FEATURE

# Control Networks and Interoperability

Interoperable systems for building automation and control applications are easier than ever to implement. The key is the appropriate use of standards that have been specifically designed for the tasks at hand.

by H. MICHAEL NEWMAN
Manager
Utilities Computer Section, Cornell University

*H. Michael Newman is the manager of the Utilities Computer Section at Cornell University in Ithaca, N.Y. He is a past chairman of ASHRAE's SSPC 135, which is responsible for the BACnet standard.*

BACnet, TCP/IP, XML, Java, HTTP, OPC, CORBA, LON, Ethernet, MS/TP—the buzzwords go on and on! But what do they really mean and how do they affect building automation and control networks and the ability of the equipment on these networks to work together in a meaningful way—to "interoperate?" The answer is that the technologies by themselves guarantee very little; it is how they are applied that counts. First and foremost, there have to be standards and they have to be used in reasonable ways.

I vividly remember the press conference in 1987 held to announce that the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) was forming a committee to develop a standard protocol for building automation and control, a standard we now know as BACnet. I had spent about 15 min explaining the project and answering various questions when a somewhat impatient spectator stood up in the back of the room and said, "I don't see why we need this standard. We already have ASCII and RS-232!" While both of these standards were, and are, important, much more is needed for devices to be able to communicate. To see why, let's look at the meaning of the terms.

ASCII is simply a set of numeric codes for representing letters and other common characters, while RS-232 specifies a way to hook computers and modems together. When we consider these definitions together, the spectator's statement essentially translated as, "We don't need a standard language for communication because we already have an alphabet and a way to get the letters from one place to another!"

However, communication requires far more than just an alphabet. The attribute that makes "Bad" significantly different from "Boot" is not the fact that one word has three letters and the other has four, but that the two words have different meanings. Moreover, we can only understand the meanings if we know the language from which the letter combinations derive. For example, the sequences of letters in "Bad" and "Boot" in German mean "Bath" and "Boat" in English. The same is true for machines: they have to know the meaning of the messages they receive to act properly upon them.

The argument that an alphabet and an interface connection are enough to enable communication has "evolved" to the equally senseless assertion, heard surprisingly often these days, that standards such as BACnet are unnecessary because "we now have XML and the Web."

The problem, again, is that these technologies, by themselves, are far from adequate to achieve communication, let alone interoperability. The eXtensible Markup Language (XML) is simply a way to represent data in a structured form that allows the component parts to be easily broken up or "parsed" using widely available software. Yet without knowing what each of the parts means, the parsed data is useless. The World Wide Web, one of the great technological developments of our

time, is intended for humans who need to access data from machines. The idea of one machine calling up another to request a Web page containing some piece of information, rather than asking for the information directly using a standard protocol, is simply misguided.

## INTEROPERABILITY DEFINED

Interoperability is the ability of equipment to work together. In a networked world, this implies the ability of the equipment to communicate mutually. It doesn't really matter what the purpose of the equipment is. The "interoperation" can be between different manufacturers' control equipment, different versions of control equipment from the same manufacturer, equipment intended for different purposes such as fire alarm, lighting control, intrusion detection, and HVAC—the point is the equipment has to be able to communicate to interoperate.

How is this communication accomplished? It's simple, at least in concept. The communicating devices have to use a common set of communication rules—a common "protocol." While I would argue that the use of a standard protocol is by far the best approach, interoperability, strictly speaking, only requires that the devices we want to work together use the same protocol. Of course we can reasonably expect to find more devices that use the same protocol if it is a standard protocol.

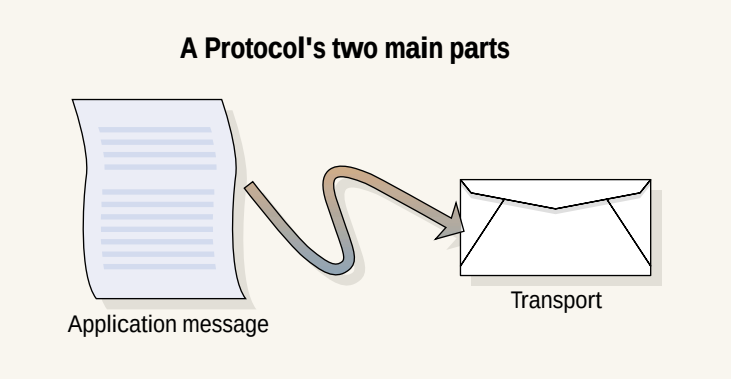


*FIGURE 1. A "protocol" can be thought of as having two fundamental parts: a message and a means to transport it.*

## UNDERSTANDING "PROTOCOL"

Essentially, a protocol consists of two parts: a message, and a way to get the message where it needs to go (Figure 1). The content and format of the message depend on the cooperating application. For example, if the application involves transferring files, then both machines have to understand information related to file size, file type, position of the data in the file, and so on. If the application involves HVAC, the machines have to understand binary and analog inputs, outputs, values, schedules, alarms, etc. For every type of interoperable application, there has to be a corresponding application message protocol that defines the semantics, or the meaning of the

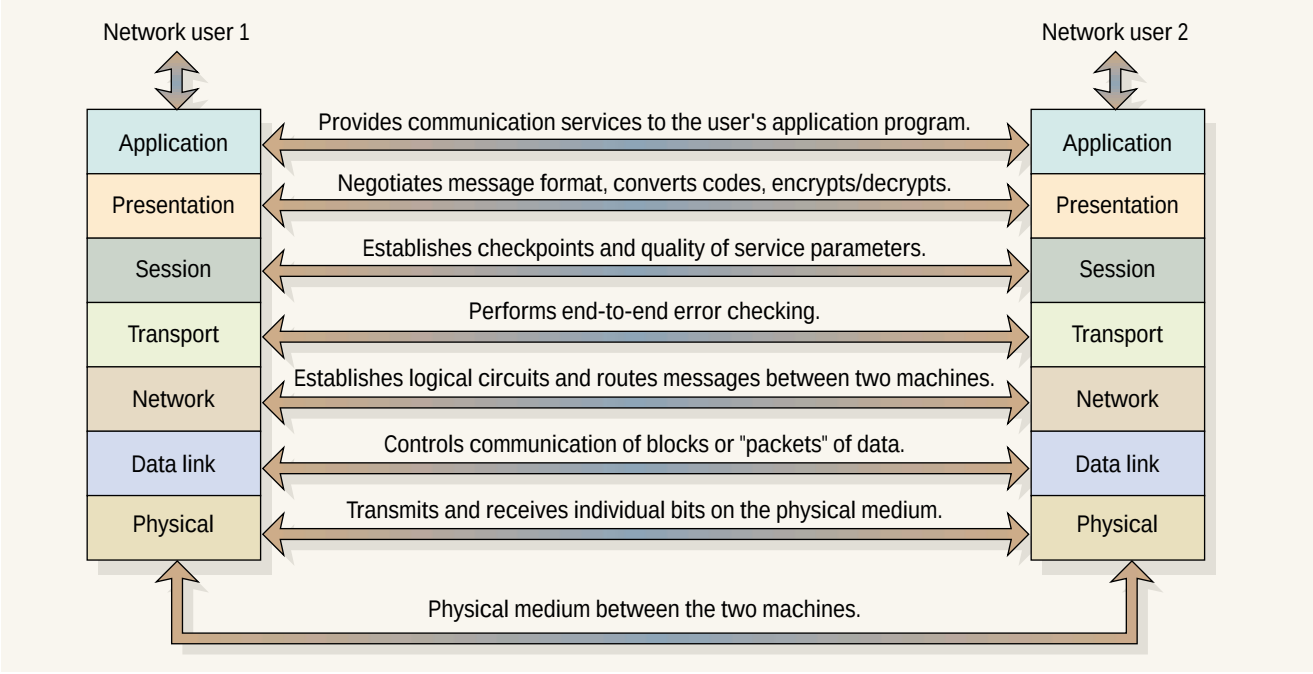


*FIGURE 2. The ISO/OSI Seven-Layer Model arranges communication functions into seven groups or "layers." Each layer provides services locally to the layer above while communicating with its peer layer in the remote device. Protocols that implement the model need only select the functions needed for the application at hand.*

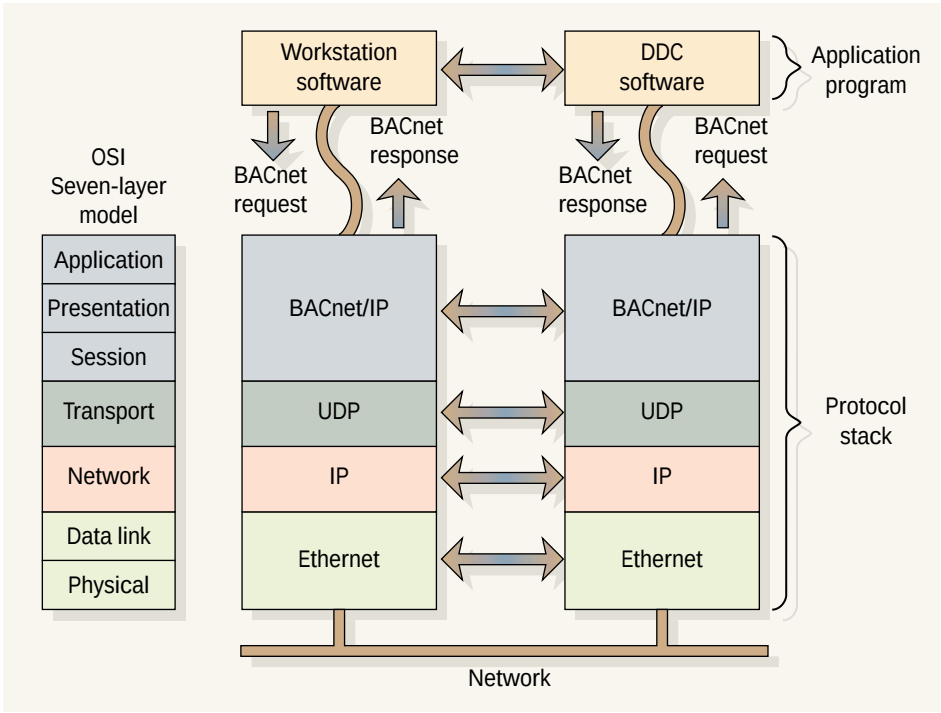


***FIGURE 3. A workstation and digital controller that use BACnet/IP as their native application protocol via an Ethernet LAN would have this protocol stack implementation.***

data, and the syntax, or the method of formatting it.

The rules governing the "transport" component of a protocol cover a vast multitude of issues that must be agreed upon. Among these are:

- Physical issues, such as the type of cable, connectors, electrical or optical signaling
- The type of "addressing" that uniquely identifies each machine on the network
- Error checking of various types
- Data security
- Data compression
- Rules related to interconnecting multiple physical networks; and many more.

## ISO/OSI SEVEN-LAYER MODEL

Fortunately, these transport issues have been around since the beginning of networking and have received intense attention from networking specialists. One result is the Open Systems Interconnection Basic Reference Model (BRM), developed within the International Organization for Standardization (ISO) in 1984, and published as Standard ISO 7498. The BRM takes all of the data communication issues and arranges them in seven distinct groups or "layers" (Figure 2).

The BRM model is intended to provide a conceptual framework that protocol designers can use to implement communication technologies. The resultant set of protocols is called a "protocol stack" because the protocols are hierarchically arranged, one on "top" of the other. The layers can also be useful for helping to focus a discussion of the various communication issues. Here are several important points about the model:

- Each layer provides communications services to the layer immediately above. As long as the interface between layers is preserved, specific implementations can be replaced without affecting the overall communication.
- Each layer communicates with its peer in the destination machine via a protocol. Also, each layer adds protocol control information (PCI), which adds to the overhead of each implemented layer.
- The functions of each layer can be implemented in separate, distinct protocols, but don't have to be. It is entirely possible to implement the functions of several layers in a single protocol. Conversely, several protocols may implement a function defined for a single layer, e.g., message segmentation and reassembly (a method of breaking up long messages and putting them back together again in the right order), although this would be inefficient.
- Not all the functions of all the layers are always needed. Many control system protocols only implement the Physical, Data Link, and Application layers. Some add a Network layer protocol if wide-area communication or communication between different local area network (LAN) types is required.

## BUILDING AUTOMATION EXAMPLE

With these points in mind, let's look at a couple of examples that show how the theory relates to the real world. I will use two application layer protocols—BACnet and HTTP—to make the points. I selected BACnet because it is the only protocol in the world that has been developed exclusively to promote interoperability in building automation and controls applications. I picked HTTP because it is at the core of the World Wide Web and I want to show how building control protocols and the Web can work together.

Look at Figure 3. The two sets of vertical blocks represent application and communication functions in two networked machines. The blocks on the left are in a "workstation" and the blocks on the right are in a "controller." At the application level, a workstation program (e.g., a graphic display generator) wants to obtain the value of a "point" known to the direct digital controller (DDC), say a temperature value, so it can display it to the operator. The horizontal bi-directional arrow between the top blocks shows that, conceptually, the workstation and DDC software are communicating directly with each other.

In reality, the workstation software accesses the communication protocol stack by invoking a procedure that is provided by the protocol stack's applica-

tion programming interface (API). The vertical arrow designated "BACnet Request" indicates this interaction. The request actually contains many parameters including the address of the destination machine (or information that can be used to figure it out, such as its "name"), the service requested (in this case, ReadProperty, the BACnet message that asks for the value of a "property" of a BACnet "object"), and other data necessary for the message to be assembled. This information is encoded by the BACnet/IP software into a bunch of zeros and ones, using the rules in the BACnet standard and passed down to the User Datagram Protocol (UDP) software. Again, the horizontal two-headed arrow shows the BACnet/IP implementations in the two machines are conceptually communicating between each other.

The UDP simply sends blocks of data (called datagrams) to a destination. The protocol control information (PCI) that is added to the data, in this case the BACnet/IP message, is just a source and destination port number (a numeric value that indicates the protocol immediately above in the stack), and a number indicating the length (in 8-bit bytes), of the message data.

Unlike the Transmission Control Protocol (TCP) that we will discuss shortly, UDP is "connectionless," meaning there is no concept of a sequence of related data messages or even of an acknowledgment for the current message. For this reason, UDP is considered "unreliable." The datagrams are simply sent out and if they arrive at their destination, great, if not, too bad!

If UPD isn't reliable, then why is it so widely used? Because other protocols in the stack can make up for its deficiencies. This is an important point because it illustrates one of the basic concepts in this whole protocol stacking business: When designing a stack, the trick is to ensure that all necessary capabilities are provided, but not more often than necessary. For example, if every protocol in the stack needed to ensure the end-to-end reliability of its data, the network would soon get bogged down in superfluous traffic because each protocol would be sending acknowledgments of its own packets, all of which contain the same embedded application message. In our example, the BACnet protocol provides the needed overall reliability by requiring acknowledgments from the destination's BACnet implementation. If no acknowledgment is received, the protocol retries the transmission or takes other actions such as notifying an operator of the problem.

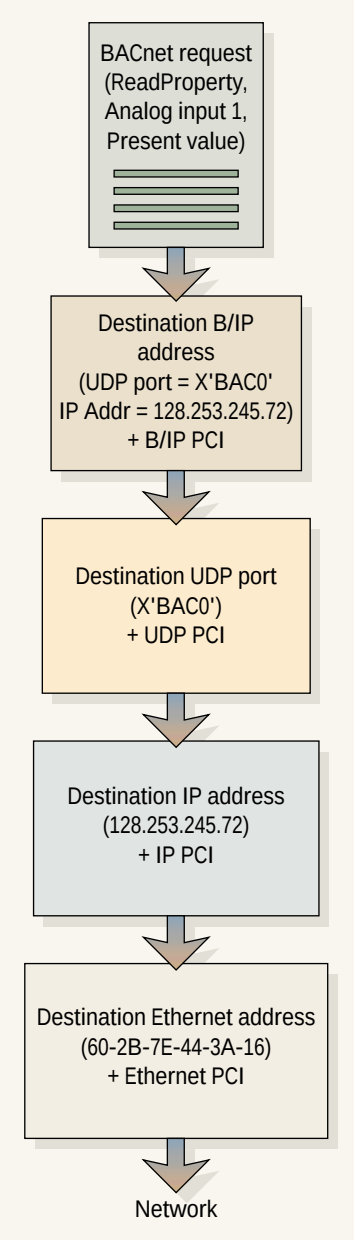


***FIGURE 4. Each protocol in the stack can be thought of as an envelope whose data comes from the protocol above. Notice how the envelopes get bigger as protocol control information (PCI) is added to the envelope from above. Each "envelope" actually consists of a string of bytes containing the PCI and the data payload. At the destination, a BACnet Response is formed that contains, hopefully, the Present Value of Analog Input 1 and is returned in a similar manner.***

Continuing down the stack, the UDP "packet" (the BACnet/IP message with the UDP port numbers and length) is passed as data along with an IP address and other PCI to the IP protocol, which in turn sends it to the Ethernet driver, which actually puts it out on the wire in accordance with Ethernet's rules. At the destination, the reverse sequence is followed until the message arrives and is processed by the DDC application, which generates a BACnet response that is returned to the requester in a similar fashion (Figure 4).

## WEB BROWSING EXAMPLE

A Web browsing application works in a completely analogous way (Figure 5). The Web browser sends a request for a Web page to the HTTP protocol implementation via its API. HTTP then builds an application message (Figure 1) that consists of a command, such as "get," followed by a Web page address (e.g., *www.hpac.com*). The latest version of HTTP (1.1) also supports a large number of command qualifiers. An example is "If-Modified-Since" followed by a date and time. This is sent if a version of the Web page has already been found in the local machine's "cache" or archive to reduce traffic on the network if the page has not been changed since the last time it was accessed.

Once formed, the HTTP message is passed to TCP. TCP, in contrast to UDP, is "connection-oriented." This means that

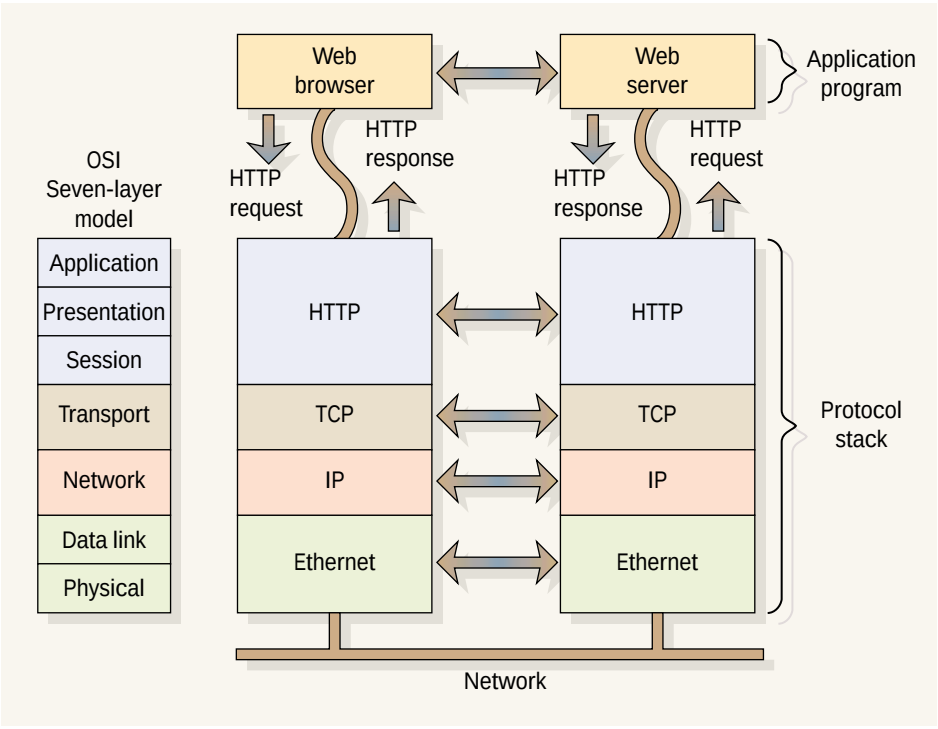


***FIGURE 5. A Web browser communicating with a Web server on an office LAN typically uses these protocols. For wide-area communication, the browser and server would be connected by the infrastructure of the Internet by means of routers attached to their individual LANs. The browser could also have data link and physical layer protocols that make use of modem communications instead of Ethernet.***

each message has a "sequence number" added to it so that a data stream whose length exceeds that of a single network packet can be broken into pieces and reassembled later on. This can easily happen with Web pages, where a single embedded image might be 100 kilobytes long but is being passed over Ethernet, whose maximum packet size is only about 1.5 kilobytes.

To achieve this, a link to the remote peer TCP protocol is established by exchanging several messages to synchronize sequence numbers and to negotiate other communication parameters needed by TCP. As the packets are transmitted, the receiving TCP protocol periodically acknowledges the receipt of the packets by sending a message to the sender that indicates the sequence number of the most recently received packet. This way, the sender can be assured that all packets up to that number have been correctly received.

Nevertheless, establishing the connection, acknowledging the packets, and terminating the connection after the data have been transferred take time and bandwidth. Thus, the fact that TCP is "reliable" comes at a cost and does not always justify its use if other protocols in the stack are available to ensure the communication's integrity.

Again, the TCP packet with its HTTP payload is passed down to IP, from there to Ethernet and, eventually, to its destination where, hopefully, the requested Web page is located, formatted according to HTTP

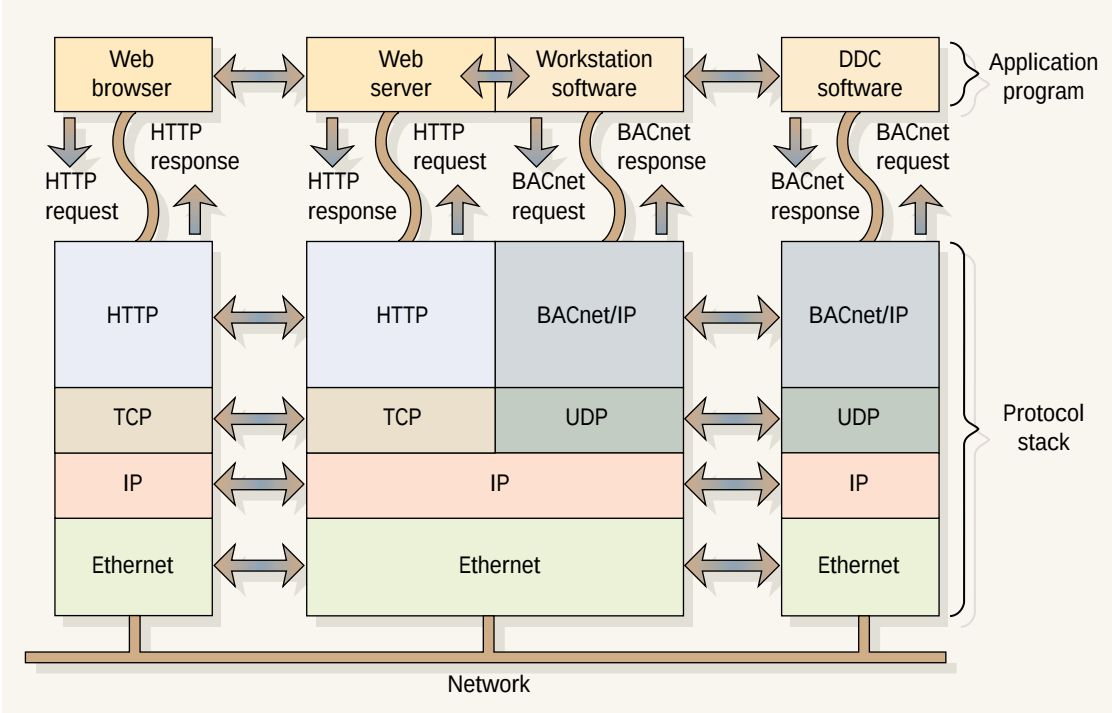


***FIGURE 6. A Web browser that accesses an appropriately programmed Web server can perform the same kind of functions as the dedicated workstation in Figure 3.***

rules, and begins its journey back to the requester, very likely as a sequence of numbered TCP packets.

## WEB BROWSERS AS WORKSTATIONS

What about using browser technology to implement a control system workstation? This is a great concept, at least in principle, because it means that virtually any PC with an Internet connection can provide workstation capability without any software beyond a standard, low-cost Web browser. However, there is no free lunch—the work still has to get done somewhere! There has to be a Web server that also implements the desired workstation functionality (alarming, trending, schedule maintenance, graphic displays, etc.), and it has to be able to communicate with the control system equipment in the field.

Figure 6 shows how this works in terms of protocols. The middle device is the Web server/control system gateway. Note how the upper layer protocols happily coexist and make use of the common IP and Ethernet capabilities. To see how the concept works in terms of hardware, see Figure 7.

The most important thing to understand is the interaction between the "Web Server"/"Workstation Software" blocks at the top-middle of Figure 6. The Web server has received, let's say, a request for a graphic containing some field I/O point value. This request must be passed to the workstation software which in turn builds (in this case) a BACnet/IP request, sends it out, gets the response, and passes it to the Web server. Here the data must be converted to a printable character string in HTML format and then, finally, returned to the Web browser for display. The software that links the server and workstation capabilities will depend on the implementation and will almost certainly be proprietary. This is what you pay for when you buy a Web server capable of talking with a control system.

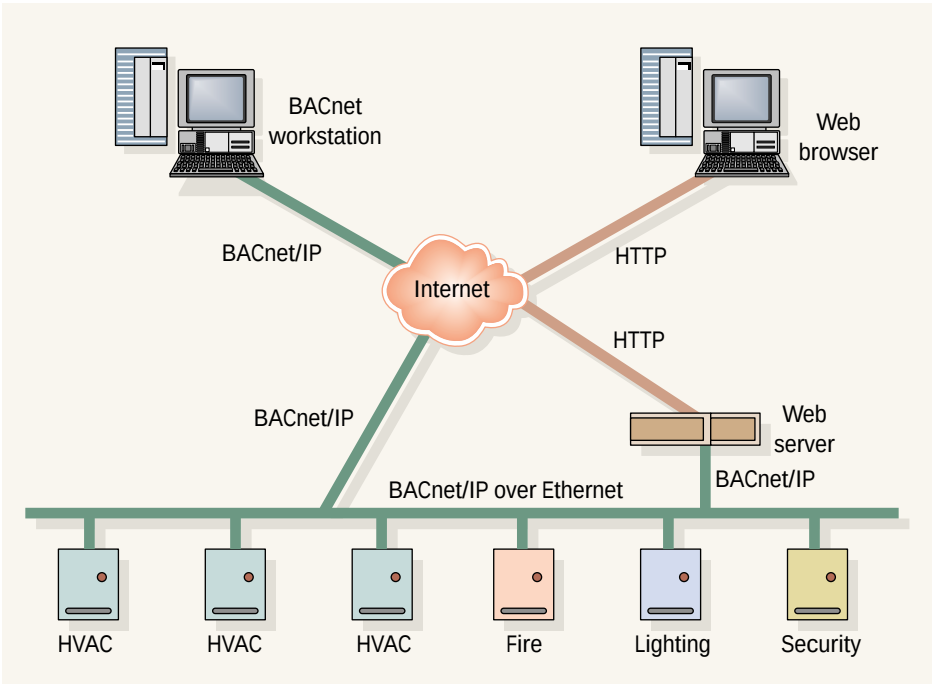


***FIGURE 7. Putting it all together: A PC with dedicated workstation software and another PC with only a Web browser can both perform "workstation" functions but the browser requires a specialized Web server that does the work otherwise performed in the dedicated PC.***

## DESIGNING INTEROPERABLE SYSTEMS

Here are some steps that can lead to achieving interoperable systems:

1. Figure out exactly which systems must communicate and what data needs to be communicated; define the application.

2. Select equipment that performs the desired functions and supports a common, preferably standard, application protocol.

3. For equipment that does not support the common protocol directly, determine if gateways are available between the equipment's protocol and your selected common protocol. If no gateways exist, it may be possible to provide some additional hardware in the form of sensors or relays to equipment that is networked to achieve the needed monitoring and/or control functions.

4. Determine your operator-machine interface needs. Are workstations needed and what capabilities must be available? Is remote, off-site access required? If so, is a Web server/gateway available that can communicate to the field equipment using your selected common protocol?

5. Finally, make sure you, your system designer, or controls contractor understands the network architecture well enough to lay out the ground rules for all participants to follow in setting up their equipment. These rules apply to such things as network and device numbering, assignment of alarm priorities, assignment of command priorities for different applications (energy management, smoke purge, night ventilation, lighting control, etc.), conventions for setting up trend logs, schedules, and so on. (If you choose BACnet, you can refer to the National Institute of Standards and Technology's Internal Report 6392 [NISTIR 6392] GSA Guide to Specifying Interoperable Building and Control Systems Using ANSI/ASHRAE Standard 135-1995, BACnet. This guide was developed for the U.S. General Services Administration [GSA] to provide suggestions about how to approach setting up these ground rules. It is available from NIST at *www.nist.gov*, or from *www.bacnet.org*).

## IN CLOSING

At the beginning of this article, I mentioned several other acronyms including Java, CORBA, and OPC. These are yet other ways of skinning the communication cat. They all provide mechanisms for communicating with "objects" (structured collections of information) in ways that shield the application program from having to know the details of where the objects reside on the network or how they are to be accessed.

However, to reiterate, the work of defining the objects and setting up the protocol infrastructure—doing the work that the various standards bodies such as ASHRAE have been doing for years—must still be done. The notion that these and other new technologies are somehow the "holy grail" of data communications seems to ignore the unassailable fact that interoperability requires, first and foremost, agreement among the participants on which technology to use; almost any technology could, in principle, be made to work. The questions, then, are which one is best suited to the specific application at hand, and how do you get an industry to agree? The second question is much harder to answer than the first. NC

## GLOSSARY

ASCII. ANSI X3.4—1977, American Standard Code for Information Interchange.

API. Application Programming Interface. A set of functions or procedures that an application program can "call" to access the underlying communication protocol stack.

BACnet. ANSI/ASHRAE Standard 135, BACnet, A Data Communication Protocol for Building Automation and Control Networks (*www.bacnet.org*).

CORBA. Common Object Request Broker Architecture (*www.corba.org*).

Ethernet. ISO/IEC 8802-3. The most common high-performance peer-to-peer LAN protocol in use today.

HTML. Hypertext Markup Language. A text-based language for creating platform-independent information display pages. HTML is one of the cornerstones of the World Wide Web (*www.w3.org/MarkUp*).

HTTP. Hypertext Transfer Protocol. The protocol used by the World Wide Web to request and receive data in the form of HTML pages (*www.w3.org/Protocols*).

I/O. Input/Output. Connections between a computer and sensors and actuators.

IP. The Internet Protocol. A network layer protocol originally created by the Defense Advanced Research Project Agency to facilitate data communication between the U.S. Defense Department and defense contractors, including universities and manufacturers (*www.ietf.org*).

Java. A programming language often used to write functional extensions to Web browsers.

LAN. Local Area Network. A network in which all devices can communicate directly without going through intervening routers.

LON. Local Operating Network. A proprietary hardware and software technology that can be used for building control and other suitable device-level applications.

MS/TP. Master-Slave/Token-Passing LAN. One of the data link layers created specifically for use with BACnet messages.

OPC. Object Linking and Embedding (OLE) for Process Control (*www.opcfoundation.org*).

OSI. Open Systems Interconnection. A set of ISO standards including the Basic Reference Model (BRM).

PCI. Protocol Control Information. Parameters such as address, length, hop count, segmentation parameters, quality of service, and protocol identifiers that are used by protocol implementations to process the accompanying data.

RS-232. "Recommended Standard" 232 of the Electronic Industries Association (now called EIA-232) that specifies the *Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange*.

TCP. Transmission Control Protocol. A connection-oriented protocol used to convey multiple related messages (e.g., file transfers, Web pages, etc.) (*www.ietf.org*).

UDP. User Datagram Protocol. A connectionless protocol usually used to convey single blocks of unrelated data or "datagrams." (*www.ietf.org*).

XML. eXtensible Markup Language (*www.w3.org/XML*).