consider, though, how useful it would be to be able to sit at a meeting with a tablet that is all screen and no keyboard, and navigate your company's database just by tapping and writing with a pen. You could do this on a legal-sized tablet, or you could do it on a pad held in the palm of your hand, all at an effective rate of 200-300 Kb/s. If you choose your network, your applications and your protocols carefully in this rather thin market, you can do it today. And when the body of available applications begins to fill out significantly, this market is going to take off, Mr. Protocol thinks. But it's one of those advances where you have to hold it in the palm of your hand before the epiphany occurs.

Mr. Protocol thinks you should give it a try. ➸

Mike O'Brien has been noodling around the UNIX world for far too long a time. He knows he started out with UNIX Research Version 5 (not System V, he hastens to point out), but forgets the year. He thinks it was around 1975 or so.

He founded and ran the first nationwide UNIX Users Group Software Distribution Center. He worked at Rand during the glory days of the Rand editor and the MH mail system, helped build CSNET (first at Rand and later at BBN Labs Inc.) and is now working at an aerospace research corporation.

Mr. Protocol refuses to divulge his qualifications and may, in fact, have none whatsoever. His email address is amp@cpg.com.