# USING EPICS REDUNDANT IOC IN UNIX ENVIRONMENT*

A. Kazakov**, SOKENDAI/KEK, Japan
K. Furukawa, KEK, Japan; M. Clausen, G. Liu, DESY, Germany

*Abstract*

Redundant EPICS IOC is being actively developed at DESY in order to achieve high availability. Current development focuses on VME vxWorks environment for cryogenics controls. However, many scientific facilities use PC-architecture and unix-like systems as Linux, Solaris or Darwin. These facilities require high availability and redundancy as well. So this paper describes the implementation of EPICS redundant IOC in PC-based environment with Linux or other UNIX-like EPICS-supported OS. This work was done by porting Redundancy Monitor Task (RMT) and Continuous Control Executive (CCE). CCE is aimed to synchronize two RSRV-based IOC servers. RMT is monitoring other parts of the system, keeps connection with the partner and it is responsible to make a decision when to fail-over; it is rather independent and may be used in a wide range of applications. It was successfully employed to add redundancy to caGateway.

## INTRODUCTION

Originally EPICS redundant Input Output Controller (IOC) was developed at DESY. And two major fields of application were defined:

1.Redundancy for cryogenic plants. In this case the failure may be caused by malfunctioning hardware as power supplies or fans. And in this case automatic fail-over mechanism should guarantee system stability. But over the years it was sometimes necessary to manually switch between main and backup processors due to maintenance work during runtime period (which usually is one year or more). Another case where it might be useful is software update. While current commercial system being used at DESY allows on-line updates to the database EPICS does not allow to add or delete records and databases while in operation.

2.Redundancy for controllers in the XFEL tunnel. Though the main origin for switch-over in the first case would be manual action, it is expected to happen automatically in the XFEL tunnel. Due to high radiation a damage to CPU and memory is highly possible. Software update is not so important because of more frequent maintenance days when this operation may be performed.

By the design draft one major goal was set: **Any redundant implementation must make the system more reliable than the non redundant one. Precaution must be taken especially for the detection of errors which shall initiate the fail-over. This operation should only be activated if there is no doubt that keeping the actual mastership definitely causes more damage to the controlled system than an automatic fail-over**. And the fail-over time in any case was defined to be more than several seconds and less then 15. Final implementation could switch in less then 2 seconds.

Originally the project was supposed to support only vxWorks and all the code written was very vxWorks specific. But later it was seen that other operating system support is desirable. Here at KEK we use software-IOC on Linux which work as “gateways” from an old control system to EPICS-environment. Also for the ILC project ATCA-based systems under Linux control will be used. And redundant IOCs are highly desirable for this project. Thus the porting of redundant IOC was done to EPICS libCom/osi, which means that current implementation should work on any EPICS-supported operating system.

## HARDWARE ARCHITECTURE

Hardware architecture consists of two redundant I/O Controllers providing control of remote I/O via a shared media like Ethernet or 1553. The redundant pair share two network connections for monitoring the state of health or each other where the private network connection is used to synchronize the backup to the primary and the global network is used to communicate data from the primary to any other network clients interested in the date.

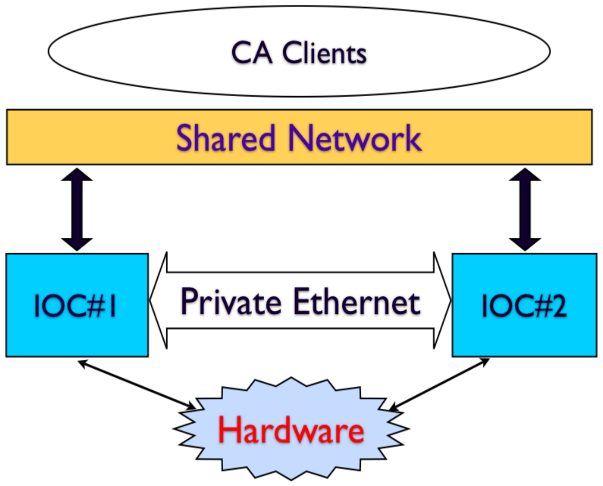

Figure 1: Hardware architecture.

## SOFTWARE COMPONENTS

Current design contains three major parts: RMT, CCE(Continuous Control Executive) and SNL executive. The latter two are responsible for synchronization of process variables (PV) and internal state. And RMT is the core of the redundant system. It establishes and maintains the connection with the partner, controls other parts of the

*Work supported by SOKENDAI, Japan
**kazakov@gmail.com

system and decides when to fail-over. All other components which has to be controlled by RMT share the same software interface (which is defined in a header file rmtDrvIf.h). This interface defines the following functions:

1. start: Get access to the IO and start processing.
2. stop: Do not access to the IO and stop processing.
3. testIO: Initiates a procedure to test access to the IO.
4. getStatus: Get status of the driver.
5. shutdown: This function is called before the IOC is rebooted. It terminates transient activities, deactivates interrupt sources and stops all driver-tasks.
6. getUpdate: This routine tells the component to get an update from the redundant IOC. It is normally called by the RMT on the inactive IOC.
7. startUpdate: This Routine tells the component to start updating data from the redundant IOC (monitoring). First it will read all fields (depending on mode) from the redundant partner.It is normally called by the RMT on the inactive IOC.
8. stopUpdate: This Routine tells the component to stop updating data from the redundant. This routine is normally called by the RMT on the inactive IOC.

RMT can call this functions using entry table which is passed from the component to RMT during initialization. To register itself in RMT and pass an entry table the component makes a call to rmtRegisterDriver() function. getInfoCCE and SNL executive are such RMT-controlled components and they implement RMT-driver interface mentioned above. Other components may be IO-drivers, or any other piece of software. For example RSRV server in redundant IOC implementation is one of the RMT-controlled components and it implements RMT-driver interface. Some of the interface functions may be left unimplemented (set to NULL in the entry table) depending on the nature of the IO-driver and the tasks it it is carrying.

## PORTING

Originally all the development was done for vxWorks only. And the resulting code was not usable on any other operating system (OS). But people at KEK were interested in using redundant IOC on Linux machines for LINAC control system. Also at DESY there was a demand for redundant CA gateway, and it felt natural to utilize RMT for that purpose. But CA gateway runs on Linux (or other UNIX-like OS), and RMT was available only on vxWorks. Thus it was decided to port redundant IOC to Linux.

### *Porting Process*

All vxWorks specific function calls were replaced with epics *libCom/OSI (Operating System Independent library)*. In order to accomplish this task the following new OSI functions were introduced:

- **epicsMutex.h:** epicsMutexLockWithTimeout(id, tmo);
- **epicsMutex.h:** epicsMutexOsdLockWithTimeout(id, tmo);
- **epicsThread.h:** epicsThreadDelete(id);
- **epicsTime.h:** epicsTimeGetTicks ();

Also, some modification to the EPICS base were done to implement CCE hooks, implement RMT-driver interface in CA server (RSRV) and in database scan tasks. These modifications were also made operating system independent. Here is the list of affected source files:

- **base/src/db/dbAccess.c**
- **base/src/db/dbScan.c**
- **base/src/dbStatic/dbBase.h**
- **base/src/dbStatic/dbLexRoutines.c**
- **base/src/rec/aiRecord.dbd**
- **base/src/rsrv/camsgtask.c**
- **base/src/rsrv/online_notify.c**
- **base/src/rsrv/cast_server.c**
- **base/src/rsrv/caservertask.c**

Please contact the authors for the patch files.

### *Results*

Ported OSI version of redundant IOC was successfully used on vxWorks, Linux, Mac OS X, Solaris. Testing showed system synchronisation speed limit is around ~5000 records/second for 2 linux machines with 3GHz P4 1core, 2x 100Mbit ethernet cards; doing nothing but redundant IOC.

## REDUNDANT CA GATEWAY

One of the demands in DESY was to implement redundant CA Gateway. And after RMT being ported to Linux it fits very well to help this task. CA Gateways are being widely used in many facilities around the globe which utilize EPICS. CA protocol was designed to be used only in one network segment, so CA gateway main purpose is to add access to the network segment from the other network segment. Also it can be used to add additional security layer or to divide network into several segments in order to reduce the amount of broadcast traffic in each segment. But in any case when CA gateway is utilized it becomes “one point of failure” and even if some or all IOCs are redundant in a case of gateway failure some clients and IOCs would be cut-off. So it seems essential to add redundancy to such application as CA Gateway.

By the nature of the Gateway task, it has no internal status needed to be synchronized, and thus it desirable to use both redundant gateway at the same time (load-balance them). It will reduce the load on each of them and increases system peak throughput. Also in case of failure of one of them only half of the clients will notice the problem and will reconnect. So the task was divided into two steps.

### *Redundancy Without Load-Balancing*

Redundancy without load-balancing was implemented by using RMT as stand-alone application, which runs separately from CA Gateway. It give the benefit of not modifying the source of the Gateway, but raises the problem of how to control the Gateway. So the chosen solution was to use firewall rules to block replies from the Gateway process. In case of load-balancing only it is also possible to block incoming search requests from CA clients. But when we add load-balancing it will not work (*see next section for details).*

In order to add and delete firewall rules so called "script-driver" was implemented as RMT-driver. Upon becoming a master this driver makes a call to an external "start-script" (which may be any executable file) and upon becoming a slave this driver calls and external "stop-script". This script driver may be used in other applications and may call any external application on specified status change. In our case "stop-script" adds a firewall rule which blocks replies from the gateway to the clients. And "start-script" removes this firewall rules, making possible to send replies to the clients. So at any taken time there is only one replying Gateway to the clients. The other one is just sitting quietly. This scheme worked well, so we proceeded to the next step.

### *Load-Balancing*

This part appeared to be more tricky, because it required more tight communication between RMT and CA Gateway. Load-balancing CA Gateway has to be informed of the current state of its partner. And in our approach of having them as separate processes it was decided to use "UNIX signals" mechanism to inform the Gateway process of status change. So some modification was done to the Gateway source, which allows to catch two signals. One of them means that the partner become "alive" and the other means that the partner become "dead".

The logic of load-balancing gateway is simple: when the partner is alive, every other reply from the gateway includes its partner's IP address (this is the same functionality of CA which is used in CA directory service). And when the partner is "dead", the Gateway replies as usual. So both gateways do the same if they are both alive. Upon receiving a PV search request from a CA client they make a search on the IOC network and if succeed create a "virtual-circuit connection" to the corresponding IOC. Also on this stage CA Gateway adds this PV to the list of known PV. And then reply to the client is sent (containing either partner's ip address or it's own). But on the slave side RMT's "script-driver" adds a firewall rule blocking replies, so the clients receive only one reply from the master. And master is load-balancing the further requests between itself and its partner. So after receiving a reply from the master gateway CA client makes a connection to one of the two gateways. And if we would block incoming request from the client instead of the reply from the gateway the client could not connect later to the slave gateway because its list of know PV would be empty and it will deny all incoming connections.

So as a result of putting all this together we get load-balancing redundant CA Gateway. It required some minor changes to the CA Gateway source code. These changes include signal handling, load-balancing functionality and new command line options for configuring partner IP address and signal numbers.

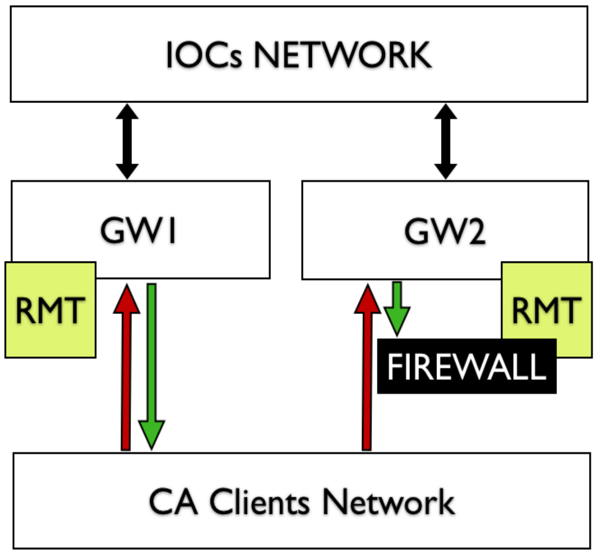

Figure 2: Redundant CA gateway.

## SUMMARY

Porting redundant IOC to UNIX-environment allowed much wider application of this system. And as it was shown RMT as core of the redundant system can be utilized to add redundancy to other software.