The Johns Hopkins Address Registration System:
Anatomy of an Application

JHARS provides centralized administration of and self-signup for access to the Johns Hopkins network

By **Mark Cyzyk**

When Baltimore merchant Johns Hopkins in the late nineteenth century bequeathed seven million dollars to found a university and a hospital, he only hinted at the nature of the relationship he intended to exist between the two institutions.[1] With separate boards of trustees, the two institutions evolved and grew significantly independent of one another. This mutual autonomy continued late into the twentieth century.

Given the enormous growth and success of both the university and the medical institutions, a prime opportunity to streamline the governance and internal operations of the Johns Hopkins institutions was being missed. Hence, at the close of the century, the two institutions were tied more closely together when the president of the university was charged with simultaneously serving as chair of the Board of Trustees of Johns Hopkins Medicine (a corporate body consisting of the Medical School, the School of Public Health, the School of Nursing, the Johns Hopkins Hospital, and the Johns Hopkins Healthcare System) and the dean of the Medical School was charged with simultaneously serving as chief executive officer of Johns Hopkins Medicine.[2] With such organizational ties in place, Johns Hopkins was poised to streamline its internal operations.

One of the first steps toward streamlining internal operations was the appointment of a chief information officer (CIO) and a chief network officer (CNO) in 1999. For the first time, information technology and network operations between the campuses at Johns Hopkins would largely be merged. In particular, the networked resources of Johns Hopkins—its campus data and telecommunications networks, its Internet connections, and so forth—would be centrally managed and operated.

The Johns Hopkins Address Registration System (JHARS) is part of this ongoing process.

The Role of JHARS

The role of JHARS on the Johns Hopkins network is twofold:

1. It makes possible the self-signup of Johns Hopkins-affiliated faculty, staff, and students for access to the network across multiple campuses.

2. It enables the centralized administration of Dynamic Host Control Protocol (DHCP) and static Internet Protocol (IP) distribution throughout all participating Hopkins campuses and network segments.

The first role pulls together several separate, tedious processes previously required for gaining IP-based access to the Johns Hopkins network. These discrete processes are consolidated into a single, simple, quick signup procedure in JHARS.

Previously, access to network segments throughout the Johns Hopkins enterprise had to be individually requested and individually granted by a member of the IT staff. This process was time-consuming and error-prone. Efforts at maintaining a comprehensive database of IP assignments failed because a decentralized IT staff administered IP management.

A better solution uses DHCP to provide dynamic IP addressing to the large majority of desktops at Johns Hopkins and to maintain a constantly updated record of DHCP clients, their owners, and the media access control (MAC) addresses to which those clients are mapped.

The second role allows central administration of both leased DHCP addresses and statically assigned IP addresses. In the first case, most DHCP stations would be desktop systems; in the second, most static IPs would be granted to servers, printers, or IP-based medical equipment. In both cases, JHARS maintains detailed records of ownership and location for each piece of equipment. It also maintains a comprehensive change log for each owner, IP, and MAC in the system, enabling the generation of reports of historical data.

Finally, because JHARS is integrated with the central Johns Hopkins directory service, it can automatically delete DHCP clients if it is determined that an owner of a piece of equipment is no longer affiliated with Johns Hopkins. In such a case, the new owner merely has to undergo the two-minute signup process to get the piece of equipment back online.

Concept of Operation

The operation of JHARS was inspired by and designed with the previous work of those at Boston College in mind. In 1996 Boston College entered its EagleNet Activation System in the annual CAUSE (now EDUCAUSE) award for Application Best Practices.[3] Essentially, the EagleNet Activation System was among the first Web-based systems to enable client self-signup for access to a DHCP network.

JHARS expands on this service by providing for

- DHCP signup throughout multiple campuses,
- streamlining the processes surrounding static IP request and distribution,
- maintaining detailed records of equipment ownership and location (both current and historical), and
- doing so with current Web technologies and application development platforms.

The following steps describe the way JHARS works for self-signup to a DHCP-based network.

1. A client with a previously unregistered station, such as a laptop or desktop system, plugs into a live network port.

2. The network port is part of a segment of the network for which a DHCP server, in this case the Cisco Network Registrar,[4] is set up to service.

3. The network interface card (NIC) in the client station broadcasts to the nearest DHCP server, requesting an IP address.

4. The DHCP server receives the broadcast and notes the requesting MAC address.

5. If the MAC address of the client was not previously registered, the DHCP server responds by supplying a very short-term IP lease as well as a "spoof" DNS address. This response is enough to get the client station up and running on the network.

6. Once the requesting station is up and running on the network with a short-term IP lease, the spoof DNS entry ensures that all named requests resolve to the JHARS Web application.

7. The client boots up a Web browser.

8. The Web browser is directed to the JHARS Web application. Again, any attempt to go elsewhere will ultimately be foiled by the spoof DNS entry—all DNS name resolutions point to JHARS.

9. The first the client sees of the JHARS application is a log-on screen. The credentials supplied via this log-on screen are passed behind the scenes—using the Lightweight Directory Access Protocol (LDAP)—to the central Johns Hopkins directory service. If the credentials are confirmed, the client is logged on to the JHARS Web application, and the username from the directory service is used as a unique identifier. If the credentials do not pass the authentication process, the client is never logged on to the JHARS Web application and cannot proceed with the self-signup process.

10. Once a client is authenticated, the JHARS Web application—again behind the scenes—returns to the central directory service and gleans several pieces of information about the client. JHARS now knows all about the client qua person. This data, stored in the backend database, can be used in the future for running queries and reports.

11. Once logged on to the JHARS Web application, the client is presented with two functions:

- The client station can be registered and immediately granted DHCP access to the Johns Hopkins network, or
- the client can submit a request for a static IP, which is then vetted by the central IT staff before being granted or denied.

12. Upon choosing to sign up for DHCP access to the network, the client sees the screen shown in Figure 1. Here the client is asked to choose an Equipment Type representing the station being registered, such as workstation, laptop, or other device, and to indicate the operating system, if any, of the client station.

13. When the client clicks the Next button, two things happen. First, a record in the backend database is created, storing the submitted Equipment Type and Operating System values and linked to the client's user record. Second, the JHARS Web application does a lookup against all participating DHCP servers for the IP presented by the client Web browser in an effort to determine which DHCP server made the short-term lease of that IP and the MAC address to which the lease was granted. Once this is complete, the JHARS application knows several pieces of information about the client request:

- It knows, based on a directory lookup, quite a bit about the person making the request.

Figure 1

JHARS Opening Screen

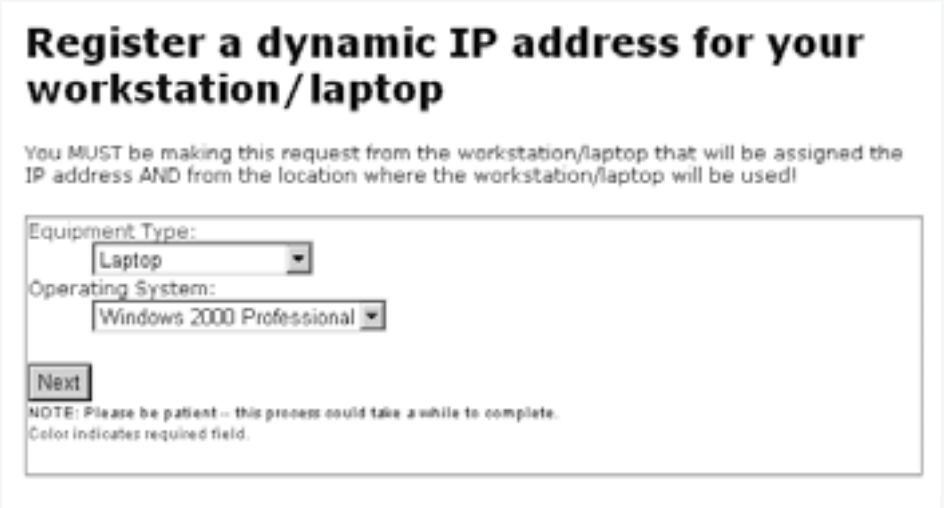


Figure 2

Information Collected by JHARS

- It knows against which DHCP server the request is being made. Based on this information, it further knows on which campus the piece of equipment is situated.
- It knows the MAC address of the client NIC.
- It knows the client-supplied Equipment Type and Operating System of the station being registered in the JHARS system.

Figure 2 provides an illustration of this screen.

14. Based on data contained in its backend lookup tables, the JHARS application can determine the range of buildings in which the client must be located on a particular campus, but it cannot determine the precise building, nor can it determine the room within a particular building. Because these are two useful things to know when it comes time to search the JHARS database and run reports, the client is explicitly asked to provide such information on this screen (see Figure 2).

15. When the client provides these two pieces of information and clicks the button to submit the final request, the JHARS application communicates with the previously determined active DHCP server for the request and registers the client MAC address for a long-term DHCP IP lease.[5] The DHCP client for this particular station is now set up.

16. Once this process is complete, the client is instructed to reboot the station.

17. Upon reboot, the client station again broadcasts to the nearest DHCP server and requests a lease of an IP address and the provision of DNS services.

18. This time, however, the DHCP server performs a lookup of the NIC's MAC address and notices that, indeed, the MAC is registered as a DHCP client.

19. The DHCP server then provides the client station with an IP address, and the client station is up and running on the network like any other DHCP client.

JHARS also enables IT staff and others to submit requests for static IP addresses. Figure 3 represents the form the client must fill out to make such a request.

Upon submission of this form, the client request enters a queue. Periodically, the IT staff check the queue and, via the JHARS Web application, grant or deny the request. The client is automatically e-mailed a message with the ultimate status of the request. If the request is denied, the IT staff member can note the reason. Moreover, the IT staff member can alter the request before granting it. For example, if the requested DNS entry is already taken or inappropriate, the IT staff member can change it, then grant the request based on the change.

Before an IP address or DNS entry is assigned, the JHARS database is checked to guard against duplicate assignments. In fact, the JHARS Web application will not allow duplicate assignments of IP addresses or DNS entries, thus enforcing good network management practices.

Administrative Features

Again, for both DHCP and static IP address requests and assignments, JHARS keeps track of all data gathered during these processes in its backend database. This continual log of the evolving content of the network enables the most powerful and useful administrative features of the JHARS application—the ability of IT staff to query and run reports on the data contained in the backend JHARS database. Figure 4 illustrates the screen that launches a search.

IT staff can search the backend database and produce reports on any of the variables that appear on the screen in Figure 4. For example, a report containing all the static IP assignments and DHCP clients in a particular building could be generated by simply choosing from the Building drop-down box and clicking the Search button. More importantly, any combination of these variables can be chosen as search criteria, thus enabling the IT staff person to generate finely detailed reports about stations on the network.

Suppose, for example, someone needed a list of all the DHCP clients in Gilman Hall running Mac OS X and owned by a member of the Department of Philosophy. The IT staff member would simply choose the corresponding values from the drop-down boxes on this screen and click the Search button, and the report would be generated. Such fine-grained searching and comprehensive reporting was not possible before the advent of this central repository of data on the Johns Hopkins network.

Authorized JHARS administrators can also alter records after their submission in an effort to keep the data in the backend database as up-to-date as possible. They do this by drilling down on a particular record to view its detail screen, as represented in Figure 5. Here, on a single convenient screen, everything that the JHARS system knows about a particular station can be found, including contact information about its owner.

From this screen, the IT administrator can update certain data elements of the record. For example, he can change the IP address, DNS entry, Building, Room, or Operating System. The administra-

Figure 3

JHARS Form to Request Static IP Address

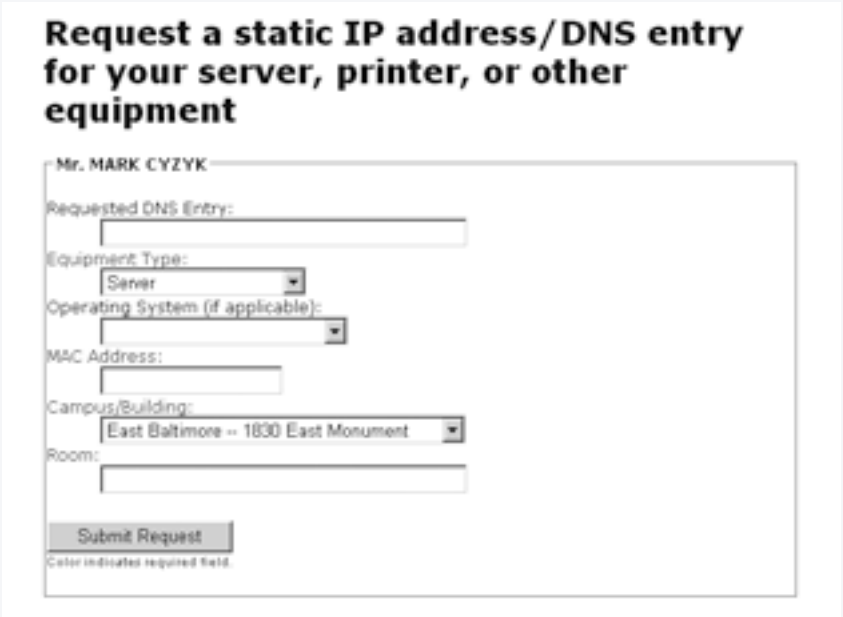


Figure 4

JHARS Search Functions Screen

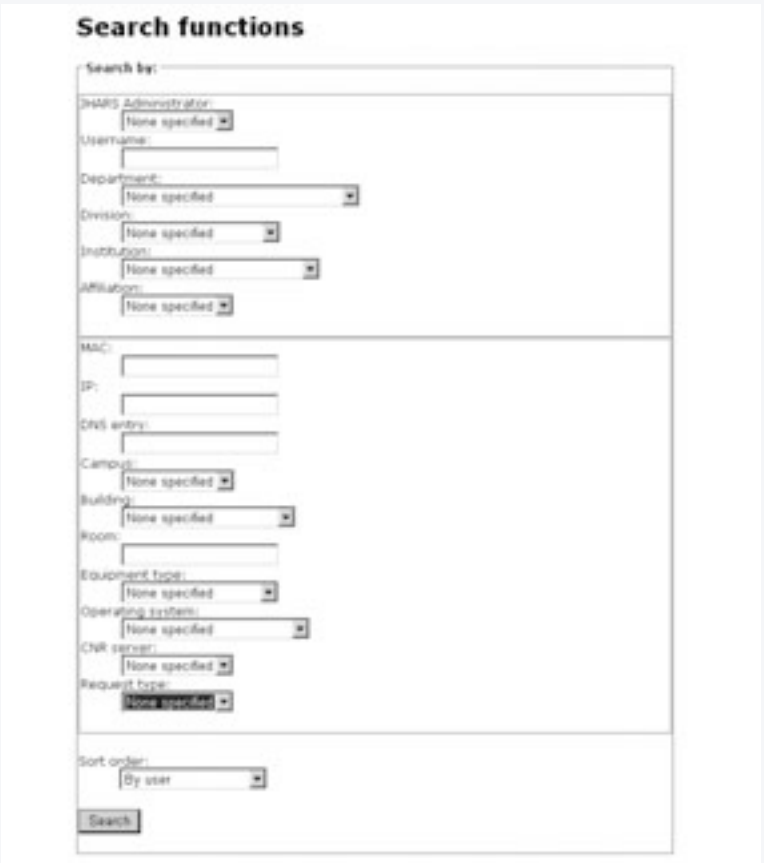
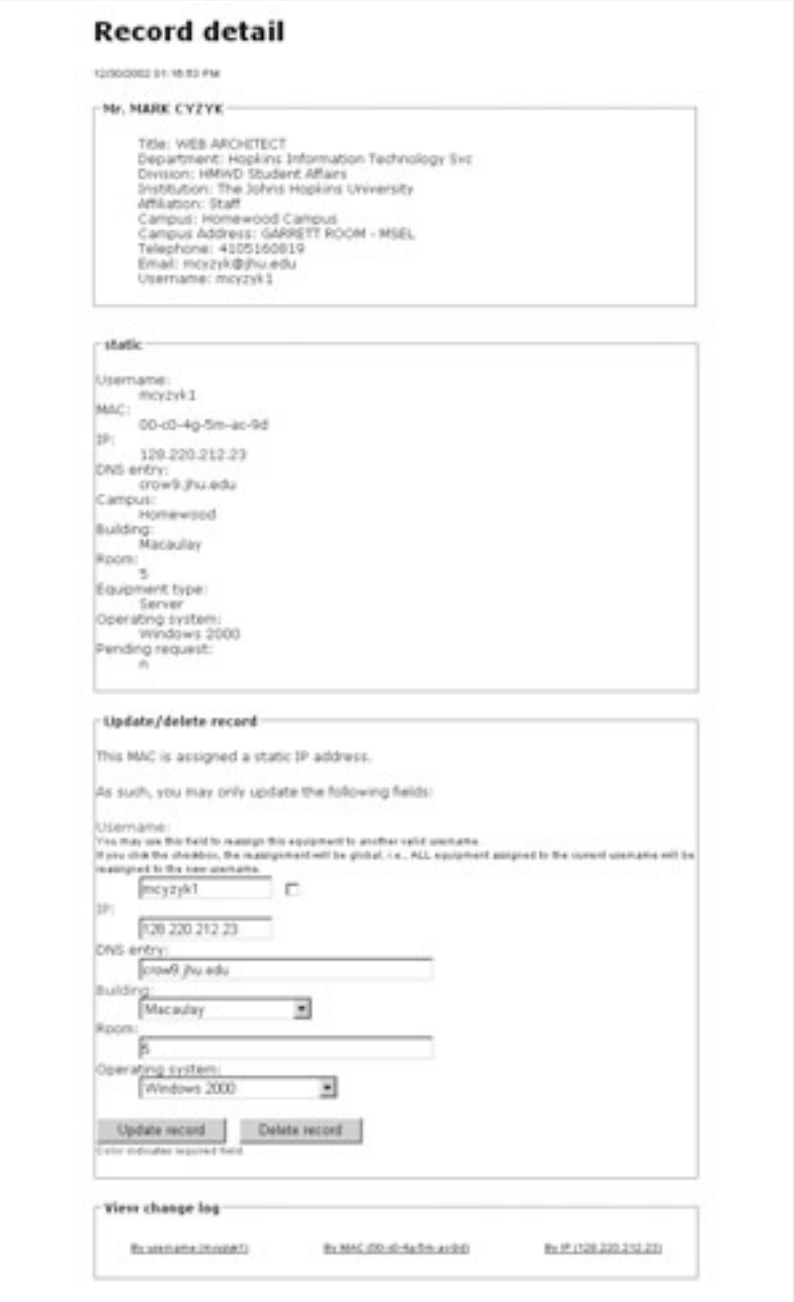


Figure 5

Changing a Database Record



tor can also change the ownership of the item by simply supplying a new username to which the station should be assigned. In the case where multiple stations need to be reassigned quickly and simultaneously—as when a lab manager or other person owning many stations leaves the organization—the IT administrator can click the checkbox and supply a new username to which all stations owned by the previous owner should, in bulk, be reassigned.

Finally, the record can be completely deleted from the JHARS application, or the Change Log can by viewed from this screen in one of three modes: username, MAC address, or IP address. The Change Log provides a comprehensive historical log of all changes made to the JHARS record.

Two other administrative features are worth mentioning: synchronization with the external DHCP servers, and synchronization with the external directory service. First, each authorized IT administrator of the JHARS system has the power to schedule a job whereby the JHARS application checks its database against a particular participating DHCP server. If a record does not have a corresponding record on the DHCP server, it will automatically be deleted from the JHARS database. In this way, if the local DHCP administrator decides to manually delete records from the DHCP server, corresponding records in the JHARS database will ultimately "fall off," leaving the DHCP and JHARS systems in a synchronized state.

Second, a scheduled job can be run—although only by the JHARS Superuser—to synchronize user records between the JHARS database and the central directory server. If a user record in JHARS is missing from the central directory service, this indicates that the person in question is no longer affiliated with the organization.[6] JHARS then informs the authorized IT administrators of the system via e-mail that a particular user record is missing from the directory service and that the associated equipment as listed in the JHARS database must be reassigned to a new username. Only after all equipment is reassigned will the user record be automatically deleted from the JHARS database. In this way, orphaned records do not become a problem, and assigned ownership of stations is kept current.

Problems, Concerns, and Solutions

The toolset used in the design and construction of this Web application and network service includes Macromedia's ColdFusion Web application development platform and the Cisco Network Registrar (CNR) DHCP server. The ColdFusion server was chosen as the Web application server because of its power, ease-of-use, and modest cost.[7] Also, Johns Hopkins maintains a large number of experienced ColdFusion developers on staff (somewhere between 50 and 200 at a given time). The CNR server is the DHCP server to which Johns

Hopkins is currently attempting to standardize across its many campuses. Specific problems, however, were encountered in the attempt to get these two technologies to interact.

The CNR documentation illustrates how to use the Perl programming language to connect to and manipulate records on the CNR server. The only other language mentioned for creating external applications to work with a CNR installation is TCL. Developers using other languages and development platforms are left to their own devices to get their applications to interact with the CNR server. The bulk of this interaction must occur between the Web application server and the CNR client, both installed on the same box.

At first, we found communications between ColdFusion 5.0 and the CNR client to be impossible. The ColdFusion Markup Language (CFML) contains a language construct, <CFEXECUTE>, similar to system callouts in other languages[8] that allows the Web application to call external executables on the server. The CNR client includes a simple command-line application programming interface (API) for communicating with it, and we thought we could simply use the <CFEXECUTE> tag from within ColdFusion 5.0 to execute the CNR client, passing in the appropriate arguments. However, initial experiments with this technique proved fruitless. <CFEXECUTE> was able to boot up the DOS batch file that executes the CNR client, but a response from the CNR client communicating the status of the requested operation was never passed back to the ColdFusion application server.

The next step was to take a close look at the CNR batch file to determine what exactly it calls. After examining this file (the "ncr.bat" file), we thought that calling the executable file directly (the "guitest.exe" file) as specified in the batch file might work. Unfortunately, while this method appeared to boot up the CNR client successfully, again, no information was communicated back to the ColdFusion application.

This failure led to a frantic search for other alternatives. Macromedia had, at the time, recently released the latest version of the ColdFusion application server, ColdFusion MX. This version represented a leap in evolution for the ColdFusion platform, having been completely rewritten to run on top of a Java application server. We hoped that something significant might have changed with the way the <CFEXECUTE> tag communicates with the underlying DOS shell between CF 5.0 and the Java-based CFMX, so made a final effort to call the guitest.exe executable from within a development version of CFMX. It worked! <CFEXECUTE> was able to boot up the CNR client, passing in the appropriate arguments via the command-line API, and to receive the appropriate response in return. (See the sidebar for a more detailed explanation of our solution.)

While the structure of communication between the ColdFusion server, CNR client, and CNR server works, it is less than optimal. In fact, it closely resembles the Common Gateway Interface (CGI) era of Web application development, when, for each Web browser request, the Web server would execute

Booting Up the CNR Client

The following code illustrates the process of calling the CNR client and passing in arguments for it to pass to the CNR server, querying whether the CNR server leased a particular IP address and, if so, what the MAC address of the station was:

```
<cfsavecontent variable="this CommandOutput">
   <cfexecute name="#theExecutable#" arguments="-e
   #getRegistry.NRCMDLocation# — -C #CNRClusterName# -N
   #CNRUsername# -P #thisCNRPassword # lease #currentIP# macaddr"
   timeout="10"></cfexecute>
</cfsavecontent>
```

The first thing to notice about this example is that the CFML <CFSAVECONTENT> tag is used to save to a variable any output generated by the <CFEXECUTE> call. The contents of this variable will be analyzed further down in the script and used to determine how to proceed. Next, the <CFEXECUTE> tag is called, first passing in a variable containing the name of the file to be executed (in this case, the guitest.exe executable), then passing in several command-line arguments. The #getRegistry.NRCMDLocation# variable contains a string representing the directory in which the guitest.exe and ncr.bat files are located. The #CNRClusterName# variable contains a string representing the name of the CNR server to which the client is being directed to connect. #CNRUsername# is the username of the administrative user on the CNR server. #thisCNRPassword# is the administrative password on the CNR server. The term "lease" indicates our wish to have the CNR client query the CNR server for current lease information. #currentIP# represents the IP address we are interested in knowing about, and "macaddr" ensures that the CNR server will return the MAC address currently leased to the passed-in IP address. Finally, the CFML "timeout" attribute is set, limiting the current operation of the <CFEXECUTE> to 10 seconds.

Lessons learned: For this system callout to work, the full path to the directory where both the ncr.bat and guitest.exe files are located must be in the PATH of the server. That directory path should not contain spaces. After a reboot to make absolutely certain the PATH environmental variable is properly set, the code will work.

a separate external process, the results of which would be passed back to the Web server, then back to the Web browser. This continual booting of external processes, not in-line with the main Web server process, was resource intensive and resulted in extremely sluggish response times. This is a problem with the way JHARS currently operates, too.

Another problem arises because the current CNR client is tied to a specific network, that is, it can only communicate with CNR servers residing on the same physical network. This works fine for most of the Johns Hopkins campuses, but if the CNR client could communicate with CNR servers on remote networks by, say, IP address instead of the current "CNR Cluster Name" method, then the remote networks of Johns Hopkins in Nanjing, China, and Bologna, Italy, could use the JHARS application for network management.

The best solution to these problems would be for Cisco, in the next iteration of their CNR Server product, to include a built-in client that would work as a Web Service over encrypted HTTP traffic. This would enable Web application developers to use any language capable of communicating with an accessible Web Service to write code that directly interacts with the CNR server over the Internet, negating the need for the local installation of a client executable and eliminating the requirement to be present on the same physical network as the server.[9] We hope that a future release of the CNR will include this key functionality.

Current Status

On December 6, 2002, JHARS went live with an overnight migration of approximately 250 stations from static IP addresses to DHCP. No problems were reported, and the migration, and its use of JHARS, was deemed immediately to be a success. Since that time, several other network segments on the Johns Hopkins network have gone live with JHARS. Plans for expansion of the use of this critical tool across yet other segments of the Johns Hopkins network are under way, most notably on the Johns Hopkins Bayview campus, at the Johns Hopkins Hospital, and at the Peabody Institute.

As of this writing (March 2003), reports are surfacing of LAN managers across Johns Hopkins using JHARS with great success and without training. They are embracing it and commenting on how easy it is to use. The fact that JHARS programmatically determines a station's MAC address, without user intervention, has eliminated the most error-prone part of the previous IP signup process: manual input of MAC addresses. Elimination of this problem has enabled the compilation of far more accurate records and, overall, simplified IP management.

Hodson Hall, a Johns Hopkins facility recently built with the express purpose of supporting information and instructional technology in the classroom, is now fully online with JHARS. Students plug laptops into Ethernet ports or configure their wireless cards for DHCP and are online in minutes. No problems have been noted.

These early reports are anecdotal, and no systematic survey of user satisfaction with the JHARS service has yet been undertaken. Nor has there been a rigorous effort to predict the long-term effects of JHARS on network management efficiencies and its probable impact on staffing. Still, the reception of JHARS has been overwhelmingly positive, and all indications are that it will continue to enable students, faculty, and staff to easily access the Johns Hopkins network. It also greatly streamlines and adds value to the network management processes required of central IT staff. e

Acknowledgments

I wish to gratefully acknowledge and thank the following individuals at Johns Hopkins for their direct and indirect contributions to the creation of the JHARS Web application: Eric Cronise, Brandon Lockett, Craig Ponton, Calvin Sproul, and Dean Zarriello.

Endnotes

1. Johns Hopkins did specifically note that the hospital was to "ultimately form a part of the medical school of [the] University." But he didn't explicitly assert his wishes as to the exact nature of the relationship. This was complicated further by the fact that each institution had its own board of trustees. See John C. French, *A History of the University Founded by Johns Hopkins* (Baltimore: The Johns Hopkins University Press, 1946), pp. 4–5.
2. The organizational chart for Johns Hopkins Medicine, clearly illustrating the interlocking governance structure of the university and Johns Hopkins Medicine, is posted on the Web at <http://www.hopkinsmedicine.org/facts/gover/org.html>.
3. See the Web at <http://www.educause.edu/awards/epit/96/96bp.html>. Also, in 1998, Cisco Systems released a white paper detailing the Boston College project. This paper is posted on the Web at <http://www.cisco.com/warp/public/cc/pd/nemnsw/nerr/profiles/bosco_cp.pdf>.
4. For current information about the Cisco Network Registrar product, see <http://www.cisco.com/warp/public/cc/pd/nemnsw/nerr/index.shtml>.
5. This server functions as the primary DHCP server in a particular CNR installation. For each primary server there is a corresponding secondary, backup server. At this point in the registration process, a DHCP client is generated on both the primary and secondary servers.
6. It is Johns Hopkins policy to delete, not merely flag, records from the central directory service when a person is no longer affiliated with the organization.
7. For an overview of the ColdFusion platform and a brief comparison with its main competitors, see Mark Cyzyk, "Script Junkie: ColdFusion Markup Language," *Web Techniques*, Vol. 5, No. 8, August 2000; on the Web at <http://www.newarchitectmag.com/archives/2000/08/junk/>.
8. For instance, the exec() and system() functions and the backtick (` `) operator in PHP and Perl.
9. Communications with Web Services are easy to write in CFML. Some sample code:

```
<cfinvoke
    webservice="https://someCNRserver.
    jhu.edu:8500"method="get MAC"
returnVariable="theMAC">
<cfinvokeargument name="ip"
    value="#ip#">
</cfinvoke>
```

In this case, a particular method of a Web Service is being invoked (the "getMAC" method), passing in an IP address as an argument. The result is saved in the #theMAC# variable.

Mark Cyzyk (mcyzyk@jhu.edu) is the Web Architect at the Johns Hopkins University in Baltimore, Maryland, and is currently in the doctoral program in Information Systems at the University of Maryland, Baltimore County (UMBC).