

An communication protocol for a distributed control system with LabVIEW™

L.Catani
INFN-Roma Tor Vergata, Via della Ricerca Scientifica 1, Roma, Italy

Control systems for accelerators at INFN-LNF (Laboratori Nazionali di Frascati) are mainly based on LabVIEW. VME crates housing local controllers are interconnected using bus extenders that translate local memory spaces into a global shared memory. Mailboxes are then used for communications between distributed processors.

While the development of control systems for new accelerators under construction at INFN-LNF should be based on well-established and more flexible technologies for communication, i.e. network communication, reuse of instrument drivers, sub-system controls and other tools already developed must be guaranteed. This paper presents the development of an RPC-like communication protocol based on the TCP/IP and XML, tools provided by LabVIEW. It extends the features of these built-in libraries, including the managements of large binaries, while preserving compatibility between different platforms supported by LabVIEW.

Overview of XMLvRPC

XMLvRPC TCP/IP Server

It runs on each controller and on the Configuration Database. It opens XMLvRPC methodsCalls. For each controller, available methodsCalls are those listed in the XMLvRPC_ClientNameMethods.xml directory. Elements under control are those listed in XMLvRPC_ClientNameElements.xml directory.

XMLvRPC TCP/IP Client

It runs on each control to client in general and send XMLvRPC methodsCalls to XMLvRPC TCP/IP Server according to the requests of the control panel or user application.

XMLvRPC UQP Receiver (Configuration Database)

It runs on the Configuration Database and stores requests, then it registers also methodsCalls sent by controllers or clients, relevant to controllers at startup.

XMLvRPC UQP Sender

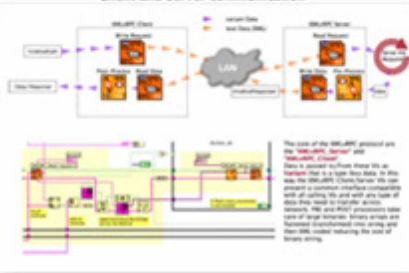
It runs on the controllers, controls at startup. It sends requests, then it registers also methodsCalls to Configuration Database to register the new controller in the system. Controller use it to locate the Configuration Database.

Configuration Database

It is the repository of the configuration files. It provides to the controls information about the controller in charge for a given element.



Client and Server communication

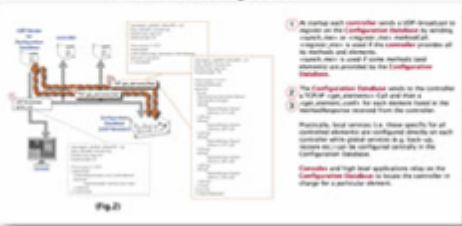


XMLvRPC package

The folder below contains all the files (except the "system database") used by the different components of the XMLvRPC protocol, i.e. controller, client and Config Database (XMLvRPC).

File Name	Description
XMLvRPC_ClientNameMethods.xml	Configuration file listing the methods available to the client.
XMLvRPC_ClientNameElements.xml	Configuration file listing the elements available to the client.
XMLvRPC_ServerNameMethods.xml	Configuration file listing the methods available to the server.
XMLvRPC_ServerNameElements.xml	Configuration file listing the elements available to the server.
XMLvRPC_ClientNameMethods.xml	Configuration file listing the methods available to the client.
XMLvRPC_ClientNameElements.xml	Configuration file listing the elements available to the client.
XMLvRPC_ServerNameMethods.xml	Configuration file listing the methods available to the server.
XMLvRPC_ServerNameElements.xml	Configuration file listing the elements available to the server.
XMLvRPC_ClientNameMethods.xml	Configuration file listing the methods available to the client.
XMLvRPC_ClientNameElements.xml	Configuration file listing the elements available to the client.
XMLvRPC_ServerNameMethods.xml	Configuration file listing the methods available to the server.
XMLvRPC_ServerNameElements.xml	Configuration file listing the elements available to the server.

Initialization and registration



Development and test bed



A communication protocol for a distributed control system with LabVIEW

L.Catani
(INFN Roma Tor Vergata – Italy)

what is it?

- it is a tentative to develop a communication protocol to support distributed controls using the LabVIEW TCP/IP and UDP libraries.
- it is also potentially compatible with other standards or widely used protocols
- actually, it started as a development of XML-RPC in LabVIEW, let's call it XMLvRPC

motivations

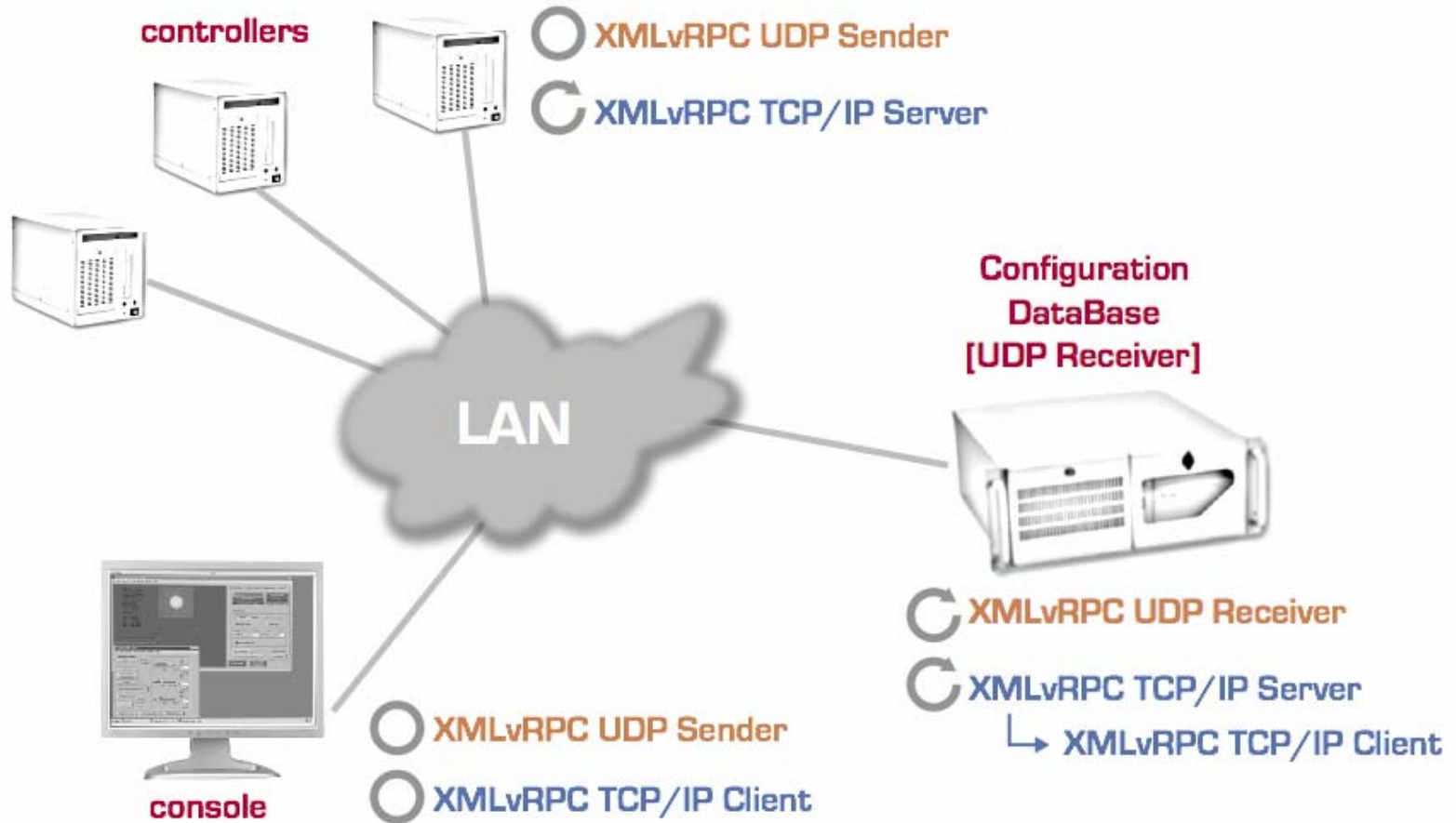
LV supports **distributed controls** in different ways:

- VI Server
- DataSocket
- VI reference
- TCP/IP and UDP
- http server
- and interface to .NET

they all work fine but only **TCP/IP and UDP** provides enough flexibility....

..and the possibility to integrate non-LabVIEW systems

main components in XMLvRPC



the XMLvRPC client and server

```
<?xml version="1.0"?>
<methodResponse>
  <methodName>get_image_data</methodName>
  <params>
    <String>
      <Name>image(2)(U8)</Name>
      <Val>f1EW□Ó□-□bHÇæ'S0å□f#L Y-öÿ~xrQI□TXÍ□æ3ioHÜi~ /Z/Íç□ç_ÿ÷~_1å□¥Oy#□
        Jt°†→=àWrÕmØµ `` @ Σ ~ ∂ lñl π °Óiw_ô>iùl/Y□Q□1Ô»f1□^iñ/g©E~\ Y□Δ
        .....
        ∂ ê·g*ç° CÆzö[p2 ∫ n ∫ +f1çÑÊR□†Õ~□7†g ~ «ò3≈r,,·□ùò9ÉÇÿÇ°;°<□
        ~F□+<nË~M`□Íu□`â)©«ç»È-f1`ó□gæf π,
      </Val>
    </String>
  </params>
</methodResponse>
```

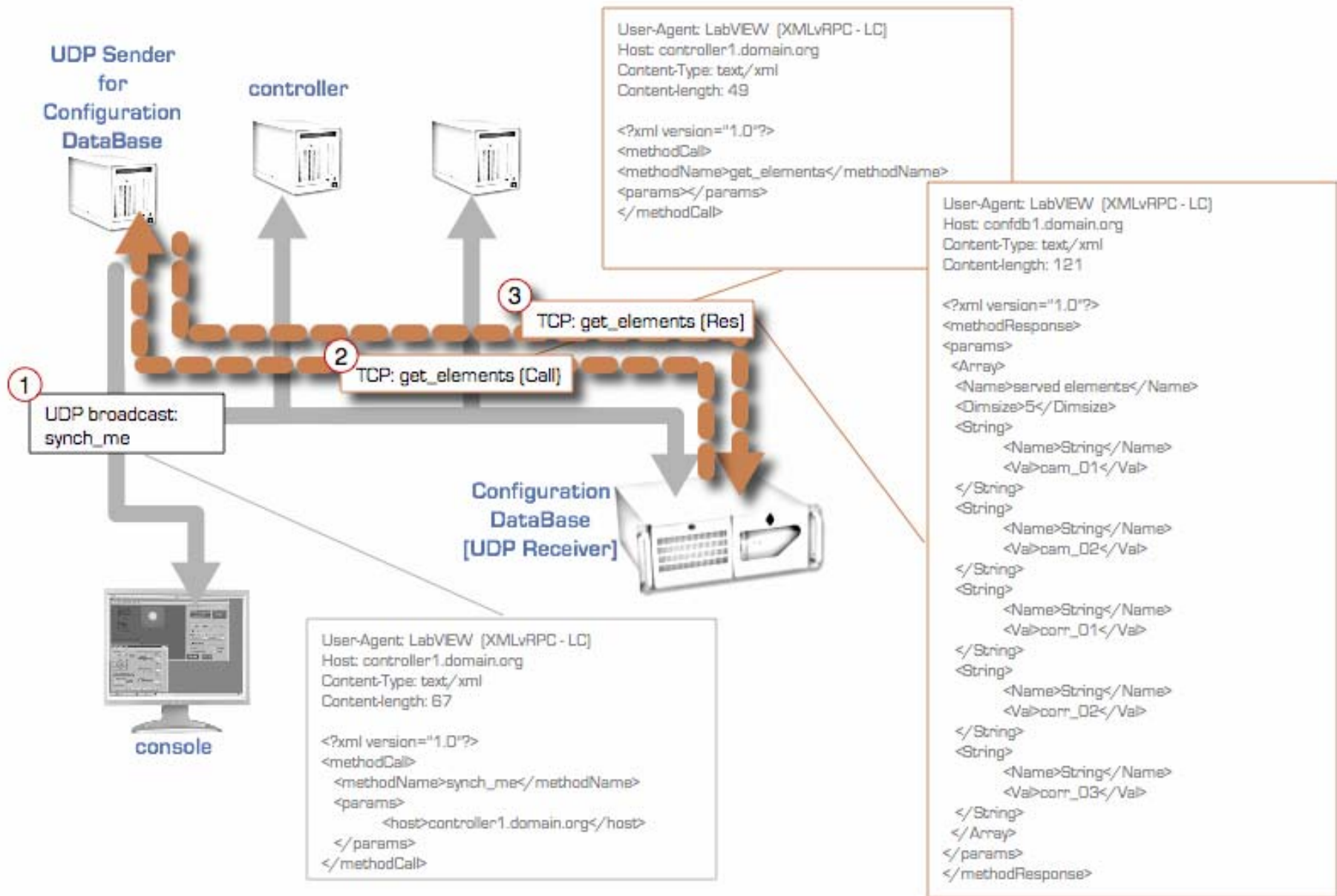
- methodCall (ser

the package (same for all)

Name	Kind
Config_local.xml	BBEdit text document
Flatten_and_Unflatten_XML-RPC	Folder
XML_processors	Folder
XMLvRPC_ClientServer	Folder
classes_all	Folder
elements_svr	Folder
libraries	Folder
methods_cit	Folder
methods_svr	Folder
XMLvRPC_browser.vi	LabVIEW VI Document
XMLvRPC_server.vi	LabVIEW VI Document
XMLvRPC_ConfigDB	Folder
methods_cdb_rec	Folder
methods_cdb_sdr	Folder
system_database	Folder
sparc_vxp1.infn.it	Folder
cam_01.xml	BBEdit text document
cam_02.xml	BBEdit text document
corr_01.xml	BBEdit text document
corr_02.xml	BBEdit text document
corr_03.xml	BBEdit text document
sparc_vxp2.infn.it	Folder
cam_11.xml	BBEdit text document
cam_12.xml	BBEdit text document
corr_11.xml	BBEdit text document
corr_12.xml	BBEdit text document
corr_13.xml	BBEdit text document
XMLvRPC_UDPcdb_rec.vi	LabVIEW VI Document
XMLvRPC_UDPcdb_sdr.vi	LabVIEW VI Document
xtras	Folder
Global_XMLvRPC.vi	LabVIEW VI Document
XMLvRPC_startup.vi	LabVIEW VI Document

- Configuration file; defines the role of the component
- Contains PRE and POST XML processors (binaries management)
- Classes and elements known in the system
- subVIs library
- known methods for this unit (if it is a client)
- known methods for this unit (if it is a controller)
- known methods for this unit (if it is a UDP receiver)
- known methods for this unit (if it is a UDP sender)
- system Database if this unit is a Configuration DB
- UDP receiver: running on Configuration DB
- UDP sender: running on controllers and consoles
- Global variables
- Initialize global var. and service according to the Config_local.xml file

services registration at startup triggered by UDP broadcast



under development on VMs on a MS Virtual Server R2

The screenshot displays a Windows Server 2003 desktop environment with several LabVIEW applications running. The desktop includes icons for Security Configuration, Firefox Setup, Mozilla Firefox, and various help files. The taskbar shows the Start button and several open LabVIEW windows.

Key windows and their contents include:

- XMLRPC_server.vi**: A log window showing repeated entries for `get_element_conf` and `get_elements` at 5:32:53 PM and 5:32:55 PM. It features a `STOP` button and a `6340` value field.
- XMLRPC_UDPcdb_rec.vi**: A receiver window with a `Send` button and a `STOP` button. It displays an `XMLRPC String` with headers like `User-Agent: LabVIEW (XML.bin.RPC - LC)` and a `<methodCall>` block.
- XMLRPC_UDPcdb_sdr.vi**: A sender window with fields for `Port` (58432), `Multicast Address` (234.5.6.7), and `Send to Port` (58432). It includes `Send` and `STOP` buttons. The `Data Received` field shows `synch_me`.
- XMLRPC_UDPcdb...**: A window showing `served methods` with `synch_me` listed.

The system tray at the bottom right shows the time as 5:34 PM. The Windows taskbar includes the Start button and several LabVIEW application icons.