

# Managing

*At a glance* ..... 50  
*Acronyms* ..... 50  
*E-mail is timeless* ..... 50  
*Other considerations* ..... 52  
*Power-line backbones* ..... 54  
*For more information* ..... 56  
*Firewalls* ..... 58

# Internet-enabled DEVICES

YOU'VE GOT A  
DEVICE—MAYBE EVEN  
1000 OF THEM—  
CONNECTED TO THE  
INTERNET. NOW WHAT  
DO YOU DO?

**F**OR YEARS, VENDORS HAVE TALKED about how every device in the world will one day be connected to the Internet. Many of the scenarios presented are sheer fantasy, such as refrigerators that order milk for you. Many are questionable in the value they add, such as alarm clocks turning on coffee pots or consumers being able to

adjust their home thermostats from PCs at work (so their cats don't get too cold, presumably). Some propose business models that just can't seem to get off the ground; salespeople have enough trouble selling banner ads on the Web, never mind pitching advertising on tiny black-and-white LCD panels on gasoline pumps. Many tout content, such as streaming video, as the killer app, but who's going to watch a movie on an Internet appliance when he or she can watch it on a big-screen TV? Content alone as the killer app doesn't cut it, because the alternatives to streaming content do a better job and provide better quality. Most of the scenarios simply make no financial sense; intelligent light bulbs haven't been dubbed the "zero-billion-dollar" market without reason. And some ideas, quite frankly, are just plain stupid.

Depending on whom you speak to and the functions you need, connecting a device to the Internet will run you \$10 to \$70 on the low end. To justify this cost, connectivity has to add real value and

provide guaranteed returns, not just promised income *if* (and it's a big *if*) users decide to activate a new pay-per-use feature. For most devices, this value will arise from remote management.

Remote management of a device offers several benefits. A device can set off an alarm when it approaches operation thresholds that indicate it will soon fail. This warning enables technicians to make service calls only to those devices that need it and perform maintenance before greater damage occurs. Events as mundane as a vending machine running out of a brand of soda or as serious as a mechanical failure could trigger alarms. Devices can also automatically receive code updates without costly service calls. Updates also feature optional upgrades that users can request (and pay for). Devices can report on what is happening around them (for example, a user purchased a pay-per-view movie yesterday) or receive information (downloading new photos to a digital photo frame). They can also collect statistics for inventory (consump-

tion patterns for automatic restocking) or performance metrics (indicating why one device outperforms another). Additionally, you can direct the actions of a device based on information collected at an aggregated level.

### GETTING CONNECTED

The most flexible designs require no specific connection to the Internet; they can use a variety of physical connections, the most common being phone lines, Ethernet, power lines (see **sidebar** “Power-line backbones”), and wireless connections. A flexible connection allows you to add connectivity across multiple product lines and applications. However, each of these physical connections has characteristics that greatly affect a device’s functionality. The timeliness of the data also affects the type of connection. A set-top box can wait a few days before calling in a charge for a pay-per-view movie, but a news appliance needs to load information regularly in the morning and evening, and a security system needs immediate access.

Phone connectivity is one of the most common methods of connecting devices. Most environments have a phone line available, making the bigger challenge finding a phone jack in a convenient place. Given that, at some point in the day, the phone is not in use, the device can “borrow” the line without interfering with normal use. This model represents a passive server model. A management server collects information passively when the device calls in. Without a spe-

#### AT A GLANCE

- ▶ Effective remote management can either reduce costs or increase efficiency to the point of justifying connecting a device to the Internet.
- ▶ Connecting to a LAN using Ethernet usually results in the most cost-effective connection, but firewalls can be a real pain.
- ▶ Web servers just don’t cut it when you are managing more than a few devices.
- ▶ You must take into account the cost of developing and managing a management server from the start of development.
- ▶ Once you’ve established remote-management capabilities, streaming content and e-commerce come for “free.”

cial ring, a server could not call the device without interfering with normal use of the phone.

The problem with a passive model, however, is that if the device doesn’t call in (if the phone is left off the hook for a day or two, the wireless connection is jammed, or the ISP is down), the server has no way of knowing whether the device is down, can’t connect, or has nothing to report. Also, if the device needs a critical upgrade, the upgrade can’t take place until the device calls in. Put another way, if the device calls in once a month, then one month of additional revenue is lost. In this light, it makes sense to support some kind of active model, in which the server can initiate an exchange, even if only for emergencies. Methods of getting through to the device could range from something as straightforward as the customer’s pressing a reset button so that the device calls in at the next available opportunity (manual) or the device’s listening in on the beginning of every incoming call to see whether the call is for it (automatic).

Supporting only an active model also has its problems. For example, if a device hasn’t heard from the server in a while, it needs some way of contacting the server and acting autonomously in the meantime. The server should be aware when it can’t contact a particular device so it can alert a technician to check out the device. But this process could be expensive, especially if the server has no idea why the connection is down. The device could

have a second, physical “emergency” connection or some way of alerting the person servicing the machine that the connection is broken. Sometimes the connection is OK, but the server or ISP is faulty or down. One recovery method is to have the device check in with a known secure site instead of the server it normally contacts.

The most common direct network connection is Ethernet. The advantages of an Ethernet connection are many but only if you can actually connect to a LAN. Devices connected to a network via Ethernet have a continuous or persistent connection that can also be a two-way connection (the device or the server can initiate an exchange). Also note that Ethernet doesn’t have to run at 100 Mbps to be useful; using 8-bit processors, Ethernet can cost you much less than an embedded phone modem. The major issue with using the two-way Ethernet connection, however, depends on whether the device falls behind a firewall (see **sidebar** “Firewalls”) and whether you can get permission to connect your device to the network.

Wireless is a tempting connection technology because it promises the most flexibility in device placement. However, many environmental factors can knock a wireless link loopy. For example, 2.5-GHz spread-spectrum phones can pound devices communicating using the Wi-Fi standard. The Wi-Fi group suggests that home users stick with 900-MHz phones to eliminate the problem. However, doing so doesn’t prevent your neighbor from using equipment that

### ACRONYMS

- API: application-programming interface
- FTP: File Transfer Protocol
- HTML: Hypertext Markup Language
- HTTP: Hypertext Transfer Protocol
- IP: Internet Protocol
- IPSec: Internet Protocol security
- ISP: Internet-service provider
- LAN: local-area network
- LCD: liquid-crystal display
- MIB: management-information base
- SNMP: Simple Network Management Protocol
- SSL: Secure Sockets Layer
- TCP: Transmission Control Protocol
- UDP: User Datagram Protocol
- WAP: Wireless Application Protocol
- Wi-Fi: Wireless Fidelity
- XML: Extensible Markup Language

### E-MAIL IS TIMELESS

You might also consider designing a device that supports e-mail. Being able to send e-mail offers several distinct advantages over having to communicate directly with a server. For example, you can both send and retrieve e-mail at unscheduled times, meaning that a device can send an e-mail update through an ISP at a time convenient for it (instead of during a particular time window), and the server can retrieve the e-mail at a time convenient for it. Note that standard software is available for extracting data from e-mail. Devices can also send e-mail directly to individuals as alerts. If the device can also receive e-mail, then it can receive information at unscheduled times as well.

brings your link to a crawl. A wireless connection can be as simple as a pager. The infrastructure for paging already exists and could be less expensive in the long term, depending on the application. Note also that paging can support broadcasting in the same manner as it sends out sports scores or stock quotes. This setup enables fast transfer of information on a mass scale (such as might be necessary in an application in which thermostats are throttled back to prevent an impending blackout).

Computing costs for a connection go beyond merely hardware and software. To avoid ringing up long-distance charges on a borrowed phone, you'll either have to dial an 800 number or go

through a local ISP. Many devices in the same area could share an ISP account, which could cost less than the 800 number charges over time. There are also extra service charges to consider, such as initiating pager alerts. Also consider that a cellular connection could cost less than monthly phone-line fees. Additionally, if the device accesses the Internet through a local ISP, the device has to work with various ISPs and phone systems. Consider an 8-bit processor to handle your connection if your data rates are low enough; the speed your processor requires depends on the data rate you need. The hardware savings over 16- or 32-bit systems can be substantial. However, the extra horsepower of a 16-bit

processor might be worth it if you need more speed, functions, or both from your connection, or you can use the headroom on the processor. You might also consider using two 8-bit processors—one for connectivity and one for applications.

Another consideration is whether to embed connectivity in a device or use an external peripheral. Adding a peripheral that can query a device for status and then pass that information on to a management server may be less expensive than redesigning and replacing the device. The peripheral could be as simple as a pass-through device that makes a device think it is connected to a network via a LAN when it's actually connected via a

## OTHER CONSIDERATIONS

There are many issues to keep in mind as you design Internet-enabled devices. The first question to ask is what infrastructure you can count on being available. For example, a device placed in a home most likely has access to a phone, but a device on a factory floor may not.

Devices contain two kinds of connectivity functions: connectivity for standard operation and connectivity for "emergency" operation when standard connectivity fails. If you include emergency connectivity, it pays to consider how you might add standard functions through this connectivity, because you're already absorbing its cost.

Devices that interact with people can reduce costs by offering online manuals and help text. However, you can't get online help if you can't get online. Such devices need to support some kind of device-resident help or display a support-line phone number. Also, small stacks, while perhaps not providing all the bells and whistles of TCP/IP, may enable you to add connectivity to a device using existing headroom, so you don't have to redo your design.

Another consideration is that some vendors provide code that performs the function (such as FTP) but not the application

(such as supporting code upgrades). In addition, if a device is going to call out late at night, it has to know what time it is. Better if the user doesn't have to take care of setting the clock, or it could end up blinking endlessly like the one on many VCRs. A somewhat accurate clock (even an interrupt counter) could be effective enough if it recalibrates to the correct time each time the device calls in. A device should also have a controlled maximum latency, meaning that if it hasn't heard from the server within a specified time, it should try to contact the server or a known secondary server.

It's important to remember that the connectivity portion of your design is not independent of the rest of the design. The processor handling your connection can run applications if they don't need to run while a connection is live.

Depending upon the sensitivity of the data you collect, you may want to consider encrypting data during an exchange. A proprietary data format offers limited security, because a hacker doesn't know what each bit means. You might consider using standard encryption formats, such as SSL or IPSec, although there are holes in this kind of

security as well. Also keep in mind whether the data-collection software you're considering also supports remote control and upgrading of devices.

It's easy to focus only on the device and not think about the overall network architecture and tools you'll need. Ask yourself whether the management software you're considering assumes a direct device-to-server relationship, thus eliminating your ability to use remote managers. Remember that, in a passive network, devices have to contact the server. If remote managers are also passive, the latency between the posting of a message and its receipt depends on the depth of the hierarchy and frequency of connection.

The difference between how you perceive a device will be used and how the final user actually uses it can be extreme. Being able to update only a small percentage of installed devices lets you test enhancements to check whether they are really problems in waiting without bringing down the entire network base.

You also need to think about whether you've designed for the future. Ask yourself whether you left processing headroom for future functions, and, if you intend to support content such

as advertising or video, think about whether you have sufficient memory resources available.

Is the development system shrink-wrapped? Is it an API with complex commands behind simple requests? Or is it built using fundamentals you can do much more with (but require a greater investment)? How important is security to this application or the consumer? If you want to expand the application to include e-commerce, you'd better put security in or leave room for it. Is authentication built into the management software you're considering using? If so, just how secure is it?

Check pricing early in the evaluation process. Pricing models on software going into high-volume devices (one-time fee, royalty per device, cost per deployed device, and other models) can price your device right out of its target market. Creating a large network also requires that you design your devices for ease of installation. Otherwise, it will take you forever to deploy devices in large volumes.

Finally, remember that remote monitoring has the potential to track the usage patterns of users. Consider privacy issues before you collect and expose such data.

phone, or it could use a scheme as complex as monitoring the device, making decisions, and managing the connection to a server. The SmartStack Box from eDevice, for example, is a socket modem that connects to devices via a serial connection and to the Internet via either a V.32 modem or Ethernet. The box sells for \$73 (1000). Peripherals support legacy equipment and speed time to market. They also allow you to test the viability of adding connectivity, and you can integrate them over time to reduce design costs.

The connection rate you need depends mainly on the amount of data you need to exchange and the amount of time you have to exchange the data. For example, taking an hour to download photos to a frame may not matter if you're doing it at 3 a.m., but using a whole hour limits the number of devices a single server can connect to in an evening. Also, when you consider a large number of devices, the frequency with which exceptions occur increases to the point at which you can count on them happening. A server's efficiency must leave capacity for busy signals, broken connections, and failed downloads. Compression is one way to increase the efficiency of a connection. However, you need to balance the cost of the connection with the added cost of supplying enough processing power to perform the compression (see **sidebar** "Other considerations").

#### IT'S NOT WHAT YOU SAY BUT HOW YOU SAY IT

Once you establish a connection, you have a choice of formats for how to send data. Internet-appliance advocates used to tout sending data in formatted Web pages using embedded a Web/HTTP server as *the* method for transferring data from Internet-enabled devices.

One advantage of using a Web server is that any browser attached to the Internet can query the device and view data. It also means that a device does not require its own display or input interface (a keyboard, for example), dropping total device cost. There are also servers that support fewer functions for devices with limited memory and processor resources.

However, when you begin to consider connecting many devices, viewing the status of each device becomes impractical, and automated data aggregation becomes necessary. At this point, it becomes less important to format the data for human eyes and to pay for the extra bandwidth required by HTML coding, which increases the cost of connection (time) and limits the number of devices a single server can manage. Additionally, when you consider the different formats in which you might want to transfer information—WAP for wireless browsers, e-mail (see **sidebar** "E-mail is timeless"), different report types/formats, and others—the overhead to maintain these multiple formats in the device, as opposed to letting the server format the data for human eyes, can become quite expensive.

One data format that is gaining acceptance is XML. Note that XML is not the panacea some vendors make it out to be. For example, if you have one server querying devices, then you don't need XML; a proprietary format can reduce overhead to a minimum. Some vendors claim that XML is useful if you want to access the device from anywhere on the Internet. However, you still need a browser that understands all the XML data objects and what they mean, which is effectively the same thing as using a browser plug-in that interprets your proprietary format. Another claim is that XML is

scalable, meaning that a device that hasn't been upgraded to understand new XML tags will ignore them. However, this feature is not exactly impossible to implement in a proprietary format.

What makes XML most useful is the availability of off-the-shelf software. Because accessing and creating XML data objects is based on standard methods, the theoretical portability of XML data becomes real, and you and your customers will have access to software to manage devices that neither of you has to write. For example, eMation offers the DRM (Device Relationship Management) software suite. DRM Connector, for the device side, takes care of capturing information from the device (through an API that the device calls) and putting the data in packets to send to the server. It supports creating reports, sending alerts, and encrypting data using SSL. On the server side, DRM Server captures and aggregates collected information. It takes care of displaying information so that the exchange with the device doesn't have to bear this overhead, and it can base the display on the software version the device supports. The server also handles rights provisioning (some devices may support a paid-upgrade feature) and access management (for example, a vending-machine attendant may have access to machines only in his or her service area).

Much of the value of XML resides in the use of schema. A schema defines the type of data a device collects and how it sends it. For example, a device may have a model number and several modes of communicating information. However, once devices are deployed, a troublesome condition may arise. In such cases, you need to update the schema to change the type of data that the device collects, how often it collects and reports the data, and

## POWER-LINE BACKBONES

Power-line technologies, in which data travels over a power line, are attractive because the line that powers a device can also serve as its data backbone. Power-line technology is appropriate for environments in which running a phone or Ethernet line or going wireless just isn't feasible (consider putting a vending machine inside a warehouse). One problem with power-line

networks, however, is that they require a bridge to connect them to the Internet. For example, although a photo frame using a phone line also requires a bridge (either a local ISP or dialing directly to the server), such bridges are standard and common. A power-line-connected frame requires a gateway power-line node that connects to the phone or LAN. You can

conveniently plug in this extra gateway near a phone or Ethernet line, allowing you to place the frame pretty much anywhere and move it easily but raising the overall system cost. Note that one gateway could aggregate or concentrate traffic, thus spreading the cost of the gateway across several connected devices. A wireless network would provide nearly identical

mobility. One disadvantage of power-line networks is cost. Echelon, which offers power-line transceivers and protocol software, estimates a \$50 to \$70 bill of materials for 8-bit embedded power-line designs. At extreme volumes (30 million), the company claims to drop the average cost to \$10 for a power-meter application.

how it acts upon the information. Such software also allows you to “shadow” your network. The management server effectively creates a copy of each device in the network. Administrators work with the shadow copy as an object of data, enabling an abstract view of the network at different levels, such as by group or application.

Note that the complexity of XML isn’t in creating the schema. For low-cost devices, you can code the schema directly into a device. The complexity comes in collecting, interpreting, and managing the data on the server side. Server-side software has a tremendous impact on the design and operation of devices, so it is important for you to have an idea of what software you will supply, support, or recommend to your customers. For example, if you don’t know when or where a device will be deployed, the server needs to support a dynamic database in which devices can automatically register themselves. You also need to know whether the

software is capable of managing mass upgrades, how it collects data, how efficient and timely that collection is, and so on. Intelligence is partitioned between the server and the device, and if the server doesn’t support certain critical functions, you need to enable the device to provide them. However, if the server offers these critical functions, you needn’t spend time designing them yourself or allocating resources on the device for them.

#### DISCOVERY

A server managing many Internet-enabled devices needs to be aware of these devices. The server can “discover” devices in two ways: In an active network, the server goes out on the network and finds devices; in a passive network, devices register themselves.

One method a server can use to seek out devices is SNMP (or the less reliable UDP), which discovers nodes in enterprise networks. The server broadcasts a private MIB, to which only devices pro-

grammed to understand that MIB respond. A new device responds to the MIB, and the server then registers or discovers the device. You don’t need a full SNMP implementation for discovery, and you could use an alternative proprietary protocol. However, SNMP is an established standard with many tools supporting it.

RTI (Real-Time Innovations) offers an alternative to SNMP with its NDDS (Network Data Delivery Service), which uses a publish/subscribe model. When you attach a new device to the network, the device broadcasts the “topics” it may “publish” in the future. All other devices interested in these topics then “subscribe” to them by sending a request to the device. Note that each device must maintain a directory of devices that subscribe to its topics. Because each device maintains its own discovery directory, there is no need for a central server to enable exchange of data between devices (peer to peer). This setup enables devices

## FOR MORE INFORMATION...

For more information on products such as those discussed in this article, go to [www.ednmag.com](http://www.ednmag.com) and click on the Reader Service link under the Tools & Services section. When you contact any of the following manufacturers directly, please let them know you read about their products in *EDN*.

#### Atmel

1-408-441-0311  
[www.atmel.com](http://www.atmel.com)  
Enter No. 301

#### eDevice

1-212-856-0000  
[www.edevice.com](http://www.edevice.com)  
Enter No. 307

#### IBM

[www.ibm.com](http://www.ibm.com)  
Enter No. 313

#### Microchip

[www.microchip.com](http://www.microchip.com)  
Enter No. 319

#### PointBase

1-877-238-8798  
[www.pointbase.com](http://www.pointbase.com)  
Enter No. 325

#### Smart Network Devices GmbH

+49 2461 690620  
[www.smartnd.com](http://www.smartnd.com)  
Enter No. 330

#### Bsquare

1-888-820-4500  
[www.bsquare.com](http://www.bsquare.com)  
Enter No. 302

#### Elmic Systems

1-415-421-2700  
[www.elmic.com](http://www.elmic.com)  
Enter No. 308

#### Intellon

1-352-237-7416  
[www.intellon.com](http://www.intellon.com)  
Enter No. 314

#### NetBurner

1-800-695-6828  
[www.netburner.com](http://www.netburner.com)  
Enter No. 320

#### Precise

1-613-596-2251  
[www.psti.com](http://www.psti.com)  
Enter No. 326

#### SNMP Research

1-865-579-3311  
[www.snmp.com](http://www.snmp.com)  
Enter No. 331

#### CMX Systems

1-508-872-7675  
[www.cmx.com](http://www.cmx.com)  
Enter No. 303

#### eMation

1-508-337-9200  
[www.emation.com](http://www.emation.com)  
Enter No. 309

#### InterNiche Technologies

1-408-257-8014  
[www.iniche.com](http://www.iniche.com)  
Enter No. 315

#### NetSilicon

1-781-647-1234  
[www.netsilicon.com](http://www.netsilicon.com)  
Enter No. 321

#### Questra

[www.questra.com](http://www.questra.com)  
1-888-515-5379  
Enter No. 327

#### Ubicom

1-650-210-1500  
[www.ubicom.com](http://www.ubicom.com)  
Enter No. 332

#### Connect One

1-408-968-9602  
[www.connectone.com](http://www.connectone.com)  
Enter No. 304

#### emWare

1-801-453-9300  
[www.emware.com](http://www.emware.com)  
Enter No. 310

#### Intrinsyc

1-604-801-6461  
[www.intrinsyc.com](http://www.intrinsyc.com)  
Enter No. 316

#### Opto22

1-800-321-6786  
[www.opto22.com](http://www.opto22.com)  
Enter No. 322

#### Rabbit

**Semiconductor**  
1-530-757-8400  
[www.rabbitsemiconductor.com](http://www.rabbitsemiconductor.com)  
Enter No. 328

#### Yipee

1-716-250-0481  
[www.yipeeinc.com](http://www.yipeeinc.com)  
Enter No. 333

#### Dallas Semiconductor

1-972-371-6031  
[www.dalsemi.com](http://www.dalsemi.com)  
Enter No. 305

#### Gatespace

1-650-846-6580  
[www.gatespace.com](http://www.gatespace.com)  
Enter No. 311

#### Lantronix

1-949-453-3990  
[www.lantronix.com](http://www.lantronix.com)  
Enter No. 317

#### OSE Systems

1-408-392-9300  
[www.ose.com](http://www.ose.com)  
Enter No. 323

#### RTI (Real-Time Innovations)

1-408-734-4200  
[www.rti.com](http://www.rti.com)  
Enter No. 329

#### Zilog

[www.zilog.com](http://www.zilog.com)  
Enter No. 334

#### Echelon

1-408-938-5200  
[www.echelon.com](http://www.echelon.com)  
Enter No. 306

#### Hyundai Electronics America

[www.he.com](http://www.he.com)  
Enter No. 312

#### LiveDevices

+44 (0) 1904 562 580  
[www.livedevices.com](http://www.livedevices.com)  
Enter No. 318

#### PlanetWeb

1-650-551-5500  
[www.planetweb.com](http://www.planetweb.com)  
Enter No. 324

#### SUPER INFO NUMBER

For more information on the products available from all of the vendors listed in this box, go to [www.ednmag.com](http://www.ednmag.com), click on the Reader Service link, and enter no. 335

to take on roles such as temporary remote managers or to provide redundant connectivity. You can keep the directory up to date using a “heartbeat” approach, in which a device sends a message on a regular basis, and devices that don’t reregister themselves are considered dead nodes. RTI devices cost \$7.50 (10,000).

The problem with using active discovery, however, is that you need the ability to broadcast to devices. Broadcasting across the entire Internet simply isn’t an option (firewalls can be so bothersome), so SNMP use is limited to a closed, controlled network. Broadcasting also doesn’t reach devices that do not have persistent connections. In these cases, devices must register themselves.

The most effective registration is automatic and requires no human intervention, so it is not subject to human error. For example, requiring the purchaser of a photo frame to call in and register the frame increases the probability that the user won’t understand how to use the device and will return it. Instead, the user could connect the device (via Ethernet, phone, or other method) and the device, upon powering up for the first time, could call a “well-known” site to both register and obtain a device ID as well as instructions for how to make subsequent connections (for example, using a local number rather than a costly toll-free number or determining the location of an appropriate remote manager). At this

time, or during any other exchange, the device can download an upgrade. Note that both the device and the server must be able to recover if the connection is disrupted during discovery.

#### HIERARCHY

Managing one device is a fairly straightforward proposition. The device can have a direct relationship with the management server, and a fair variety of off-the-shelf software is available for “one-offs” of this nature. For example, you can embed a Web server in the device that a person using a browser from anywhere in the world can use to access, monitor, and control the device. The advantage of using a Web server is that it formats data so that a person can read it. If you need to collect information from 100,000 devices, or even just 100, people won’t be able to manage the devices.

A network of this size requires automatic device management. A management, or back-office, server can query devices and aggregate the data for a person to view. The function of the Web server moves from being a part of the device to being part of the management server, and the device needs only to submit its data to the server. Here, the number of devices the server must eventually manage becomes an important factor. For example, if devices call in to the management server, the server can simultaneously communicate with only so many devices. At

some point, connecting to the server becomes a bottleneck.

For larger networks, it makes sense to break the network into a hierarchy of devices, remote managers, and management servers. Devices communicate with the remote managers, which act as a proxy for the server and data aggregators, and send information from many devices to the server. Several layers of remote managers could also exist. For example, a remote manager could be responsible for devices in a single location, such as a warehouse, and report directly to a remote manager that manages several remote managers in a region. Breaking up the network into a hierarchy makes it easier to arrange devices into logical groups rather than maintain a long and incomprehensible list of nodes. Note that adding remote managers adds points of failure to the network. (If a remote manager gets knocked out, you lose contact with all the devices it manages.) Therefore, adding redundancy—in which a device can connect to a second remote manager if its primary one fails—becomes a consideration.

One advantage of remote managers is that they can communicate with legacy devices, and they can bridge devices (for example, sensors or two merging networks that use different technologies). Remote managers also enable you to create less intelligent nodes, in which the remote manager makes the decisions, or

## FIREWALLS

Just because you have an Ethernet connection and a LAN to hook into doesn’t mean you have your connectivity problem licked. If you’re on someone else’s network, you’re also behind their firewall. Even home networks (either through routers or gateways) support firewalls.

Firewalls create two kinds of problems. First, firewalls block unrequested incoming traffic. For an external server to initiate an exchange with a device, the network administrator must expose the device to the Internet, which means opening up the firewall. Consider a vending machine placed in a building and connected to the LAN. Such exposure requires action

on the part of the LAN’s network administrator, whose reliability you can’t trust.

More importantly, few, if any, network administrators are going to tolerate an exposed device that an outside company maintains. Therefore, only the device (and not the server) can initiate an exchange. Second, the firewall may prevent the device from accessing the Internet. However, if the LAN allows PCs to access the Internet, then there’s probably no issue in allowing the vending machine to send information out and receive replies. In any case, you want to keep the amount of configuration by the network administrator to a minimum.

For those applications in which an external server must be able to penetrate the firewall, a good compromise is to configure one remote manager that can penetrate the firewall and access the device. Using a two-stage approach, the external server talks to the remote manager, which can then penetrate the firewall and initiate an exchange with the device. In other words, instead of exposing the device (which lets any computer that knows the name of the device behind the firewall), only a single, known remote manager has permission to penetrate the firewall, and it can talk only to the one device.

If you can’t get permission to

penetrate the firewall, the server can initiate exchanges if you employ a periodic query from the device. On a regular basis—but not so regular that the network administrator notices or cares—the device initiates an exchange with the server. If the server wants something, then it replies, and the device goes into slave mode. Otherwise, the server declines the request. In this way, the server can initiate an exchange with latency no greater than the frequency with which the device queries it.

For devices outside a firewall, you might consider adding one to them to protect them from attacks.

use lighter weight protocols than TCP/IP, in which the remote manager acts as a bridge or gateway between the protocol and TCP/IP. For example, emWare offers emNet, a low-level protocol that requires only 2k- to 4k-word ROM and hundreds of words of RAM on an 8-bit processor. The cost of emNet is a couple of dimes per device. Devices using emNet can communicate with the \$20 emGateway, which bridges emNet to TCP/IP. Less intelligent nodes make sense for cost-sensitive applications or applications in which little decision-making takes place.

Sometimes hierarchies get in the way. For example, in a door application, a card reader that communicates with a server to tell the deadbolt to open creates a possible point of failure, unlike a card reader (a device that creates information) that communicates peer-to-peer with the deadbolt (a device that consumes information). Also, with several layers of hierarchy, it can take a while to exchange messages. For example, checking whether a card key is valid could take a few seconds. In such a case, keeping valid card-key data directly in the card reader makes

sense. However, you still need to be able to update the card reader's valid list through the hierarchy in a timely fashion. Another example involves two vending machines in the same building. If one machine runs out of an item, it has no easy way to check whether the other machine still has it. In this case, a peer-to-peer connection between the machines would enable the first machine to tell the customer that the item is available at the second machine. Additionally, some information may be lost in the aggregation process; if a remote manager clumps data, access to the status of individual devices is lost. If the remote manager has enough room to store individual device data, however, the management server can query the remote manager to flesh out the aggregated data if necessary. Remote managers also need to keep and maintain a directory of devices they are responsible for.

Note that at the top of your hierarchy, and at each of the levels, you're going to need management software. One advantage to

using standards such as TCP/IP, XML, and SNMP is that there are standard network-management tools you can use to aggregate, monitor, and act on data. Several companies offer such tools. Some companies, such as PlanetWeb, will also let you use their servers so neither you nor your customer has to set up and manage a server. You'll pay a premium for this service, and you'll have to make sure the company doesn't overprovision its server by supporting too many other customers.

Don't overlook or underestimate the upfront costs of developing and managing a management server. Yet once you have all the pieces together (connectivity, data exchange, and remote server), you've created an infrastructure that you can pay for and maintain with the savings that remote management enables. Then you can add all those pie-in-the-sky features, such as streaming content and e-commerce, with only an incremental cost and at significantly less risk. □

