ISSN 2277-2685
IJESR/Oct. 2016/ Vol-6/Issue-4/1-11
Ram Joshi [*1] *et. al.*, / Journal Engineering & Science Research

A NETWORK-BASED DAC OPTIMIZATION PROTOTYPE SOFTWARE

Ram Joshi [*1], Mano Singh[2]
[*1] PG Student, Information Technology, Shantilal Shah Engineering College, Bhavnagar, India
[2]Engineer-SE, ICRH-RF Division, Institute for Plasma Research, Gandhinagar, India

ABSTRACT

For the purpose of remote control of heating experiments, a prototype client of an EPICS-based Data Acquisition Control system (DAC) is commissioned. Physically, the ICRH-DAC is split between the RF Lab and the SST-1 building. There are separate DAC servers for the RF Generation area of the RF Lab and the Transmission line, interface, and antenna area of the SST-1 hall. Both are based on Versa Module Euro card (VME) technology. Both DACs' Linux clients are now linked through Ethernet. The parameters utilized in an experiment are exchanged between the master and slave DAC clients via the master's role as an Input/Output Controller (IOC) server in the Experimental Physics and Industrial Control System (EPICS). The Control System Studio (CSS) suite includes a client called Best OPI Yet (BOY). It includes both a coding environment and a runtime for creating control panel operator interfaces. In this paper, we present the user-perceived performance test findings and describe the prototype software. We provide an analysis of the performance data and its meaning in light of the memory architecture, the processing power, and the particular networking protocols used. We can pinpoint the performance bottlenecks and determine how to fix them thanks to our in-depth investigation. EPICS's Channel access layer and the operator panels built with CSS BOY have been incorporated into the software that was developed using EPICS's Input-Output Controller (IOC).

I. INTRODUCTION

The Ion Cyclotron Resonance Heating (ICRH) experiment's data acquisition and control system (DAC) has been approved for use. For a fusion device, ICRH holds much promise because of its confined power deposition profile, direct ion heating at high density, and well-established technology for managing high power at low expense. At pulse widths of up to 1 ms, 1.5 MW of RF power will be sent through the plasma [1]. This process is managed from afar by a Master DAC system, while a slave DAC system handles transmission of the generated power along with corresponding network and antenna diagnostics. Hardware and software integration have both been built, tested, and kept up to date in advance of any experiments [2,3]. The 22–25 MHz, 45.6 MHz, and 91.2 MHz RF power generation equipment in the RF Lab is controlled and monitored by a master DAC. The VME terminal is the endpoint of the front-end electronics and signal conditioning chain that connects signals from several stages. The software can manage anything from a few channels of simple data streaming gear to racks of complex data collection instruments. The diagnostics and control of the RF transmission over two transmission lines, employing both offline and online matching, are under the purview of the slave DAC system.

DACs that are connected to Ethernet do so via TCP/IP sockets. Data transmission to other networked systems will be required as determined by the system's parameters. In order to receive data or events from another networked system, one system must first establish a control link. Once a control connection has been established, data can be transferred using TCP or UDP. Information has been transferred between the networks. Now that the necessary experimental condition has been met, the link can be made. been closed or demand for another event has been occurred.

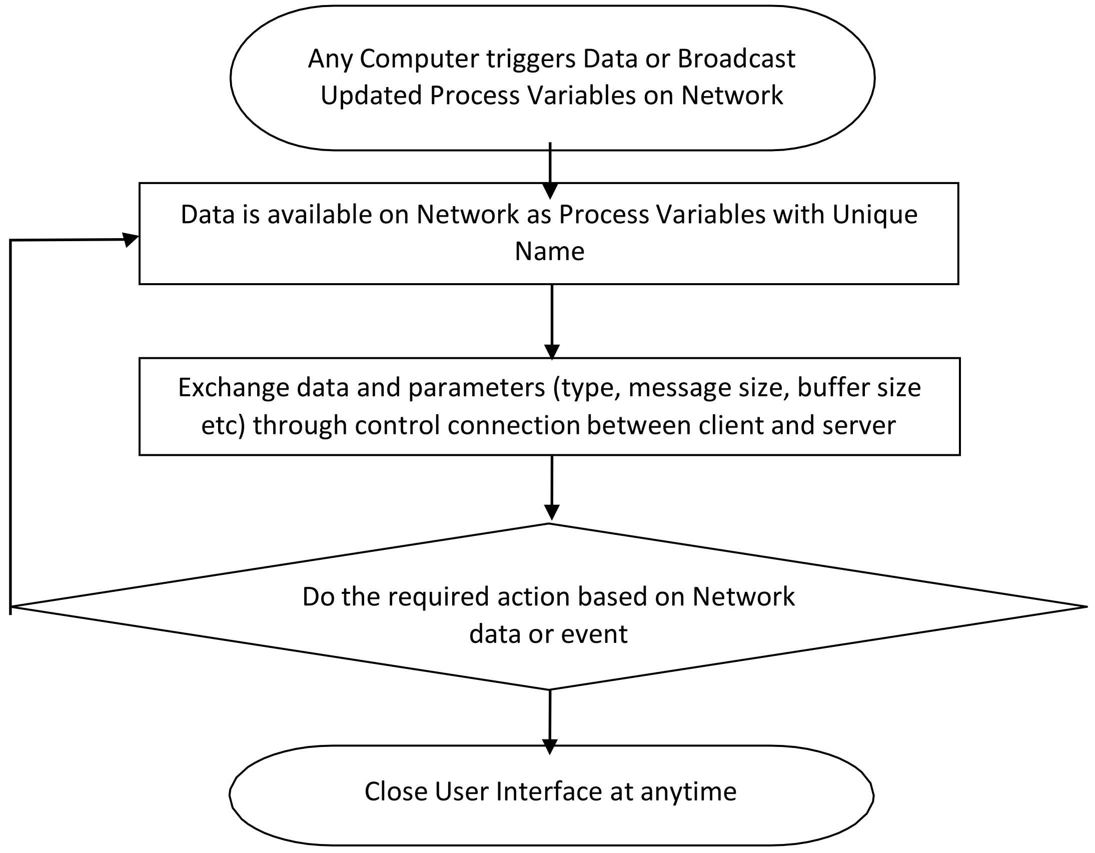

Any Computer triggers Data or Broadcast Updated Process Variables on Network
Data is available on Network as Process Variables with Unique Name
Exchange data and parameters (type, message size, buffer size etc) through control connection between client and server
Do the required action based on Network data or event
Close User Interface at anytime

Figure.1: Communication diagram

II. PROPOSED SOLUTION

A number of optimizations have been proposed as solutions to these issues. The first is to lessen the amount of data processing time required for each send and receive socket call. Large contiguous buffers are proposed as a replacement for the current memory buffers, which are seen as too restrictive. Because of its high cost, memory has to be allocated with great care. With today's powerful processors and low-cost memory, we can move our attention to improving performance while keeping our code as simple as possible. Figure.1 depicts the subsystem link communication diagram. Communication performance is limited by both the TCP/IP packet overhead and the physical communication channel. Data and events have increased the demand for networked systems. Deterministic output with experimental constraints is necessary for these systems. After looking at a number of articles, it became clear that LAN-connected computers should exchange data using the Transmission Control Protocol (TCP) and use the User Datagram Protocol (UDP) for communicating with one another. EPICS's channel access layer serves as the broadcasting mechanism for this paradigm.

III. EPICS IMPLEMENTATION

The EPICS program [4] is mostly used for what it was initially intended: a tool-based method of controlling processes. It is also crucial to have a framework in place that promotes the sound development of distributed software systems. Toolkits need communication software interfaces tailored to prevent application programmer caused mutual exclusion deadlocks, for instance, in multi-threaded distributed systems. The EPICS-based software tools consist of both primary development software, which provides a robust technology for software integrators and application users to tailor and automate the application software, and secondary development software, which gives a satisfactory solution for measuring and processing tasks. EPICS's many add-ons include not just an alarm handler but also interlocking and alarming mechanisms [5]. For user interface creation, we also have access to CSS Best Opi Yet (BOY), which integrates an XML-based markup language with a variety of pre-built widgets and supports the Eclipse IDE. It would seem that some components of the current EPICS communication software interfaces are crucial enablers for cutting-edge toolkits. Interfaces between software and autonomous systems must be able to provide an asynchronous response timed to external events. The integrated architecture of networked systems is seen in Figure 2.

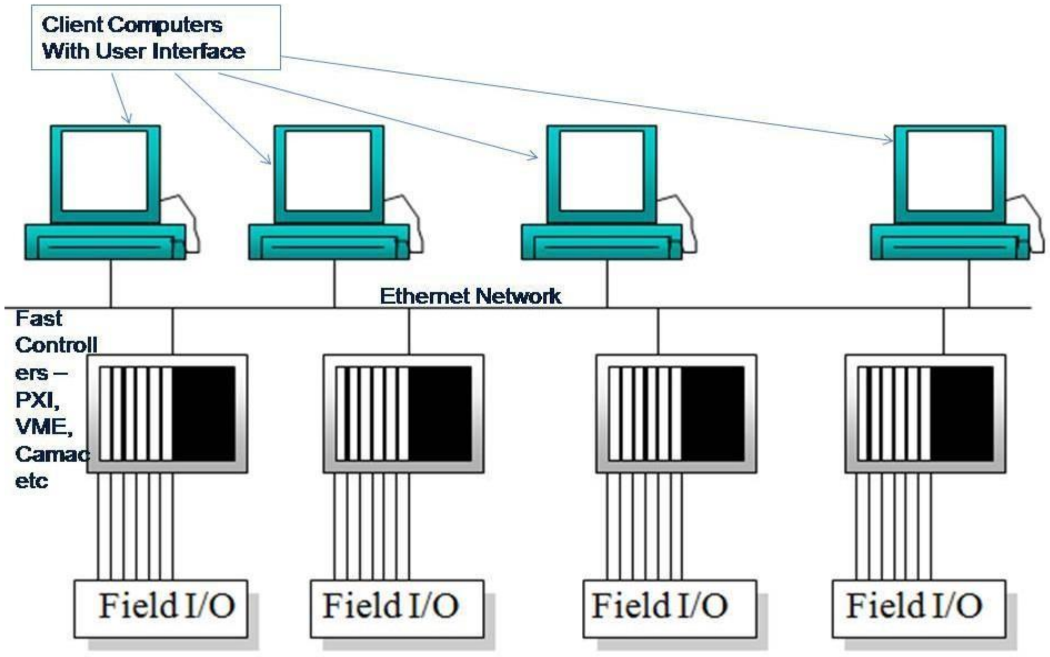
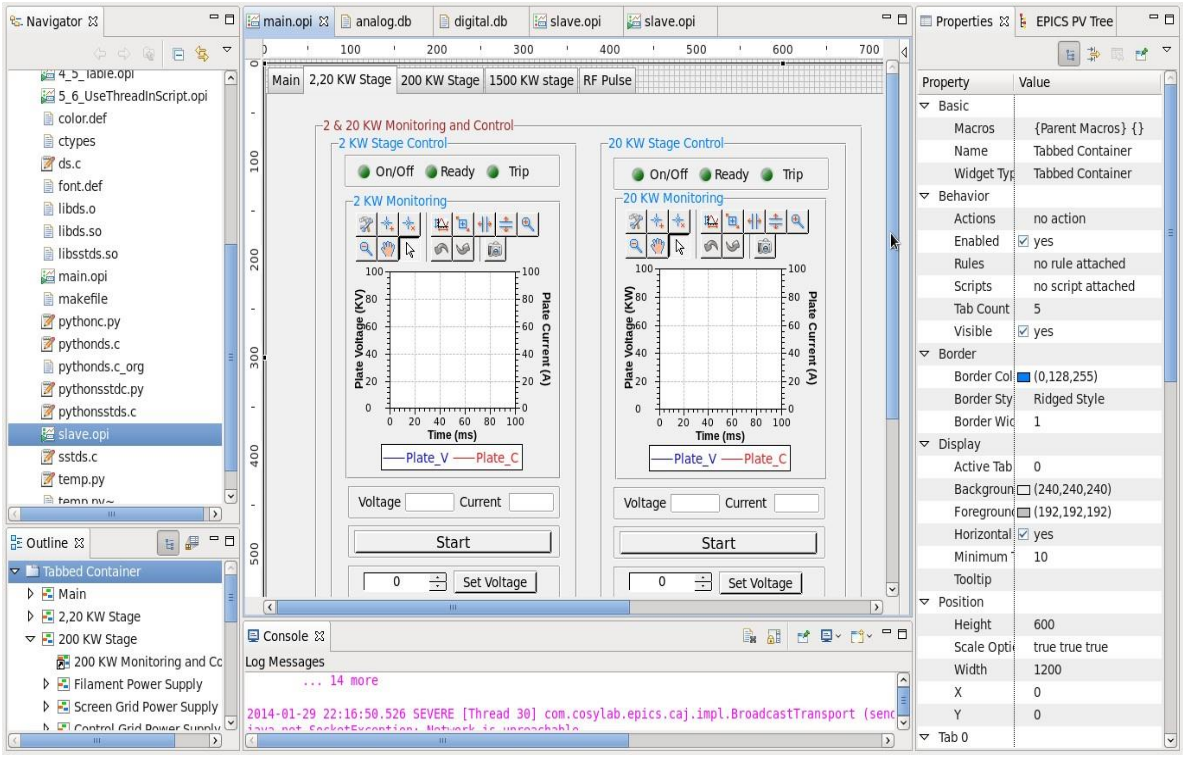


Figure.2: Integrated architecture of Connected DAC systems

IV. PROTOTYPE APPLICATION

CSS BOY is an Operator Interface (OPI) development and runtime environment. An OPI is a general GUI but with extra facilities to connect to your live data directly. CSS BOY allows building your GUI with drag and drop and connecting to your data instantly [6]. It also allows using JavaScript or Jython to manipulate the GUI in a very similar way as using JavaScript in HTML. In BOY, the OPI Editor is a WYSIWYG (What You See Is What You Get) editor which allows you to create your GUI in a similar way of creating PPT. The OPI Runtime works in a similar way as modern web browsers. One can display the OPIs either in tabs, windows or views and navigate OPIs forward or backward. An OPI is a regular XML file that can be edited in OPI editor or text editor and run in OPI Runtime. No compilation is needed. Figure.3 shows the user interface development for DAC software 2 kW and 20 kW stage. Same way in runtime the experimental shot panel has been shown in figure.4. One has to feed the required parameter and give shot in synchronous with other network-connected subsystems. The data communication layer is a separate layer, which allows BOY connecting to various data sources seamlessly. Users can provide their own data source by extending an Eclipse extension point. Figure.5 shows the terminal screen for broadcast of the process variables using EPICS module. EPCIS provides command softIOC that is used for broadcasting.

To make user interface the state notation language (XML) has been used with CSS IDE and assign required field widgets with respective process variables. The signal naming has been specified at the ICRH:<signal_name>. Using softIOC module we have broadcasted the process variables (PVs). XYgraph has been chosen for monitoring the voltage and current signals. Python script has been used for the periodic assignment of the channels process variables using caput command for apply periodic new value to the respective process variable. Separate python script is running periodically using execute command function provided on action button click event. In this script we have used pyepics [7,8] package and import epics as python module and will able to process caget and caput command as per requirements [9,10]. DIII-D has used open source solution for reliable and failsafe solution for the experimentalrequirement [11].



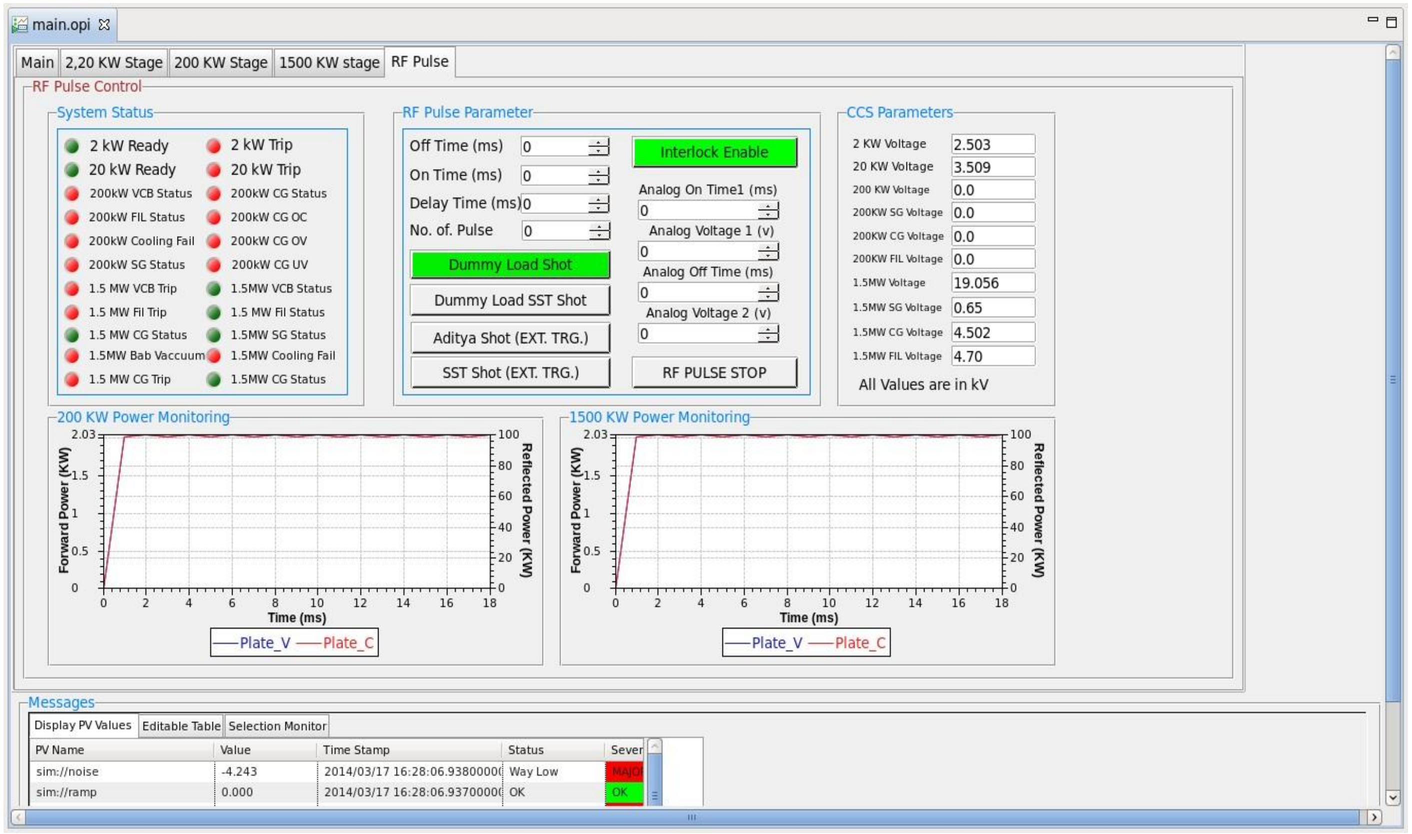
```
        ... 14 more

2014-01-29 22:16:50.526 SEVERE [Thread 30] com.cosylab.epics.caj.impl.BroadcastTransport (senc
java.net.SocketException: Network is unreachable
```

Property	Value
Basic	
Macros	{Parent Macros} {}
Name	Tabbed Container
Widget Typ	Tabbed Container
Behavior	
Actions	no action
Enabled	✓ yes
Rules	no rule attached
Scripts	no script attached
Tab Count	5
Visible	✓ yes
Border	
Border Col	(0,128,255)
Border Sty	Ridged Style
Border Wic	1
Display	
Active Tab	0
Backgroun	(240,240,240)
Foregroun	(192,192,192)
Horizontal	✓ yes
Minimum	10
Tooltip	
Position	
Height	600
Scale Opti	true true true
Width	1200
X	0
Y	0
Tab 0	

Figure. 3: CSS OPI user interface screen for DAC software

The fast fiber optic trigger network will give trigger pulse to fast controller at Master DAC. This fast controller get triggered that will trigger the fast controller at RF transmission DAC fast controller. As fast controller triggered the digitizer card buffer memory will get filled with the given on-time reference time. This data will be acquired by the Linux terminal user interface program with acquire button by socket command using Ethernet. Master DAC will be synchronized by Network Time Protocol (NTP) from Master GPS timer. Master DAC will communicate with slave DAC system with EPICS process variables. Data acquired at master DAC have been sent to the slave DAC and that will acquire data accordingly. Instead of using the original EPICS IOC [12, 13], we decided to develop our own EPICS software toolkit, custom implementation, which is capable of building EPICS Channel Access (CA) server and client programs. In J-TEXT, EPICS provides improve productivity with channel access [14]. SPIDER tokamak has also support same kind of implementation with performance [15]. The most results has been matched with ANL Lab, USA [16] and ESS Bilbao, Spain [17] for network based systems.



● 2 kW Ready	● 2 kW Trip
● 20 kW Ready	● 20 kW Trip
● 200kW VCB Status	● 200kW CG Status
● 200kW FIL Status	● 200kW CG OC
● 200kW Cooling Fail	● 200kW CG OV
● 200kW SG Status	● 200kW CG UV
● 1.5 MW VCB Trip	● 1.5MW VCB Status
● 1.5 MW Fil Trip	● 1.5 MW Fil Status
● 1.5 MW CG Status	● 1.5MW SG Status
● 1.5MW Bab Vaccuum	● 1.5MW Cooling Fail
● 1.5 MW CG Trip	● 1.5MW CG Status

Off Time (ms)	0
On Time (ms)	0
Delay Time (ms)	0
No. of. Pulse	0
Analog On Time1 (ms)	0
Analog Voltage 1 (v)	0
Analog Off Time (ms)	0
Analog Voltage 2 (v)	0

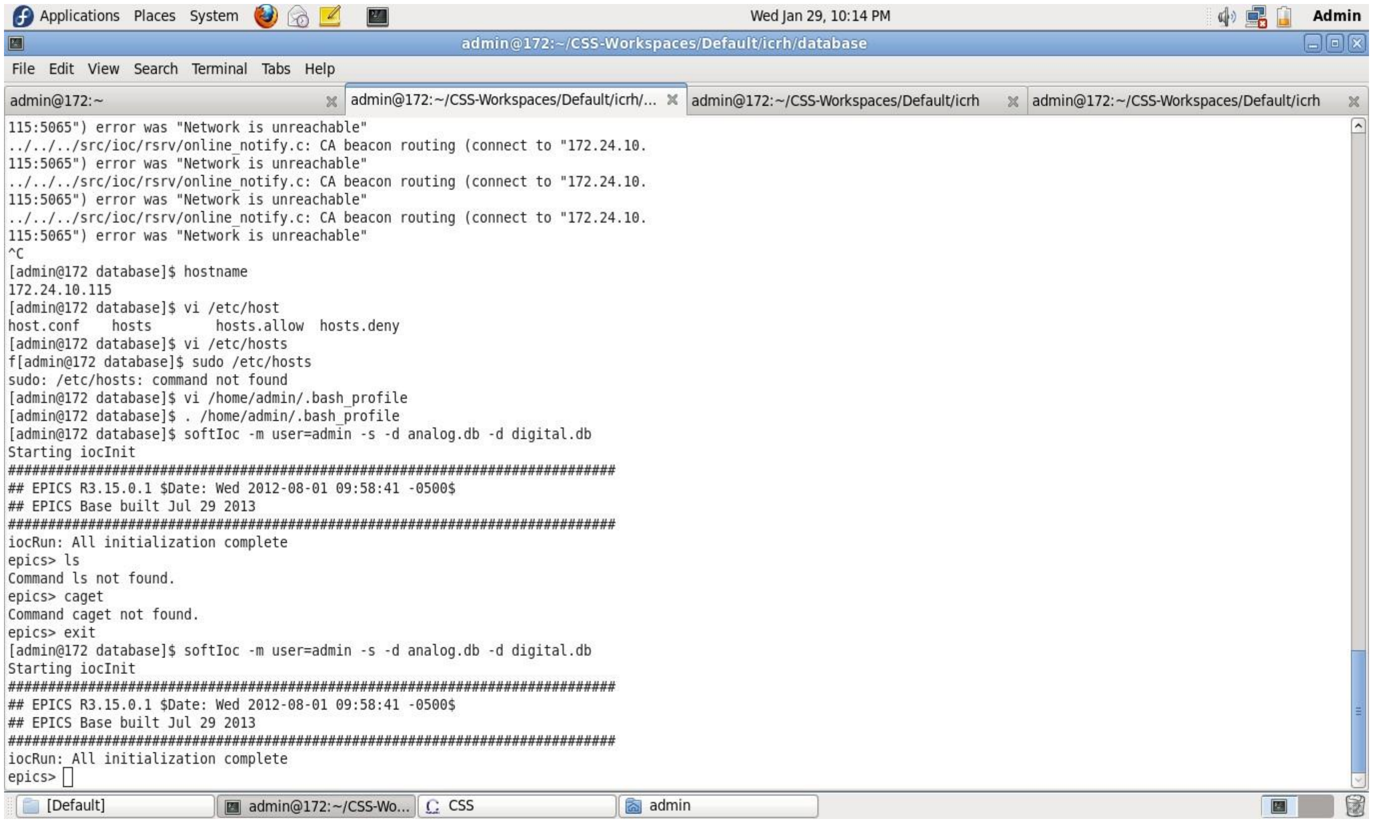
2 KW Voltage	2.503
20 KW Voltage	3.509
200 KW Voltage	0.0
200KW SG Voltage	0.0
200KW CG Voltage	0.0
200KW FIL Voltage	0.0
1.5MW Voltage	19.056
1.5MW SG Voltage	0.65
1.5MW CG Voltage	4.502
1.5MW FIL Voltage	4.70

All Values are in kV

PV Name	Value	Time Stamp	Status	Sever
sim://noise	-4.243	2014/03/17 16:28:06.9380000(	Way Low	MAJOI
sim://ramp	0.000	2014/03/17 16:28:06.9370000(	OK	OK

Figure. 4: Experimental shot monitoring and control screen for DAC software

It is important to note that most signals are not monitored by channel access clients and that monitors are only sent on change of state or excursion outside of a dead-band. The database scanning is flexible to provide optimum performance and minimum overhead.


Applications Places System
Wed Jan 29, 10:14 PM
Admin
admin@172:~/CSS-Workspaces/Default/icrh/database
File Edit View Search Terminal Tabs Help
admin@172:~
admin@172:~/CSS-Workspaces/Default/icrh/...
admin@172:~/CSS-Workspaces/Default/icrh
admin@172:~/CSS-Workspaces/Default/icrh
[Default]
admin@172:~/CSS-Wo...
CSS
admin

```
115:5065") error was "Network is unreachable"
../../../src/ioc/rsrv/online_notify.c: CA beacon routing (connect to "172.24.10.
115:5065") error was "Network is unreachable"
../../../src/ioc/rsrv/online_notify.c: CA beacon routing (connect to "172.24.10.
115:5065") error was "Network is unreachable"
../../../src/ioc/rsrv/online_notify.c: CA beacon routing (connect to "172.24.10.
115:5065") error was "Network is unreachable"
^C
[admin@172 database]$ hostname
172.24.10.115
[admin@172 database]$ vi /etc/host
host.conf    hosts        hosts.allow  hosts.deny
[admin@172 database]$ vi /etc/hosts
f[admin@172 database]$ sudo /etc/hosts
sudo: /etc/hosts: command not found
[admin@172 database]$ vi /home/admin/.bash_profile
[admin@172 database]$ . /home/admin/.bash_profile
[admin@172 database]$ softIoc -m user=admin -s -d analog.db -d digital.db
Starting iocInit
############################################################################
## EPICS R3.15.0.1 $Date: Wed 2012-08-01 09:58:41 -0500$
## EPICS Base built Jul 29 2013
############################################################################
iocRun: All initialization complete
epics> ls
Command ls not found.
epics> caget
Command caget not found.
epics> exit
[admin@172 database]$ softIoc -m user=admin -s -d analog.db -d digital.db
Starting iocInit
############################################################################
## EPICS R3.15.0.1 $Date: Wed 2012-08-01 09:58:41 -0500$
## EPICS Base built Jul 29 2013
############################################################################
iocRun: All initialization complete
epics> 
```

Figure. 5: Broadcasting of Process Variables using softIOC for DAC communication

V. TEST SETUP AND RESULT ANALYSIS

Several hundred points on periodic monitor and control with tens of physical connections are possible in the prototype system's environment. In order to meet the specific requirements of its users, the EPICS environment allows for system expansions at all levels. Our modular software architecture, which allows for modifications at any level, allows us to both connect with an existing user base and pave the way for future upgrades. With fewer connections required for each data exchange, network overhead may be dramatically minimized. The experimental configuration for a collaborative network-based environment is simplified by a lightweight broadcasting technique that enables straightforward communication. EPICS and CSS together provide a simple, open-source solution for ensuring dependable and fail-safe functioning. We want to continue developing this into an even more useful resource in the near future.

The following system setup was used for the EPICS benchmark database measurements:

- EPICS version 3.14.12.2
- CSS Opi for User Interface Development
- Intel Core i5 with Fedora OS 14
- PC with Linux OS
- 100 MBPS Ethernet segment

It's important to remember that resource consumption is proportional not just to the total records processed per second but also to the number of channel access (CA) clients. Multiple instances of the reference monitor were run to get an accurate reading. Resource use vs number of associated CAs is shown in Fig. 6. Ten hertz was utilized as the standard scan rate throughout. Thus, a throughput of 1000 records per second was achieved for 100 CA customers. Memory, network, and CPU use for various user interface configurations (just broadcasting of PVs, TK interface, CSS interface, and both interfaces operating concurrently) are shown in Figure 7. CSS's support for Java and Python means more reliable code and the possibility of creating platform-independent code using Java.

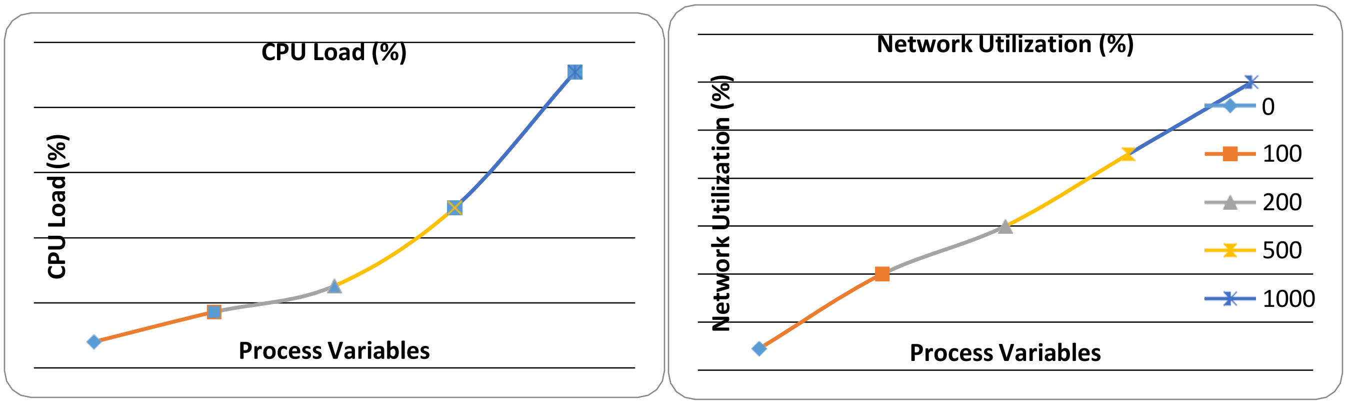
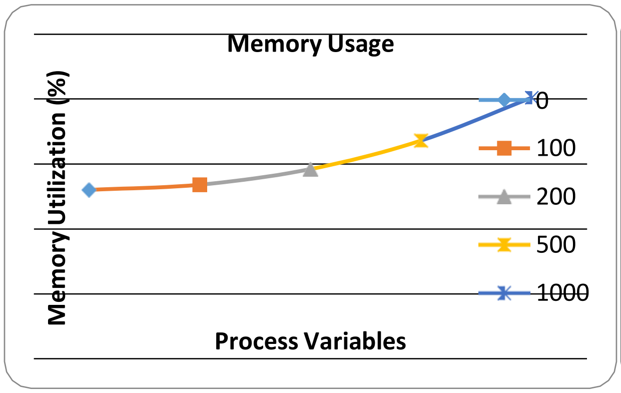


Figure. 6: IOC CPU Load and Network Utilzation at 10 Hz Scan Rate

The IOC is the point of contact between a user and a machine component. The IOC's performance is limited by the CPU's bandwidth and memory. Scanning near the conclusion of the conversion process greatly shortens the time between gating an analog input and beginning to analyze the record. The scanning of the database may be adjusted for maximum efficiency with minimal slowdown. To keep track of the passage of time, we may program a monitor PV to automatically adjust its value at certain intervals. Development platform for ITER control data access and communication (CODAC) has been identified [18]. We opted to create our own EPICS software toolkit, a custom implementation, to construct EPICS Channel Access (CA) server and client applications rather to use the original EPICS IOC [19, 20]. Physical communication medium, TCP/IP packet overhead, and the channel access protocol are the three main factors that limit communication performance. In order to make the most of the available bandwidth, channel access combines numerous requests into one.

or responses. To avoid collisions and therefore avoid non-determinism, the Ethernet load is kept under 30% [21]. At this level, we can issue 10,000 monitors per second. Use of LAN bandwidth can reduce by 50%-80% by changing the channel access protocol to variable command format and compressing the monitor response data (~ 6-15 bytes per packet). In J-TEXT, EPICS provides improve productivity with channel access [22]. SPIDER tokamak has also support same kind of implementation with performance [23].



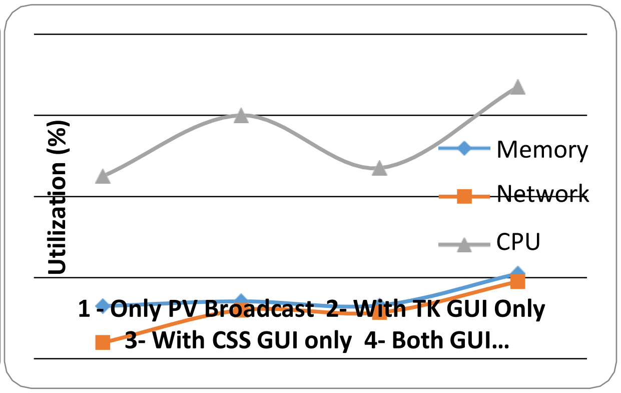


Figure. 7: Memory Utilization at 10 Hz Scan Rate and Parameters with different cases

Channels as process variables have been broadcasted every required time period so when the shot is applied to master DAC, the information has been automatically available to slave DAC on same network. A prototype system has been developed for the same. One EPICS based prototype has been about to complete by which we can produce results and check performance parameters with standard optimization results. We ran the benchmark with the scanning period of 1, 0.5, 0.2, 0.1, and 0.01 second, which corresponds to 100, 200, 500 and 1000 processed records/sec. Tables 1 shows the IOC CPU usage and network utilization respectively. Linux System monitor was used to measure the CPU and memory usage, and the Wireshark network Analyzer tool was used to measure the network segment utilization.

If the record of process variables would be broadcasted than the CPU utilization for every 100 millisecond we can get Table-1 first row results. If the record of process variables would be broadcasted in the network at every 100 millisecond we can get Table-1 second row results. During shot each connected sub systems are working together and client computer acquires data from fast real-time server which increase the CPU load up to 90 percent and network load up to 82 percent as shown in both the table last column. After completion of the experimental shot it come down to the normal state.

TABLE I. Ioc Cpu Load And Netowork Load

Record Processed (100 ms)	0	100	1000	During Shot
CPU Load (%)	2	4.3	22.7	90
Network Load (%)	0.89	4	12	82

VI. CONCLUSION

Several hundred points on periodic monitor and control with tens of physical connections are possible in the prototype system's environment. From modest test stands with a few hundred points per second to huge distributed systems with tens of thousands of physical connections, the EPICS offers an environment for developing these systems. Some of the most essential requirements of the projects that are managing user facilities are being met by enhancing the underlying software. Depending on the amount of PVs, clients, and scan pace, either the CPU or RAM might become the limiting factor in IOC. EPICS's scalability, on the other hand, permits the relocation of databases and programs in order to distribute the work more evenly. EPICS's lightweight broadcasting technique uses the network's channel access layer to reduce unnecessary traffic on the system.

VII. REFERENCES

1. SST-2.03-050199-RF Group Engineering Design Report (EDR) for ICRH-SST1

European Physical Society (EPS), 1992, p. 81 2. Durodie F., Veriver M. 3.Nucl. Fusion 46, no. 3 (2006) D. Bora et al.

http://www.iter.org/doc/www/edit/Lists/ WebsiteText/Attachments/94/PCDHv.61.pdf, The Plant Control Design Handbook

ICRH data gathering using EPICS and MDSplus, International Journal of Computer Science Research and Application, Ramesh Joshi, et al., 2013.

Proceedings of the 2011 Particle Accelerator Conference, New York, USA, Xihui Chen, et al., BOY: A Modern Graphical Operator Interface Editor and Runtime.

7 - http://www.aps.anl.gov/epics/, Homepage of the Experimental Physics and Industrial Control System

For Python's Ctypes module, see http://python.net/crwe/theller/ctypes/..

EPICS 9 Main Index. [Online] To be found at http://www.aps.anl.gov/epics

http://cars.uchicago.edu/software/python/pyepics3/ 10 PyEpics Python Package

Fusion Engineering and Design 87 (2012) 1977-1980 Benjamin G. Penaflor et al., Custom open source solution for DIII-D data collection and control system

Conference preceeding Springer, March-2014 12 Ramesh Joshi et al., Conceptual design of EPICS based implementation for ICRH DAC system

Fusion Engineering and Design 86 (2011) 1085-1090; Paulo Carvalhoa et al., "EPICS IOC module development and imple- mentation for the ISTTOK machine subsystem operation and control."

J-TEXT-EPICS: An EPICS toolbox that tries to boost productivity, Fusion Engineering and Design 88, no. 3041–3045 (2013), Wei Zheng, et al.

15) Luchetta, A., et al., Fusion Engineering and Design 87 (2012) 1933-1939, Architecture of the SPIDER Control and Data Acquisition System.

1. Proceedings of PCaPAC 2010, Saskatoon, Saskatchewan (2010); Shifu Xu et al., EPICS IOCCORE REAL TIME PERFORMANCE MEASUREMENT ON COLDFIRE MODULE.
2. Proceedings of IPAC'10, Kyoto, Japan (2010); M. Eguiraun et al., "NETWORKED CONTROL SYSTEM OVER AN EPICS BASED ENVIRONMENT."
3. http://www.iter.org/doc/www/edit/Lists/ WebsiteText/Attachments/94/PCDHv.61.pdf, The Plant Control Design Handbook
4.
5. 4 - http://www.aps.anl.gov/epics/, Homepage of the Experimental Physics and Industrial Control System
6. Fusion Engineering and Design 86,2011,1085-1090, Paulo Carvalhoa et al., EPICS IOC module development and imple- mentation for the ISTTOK machine subsystem operation and control.
7. Pages 219–220 of "Keeping the Link: Ethernet: Ethernet Installation and Management" by M. Nemzow (Magraw-Hill Book Co.).
8. J-TEXT-EPICS:An EPICS toolbox that tried to boost productivity, Fusion Engineering and Design 88, 2013, 3041-3045 7 Wei Zheng et al.
9. SPIDER control and data collection system architecture, Citation: A. Luchetta et al., Fusion Engineering and Design 87,2012,1933-1939