

RECEIVED

JLAB-ACC-97-31

NOV 19 1997

OSTP

Centrally Managed Name Resolution Schemes for EPICS

MASTER

Ding Jun

Institute of High Energy Physics of the Chinese Academy of Sciences, P.O. Box 918(7), Beijing, 100039  
P. R. China

David Bryan and William Watson

Thomas Jefferson National Accelerator Facility, MS12A2, 12000 Jefferson Ave., Newport News, VA 23606  
USA

### Abstract

The Experimental Physics and Industrial Control System (EPICS) uses a broadcast method to locate resources and controls distributed across control servers. There are many advantages offered by using a centrally managed name resolution method, in which resources are located using a repository. The suitability of DCE Directory Service as a name resolution method is explored, and results from a study involving DCE are discussed. An alternative nameserver method developed and in use at the Thomas Jefferson National Accelerator Facility (Jefferson Lab) is described and results of integrating this new method with existing EPICS utilities presented. The various methods discussed in the paper are compared.

### 1. Introduction

A key component of a large control system is the ability to access control points by name, independent of location. This process of name resolution and control point location is an important and critical piece of any large scale control system. EPICS is based on a client/server system, in which clients locate channels by broadcasting search requests to all of the Input/Output controllers (IOCs). This method is generally associated with LAN environments, and is an excellent mechanism for locating control points in a LAN environment. EPICS functions in a WAN environment by using lists of IP addresses, which can be either individual hosts or IP subnet masks, to which name resolution requests are sent. While this system offers the advantages of distributing the name resolution process and simplifying name data management, it is not without drawbacks. [1]

On large control systems or systems with a large number of hosts, this broadcast method requires a large number of broadcasts resulting in increased network load.

Additionally, each IOC must process every request, whether or not it is intended for that IOC. The time spent processing these requests is time that the IOC cannot spend on its core functionality of device control. By off-loading these functionalities to a centrally managed nameserver or servers, significant improvements in the speed of some key EPICS components may be realized.

This paper explores several approaches to using a centrally managed system. Results of a study conducted by Ding Jun and William Watson on the suitability of DCE, or

Distributed Computing Environment are presented. Results from the design and use of an alternative nameserver developed by David Bryan and William Watson are discussed.

### 2. Overview of DCE and DCE Test Environment

DCE was originally developed by OSF (Open Software Foundation) and is currently supported by many vendors, including Sun, HP, and IBM. DCE consists of several components which are designed to work closely together:

[2]

- DCE Threads: support the creation, management, and synchronization of multiple threads of control within a single process.
- DCE Remote Procedure Call (RPC): consists of a development tool and runtime service. The development tool includes a compiler for a language (IDL, Interface Development Tool) for developing applications following the client/server model. This code can be used to automatically generate code to transform procedure calls into network messages.
- DCE Directory Service: a service which maintains information about resources such as users, machines, and RPC-based applications within the distributed system. The information consists of the name of a resource and associated attributes, including the resource's location.
- DCE Distributed Time Service (DTS): provides synchronized time on the computers in a distributed computing environment.
- DCE Security Service: provides secure communications and controlled access to resources in the distributed system.
- DCE Distributed File Service (DFS): allows users to share files anywhere on the network, regardless of the file's physical location.

DCE Directory Service is used by the core DCE services and DCE applications to locate distributed, rapidly changing resources. This service is composed of three parts

- Cell Directory Service(CDS): stores names and attributes of resources located within a DCE cell.
- Global Directory Service(GDS) or DNS: used to look up a name outside of a local cell.
- Global Directory Agent: serves as an intermediary between a cell's CDS and the rest of the world.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

18

## **DISCLAIMER**

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

## **DISCLAIMER**

**Portions of this document may be illegible  
in electronic image products. Images are  
produced from the best available original  
document.**

The Directory Service uses an extended form of the client/server model. It consists of servers, clerks, and clients. A CDS server stores and maintains CDS names and handles requests to create, modify or look up data. Each host using CDS runs a CDS clerk to act as an intermediary between the server and client. The Directory Service also provides a programming interface, which was extensively used in the test code.

The test environment for DCE was set up at Jefferson Lab. A two machine cell consisting of one IBM and one HP workstation was established. The core DCE servers were placed on the IBM machine, while the HP functioned as a client. Since these two machines together formed a cell, only the CDS service was used.

The DCE application used for the test consisted of two test servers which have different resources but the same interface, and one test client. When the client sends a request by resource name, the CDS is consulted to determine which server has that resource. The correct server is then contacted and performs the required operation.

Client programs may search for resources in one of two ways. They may search by interface, such as when adding a resource. In such a case, the client doesn't care which server processes the request, as long as it has the correct interface. When deleting a resource or executing a command on a resource, the client will search by resource, as it must contact the specific server that manages that resource.

An interface to be shared between the client and the server was defined using IDL. This can then be used to generate stub code as an intermediary between the application code and the RPC runtime library. In client code, a remote procedure call appears to be a local call, because the stub code will handle these calls to execute the binding operation.

At boot time, the server location and other information is exported by the server initialization code. After the server is initialized, all remote procedure calls are executed in the server management code. Three operations (add/delete/show) can be executed on a resource by the server.

Once the code is compiled and the server initialized, another important consideration is access control. DCE has an important feature called ACL (Access Control List). These are used to protect resources in the distributed environment. In order to modify the data in the CDS, the server needs to have permission on that cell.

### 2.1 DCE Experiment and Results

Test code was written to measure the average time for DCE application clients to look up a resource. To eliminate network effects, the test client and server ran on the same machine as the CDS server. The number of resources in the CDS database scaled from 100 to 10000, and the average time for the clients to look up a name was calculated as the database size changed. Two cases were examined:

In the first case, a random name generator produced resource names to be used to populate the namespace. These random names consisted of capital letters (A-Z) and were between 8 and 16 characters long. The nameserver was populated, and 10% of the names were searched for. The average time per channel was calculated based on the searches for this fraction of the total database. This test was repeated with three different namespace sizes:

Table 1: DCE lookup time for random names

Namespace Size	Average Time (s)
100	.085
1000	.069
10000	.125

It can be seen from the table that when the namespace size is increased from 1000 to 10000, the lookup time (in seconds) approximately doubles. The increase in performance when moving from a 100 to 1000 name database is considered to be the result of the CDS cache, in which the results of searches for names are stored. This reduces the number of times that a client must go to the server for the same information.

In the second case, a sequential name generator generated names consisting of a prefix followed by a number between 1 and 5000. Again, multiple namespace sizes were used, and 100 channels were looked up in order to determine average lookup time.

Table 2: DCE lookup time for sequential names

Namespace Size	Average Time (s)
1000	.091
2000	.097
3000	.101
4000	.118
5000	.119

From this, it can be seen that the searching time does not appreciably change when the namespace size grows.

### 3. Overview of the CA Nameserver

Jefferson Lab is presently using a new nameserver, developed on site, to locate resources within the control system for the main accelerator. The control system consists of approximately 160,000 channels distributed across 70 control servers. The new nameserver maintains information for all these channels.

This nameserver is specifically designed to work with EPICS. Only information needed to locate EPICS channels is stored in this nameserver. At present, each channel has a pointer associated with it that references the data structure for the server which that channel resides on. Each server's data structure contains the IP address, port number, and CA (Channel Access - the network protocol used by EPICS) version number. [6]

EPICS utilities such as MEDM or BURT interface to the new nameserver using an adapter layer. This layer can be linked in at compile time, and intercepts the CA search commands. When the results are returned from the nameserver the adapter responds accordingly. If the nameserver found information about that channel, then the channel is connected directly, eliminating the broadcast search step. If the nameserver has no information for the channel, the existing EPICS broadcast method is employed. This method has the advantage of allowing any EPICS application using CA to be easily compiled to use the nameserver. The adapter locates the nameserver using an environment variable which defaults to a preset value. This allows test applications or small groups to work with servers other than the main nameserver.

The nameserver and the adapter communicate using the CLIP protocol as implemented by CDEV [7]. This protocol has been used for many applications at Jefferson Lab and has proven to be highly reliable. A lightweight interface was used to minimize the size of both the adapter and the nameserver. As channels are requested by the CA client, the channels are bundled in groups of 40 (or less if the application requires less) and sent to the nameserver. The nameserver searches its database and sends a corresponding number of replies. The reply is either the information needed for a connection (IP address, port, CA version number), or an explicit "Don't Know" reply. This allows the application to quickly revert to the original broadcast method if the nameserver has no information on that particular channel. Additionally, error handling routines ensure that the application will revert to the broadcast method if the server is unavailable.

The nameserver is implemented using a sparsely populated hash table. If the occupancy factor of the table exceeds 50%, the table is automatically resized to increase performance. A custom memory mapping algorithm is used, allowing the server to be much smaller than it would be using conventional C++ allocation (the new command). The executing nameserver, with 160,000 channels, uses approximately 8MB of space, including space for network buffers.

The nameserver can be populated either by loading a file or using CLIP packets sent by a registration program to the nameserver. At Jefferson Lab, each IOC writes a list of its channels at boot time. These channels are then uploaded to the nameserver by the registration program.

### 3.1 CA Nameserver Experiment and Results

To test the speed of the nameserver, a CA test program was created. This program would connect to between 1 and 23000 channels to test the speed of the nameserver. It was discovered that as the number of channels increased, the speed per channel increased, indicating that much of the time is in overhead related to connection management etc. Several trials were completed, and the average time per channel was calculated (all these trials used the operational database, with 160,000+ channels loaded)

Table 3: CA Nameserver random name lookup time

Channels requested	Average time/chan (s)
1	.22
100	.004
1000	.001
23000	$8 \times 10^{-4}$

As can be seen, this nameserver is able to resolve channels very rapidly, even with a very large namespace.

In use, the programs modified to use the nameserver appear to run identically to those that do not use the nameserver. If however, IOC load is monitored while running the old and new versions, requests by the new version do not generally affect IOC load. When requesting channels, the old method routinely causes the available processing power (CPU power not being used for other tasks) to drop by 10-15%, and for large request groups on heavily loaded IOCs, can cause 50% drops.

### 4. Conclusion

The three methods discussed here all have advantages and disadvantages. The original EPICS method of using broadcasts eliminates the need to maintain a centrally managed repository and does not require a separate executable to be run. On the other hand, the two centrally managed methods eliminate the load imposed on IOCs by search requests.

It is clear from the numbers presented that the CA Nameserver is able to locate channels more rapidly than DCE CDS. On average, DCE CDS is approximately 100 times slower than the CA Nameserver, and would most likely be more difficult to integrate with CA. On the other hand, DCE offers the advantage of having distributed, redundant servers, which could increase reliability.

### References

- [1]. D. Gurd (LANL), S. Lewis (LBL), B. McDowell (ANL), W. Watson (JLAB) "Distributed Enhancements to EPICS"
- [2]. "Introduction to OSF DCE" Open Software Foundation
- [3]. "DCE FAQ" ([www.osf.org/dce/faq-mauuney.html](http://www.osf.org/dce/faq-mauuney.html))
- [4]. J. Shirley, W. Hu, I. Magid "Guide to writing DCE Applications" (O'Reilly & Associates, Inc., 1994)
- [5]. W. Rosenberry, D. Kenney, G. Fisher "Understanding DCE" (O'Reilly & Associates, Inc., 1992)
- [6]. J. Hill, "EPICS R3.12 Channel Access Reference Manual" (Online publication, 1995)
- [7]. W. Watson, J. Chen, D. Wu, W. Akers "CDEV Reference Guide", Jefferson Lab, ([www.jlab.org/cdev/doc\\_1.5/cdevReference.html](http://www.jlab.org/cdev/doc_1.5/cdevReference.html))