

CLOUD COMPUTING AND VIRTUALIZATION IN EMBEDDED DEVICES SPACE USING WCF

Branislav T. Jevtović

*Business school of professional
studies - Blace, Serbia*

Danilo J. Oklobdžija

*Business school of professional
studies - Blace, Serbia*

Vladica S. Ubavić

*Business school of professional
studies - Blace, Serbia*

Srdan M. Tasić

Trion Group d.o.o. - Belgrade, Serbia

Abstract

This paper shows the appliance of WCF Windows Communication Foundation technology for communication with embedded devices. It defines the demands an environment should fulfill in order to enable an efficient integration of embedded devices in a heterogeneous environment like the Web. In a service oriented environment, shown in this paper, WCF services represent the basis for a middleware level in interaction between clients and embedded devices. Concrete implementation, with LabJack embedded device, of the described solution is realized for the education of students in the Business school of professional studies – Blace.

Keywords: Virtualization, Cloud computing, WCF - Windows Communication Foundation, LabJack.

INTRODUCTION

Virtualization has been a hot topic in the enterprise space for quite some time, but has recently become an important technology for embedded systems as well. It is therefore important for embedded-systems developers to understand the power and limitations of virtualization in this space, in order to understand what technology is suitable for their products. [1].

Virtualization is a term that refers to the abstraction of computer resources. Simply put, virtualization is a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications or end users interact with those resources. The concept of virtualization is very broad and can be applied to devices, servers, operating systems, applications and even networks.

Virtualization has been a major factor in the enterprise space for years; several virtualization providers have become billion-dollar companies. More recently, virtualization has started to become a hot topic in the embedded systems areas. This is demonstrated by recent moves of the dominant providers of virtualization for the enterprise, VMware and Citrix, who are clearly developing business strategies

for that space [2]. Cloud and virtualization are interlinked in much the same way cars and spark plugs are interlinked - they are different layers of a comprehensive service.

Cloud computing (referring to infrastructure services rather than SaaS) is about more than VMs on-demand; it's about turning data centers into online services. All the infrastructure components you'd have deployed in a physical data center in the past now have to be exposed programmatically for the cloud to accommodate your applications. Security, storage, networking, life-cycle control, inventory, HA are all required to be part of the cloud. This is obviously quite a different technology space from virtualization.

Cloud computing could not have happened without virtualization as the complexities of trying to expose a traditional data center as an online service are too complex to have ever been reliable. However, as cloud matures, VMs will disappear under the hood [3].

WCF – WINDOWS COMMUNICATION FOUNDATION

The global embrace of Web services-standard protocols for application-to-application communication has changed software development.

The functions that Web services can now provide, including effective security, distributed transactions, and reliable communication, enable this single technology family to address what were once separate problems. And the new view of services as a distinct software abstraction provides a solid foundation for the service-oriented architecture (SOA) that many organizations are putting in place.

Software development environments must keep pace with these changes. The benefits Web services bring should be reflected in the tools and technologies that developers use. Going forward, a simpler approach to distributed computing, broad interoperability, and direct support for service-orientation will be fundamental for developers.

Windows Communication Foundation (WCF), Microsoft's forthcoming technology for service-oriented applications, is designed to address these requirements.

WCF is implemented primarily as a set of classes on top of the .NET Framework's Common Language Runtime (CLR). Because it extends their familiar environment, WCF allows developers who create object-oriented applications using the .NET Framework today to also build service-oriented applications in a familiar way [4].

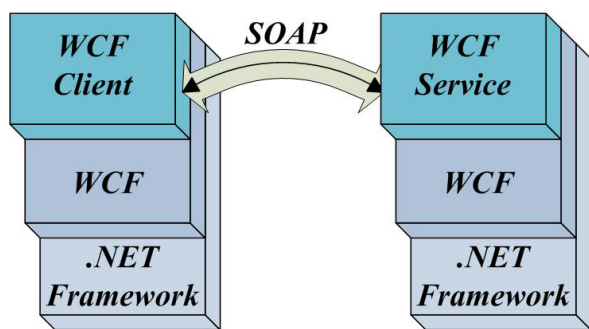


Fig. 1. WCF Client/Server model

The figure 1 shows a simple view of a WCF client and service. The two interact via SOAP, WCF's native protocol, so even though the figure shows both parties built on WCF, this certainly is not required. WCF addresses a range of problems for communicating applications. Three things stand out, however, as WCF's most important aspects:

- Unification of existing .NET Framework communication technologies;
- Support for cross-vendor interoperability, including reliability, security, and transactions;

- Explicit service orientation.

While WCF introduces a new development environment for distributed applications, it's designed to interoperate well with the non-WCF world. There are two important aspects to WCF interoperability: interoperability with platforms created by other vendors, and interoperability with the Microsoft technologies that preceded WCF. Because WCF's fundamental communication mechanism is SOAP, WCF-based applications can communicate with other software running in a variety of contexts. As shown in the figure 2, an application built on WCF can interact with all of the following:

- WCF-based applications running in a different process on the same Windows machine
- WCF-based applications running on another Windows machine

Applications built on other technologies, such as J2EE application servers, that support standard Web services. These applications can be running on Windows machines or on machines running other operating systems, such as Sun Solaris, IBM z/OS, or Linux.

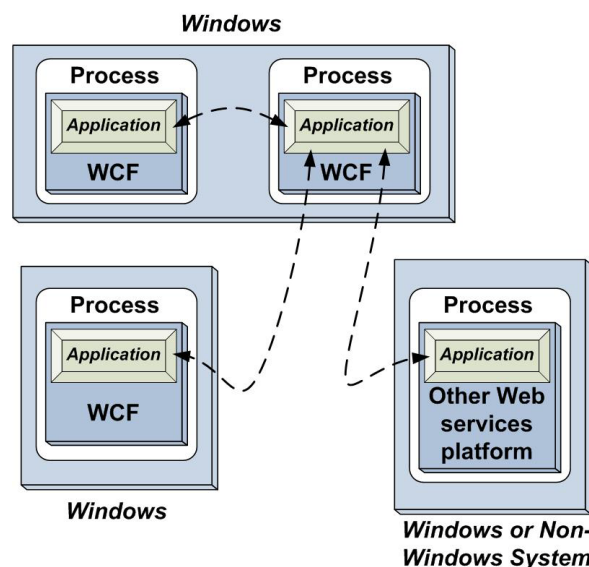


Fig. 2. WCF applications

To allow more than just basic communication, WCF implements Web services technologies defined by the WS-* specifications. All of these specifications are ownership by standards bodies such as the Organization for the Advancement of Structured Information Standards (OASIS). WCF supports all the specifications shown in figure 3. Grouped by function, those specs are:

- Messaging
- Metadata
- Security
- Reliability
- Transactions

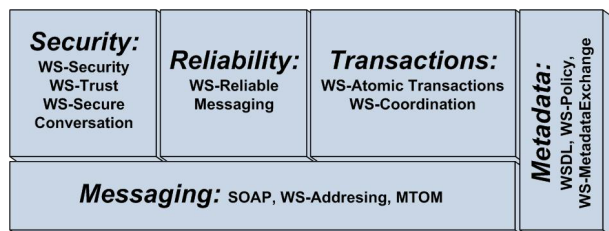


Fig. 3. WS- specifications supported by WCF*

EMBEDDED DEVICES FRAMEWORK

Today, and this will certainly be a trend in future, Internet represents the most significant technology for global integration of documents, knowledge, people and devices. Furious development of wireless and mobile communications, software, embedded systems and other technologies have significantly contributed to mass usage of the Internet as a transport medium in all areas of human activeness. Already today, many authors see the Internet, especially the Web, as a central server which connects most different resources.

This paper shows the requirements that a certain framework should fulfill in order to enable efficient integration of embedded devices into a heterogeneous environment like the Web. Modularity and service architecture of the represented framework make implementation of new technologies into specific solutions easy possible.

A framework, which aims for a fast and easy integration of embedded devices into a distributed environment, must fill out a large number of functional and nonfunctional requirements. Basic nonfunctional requirements are:

- A framework must be modular, expandable and flexible enough to provide possibility of interaction with a wide spectrum of hardware and software environments.
- A framework must have the capability of total control over the embedded device, either locally over the LAN network or globally over the Web.
- A framework must provide reliability, considering functionality as well as safety.

Safety mechanism must be one of the components of the framework. Authentication, authorization and administration of users, which interact with the device, are mandatory functionalities of every specific implementation of the represented framework

From architectural point of view components of the distributed framework, for interacting with embedded devices in network environment, can be classified in three basic layers [5]. Those are:

- Client application – corresponds to presentations layer in the classic N – layer architecture. Beside responsibility for user authorization and authentication, client application is responsible for interaction with a WCF service. Interface with the WCF service is realized by adding reference to the service proxy class. By using AJAX technology, periodically interaction with embedded device can be realized without user's intervention. AJAX and WCF service technologies present excellent platform for developing WEB based SCADA (Supervisory Control and Data Acquisition) systems. In the shown framework, the whole interaction between a user (browser, a client application) and embedded devices is done by exchanging XML messages i.e. by using "semantic" commands.
- WCF service i.e. middleware layer, is the place where all logical rules, for concrete application are realized. Validation of "semantic" commands and their transformation into a corresponding byte stream are realized at this layer. Hiding low level details of hardware and binary command set from client, WCF service provide infrastructure for embedded devices integration. In this manner WCF service provides "semantic" interaction between clients and embedded devices. WCF service main role is to enable distributed middleware layer creation. In the distributed network applications middleware layer seeks primarily to "hide" the underlying networked environment's complexity, by insulating applications from explicit protocol and network resources handling. Owing to middleware layer high level of abstraction is achieved during application design, by "hiding" complexity which is inherent to distributed network environment. Further,

middleware “masks” the heterogeneity of computer architectures, operating systems, programming languages, and networking technologies to facilitate application programming and management. In other words, middleware is “glue” between distributed application, from one side, and networks functions which are responsible for sending messages to remote network components, from other side. Those remote components can be other distributed applications, servers or embedded devices with appropriate sensors and actuators.

- Embedded application, which is a part of embedded device firmware, is responsible for communicating with hardware on binary level. Beside binary communication embedded application is responsible for sensors status monitoring and actuators control.

LABJACK

The LabJack U12, Figure 4, is a low-cost USB based analogue and digital I/O unit for data logging, data acquisition, measurement, and control applications.



Fig. 4. LabJack

The LabJack U12 is a measurement and automation peripheral that enables the connection of a PC to the real-world [6]. The hardware and software features of the LabJack U12 are:

- Analog inputs (AI0-AI7). These can be configured individually as 8 single-ended channels, 4 differential channels, or combinations in between. Each input has a

± 10 volt input range with 12-bit resolution. Differential channels can make use of the low noise precision PGA to provide gains up to 20, giving an effective resolution greater than 16-bits.

- Analog outputs. The LabJack U12 has 2 screw terminals for analogue output signals (AO0 & AO1). Each analogue output can be set to a voltage between 0 and the supply voltage (+5 volts nominal) with 10-bits of resolution. The analogue outputs are controlled in command/response mode at up to 50 Hz per channel.
- Digital I/O. The LabJack U12 has 20 digital I/O channels which can be individually configured as input or output. Connections to 4 of the digital I/O are made with the built-in screw terminals (IO0-IO3). These 4 channels have built-in overvoltage/short-circuit protection. As inputs or outputs, they are controlled/read in command/response mode at up to 50 Hz per bit. As inputs only, they can be read with the high-speed burst and stream modes. The remaining 16 digital I/O are accessed through the DB-25 connector and controlled/read in command/response mode at up to 50 Hz per bit. These channels can sink or source up to 25 mA each (total sink or source current of 200 mA max for all 16), allowing direct interface to many relays, such as the RO-16 board.
- Counter. There is one 32-bit counter available on the LabJack U12 (screw terminal CNT), capable of counting frequencies up to 1 MHz. The counter is read in command/response mode at up to 50 Hz.
- Testing and calibration terminals CAL – STB. These terminals are used during testing and calibration. CAL is a precision 2.5 volt reference, and can be used during normal operation, but care should be taken to observe the current limits specified in Appendix A. The CAL pin is protected from ESD and overvoltage, but severe overvoltage (steady-state or transient) can damage CAL, and result in the failure of all analog inputs.
- Watchdog Timer. The LabJack U12 also has a watchdog timer function available which can change the states of digital I/O if the LabJack does not successfully

communicate with the PC within a specified timeout period. This function could be used to reboot the PC allowing for reliable unattended operation.

- LabJack software include Software Toolkit (DLL, ActiveX) for Windows 98/2000/NT/XP, Many Visual Basic, C++, Delphi etc. examples, Software Toolkit for Labview, Wonderware Intouch, DAQFactory Driver and examples.
- LabJack hardware and software support 38 functions exported by the LabJack DLL, and matching functions in the OCX and LabVIEW VIs. To maintain compatibility with as many languages as possible, the every attempt has been made to keep the parameter types very basic

CASE STUDY

In accordance with the principles of design of distributed business environment for embedded devices networking, shown in the previous chapter, a software module LabJackWCF Service has been realized.

Namely Labjack hardware, and the appropriate commands supported by basic driver (LJDotNet.dll) represent embedded level of realized system. At this level, communication between a embedded device and users is done at the binary level. The main role of the WCF service is to enable the “semantic” approach to the basic features of the hardware. LabJack WCF service is therefore middleware level that allows the use LabJack hardware from remote locations.

LabJackWCFService has been realized by Microsoft Visual Web Developer 2008 Express Edition [7] tool, for the education of students in the Business school of professional studies – Blace. Starting from the core generated by a New WCF service wizard, a complex module that contains methods for all functions LabJack hardware has been realized. Semantic communication between users and the LabJack hardware is enabled by the set of public Web methods. This means that Web methods represent wrapper about the basic functions of LabJack devices. According to the functions these Web methods were divided into several groups. These are:

- Methods that allow LabJack device identification and configuration
- Methods for analog inputs
- Methods for analog outputs
- Methods for digital lines configuration
- Methods for reading digital inputs
- Methods for setting of digital outputs
- Methods for working with the counter
- Other service methods.

The Listing 1 provides an example of a Web method that recognizes all LabJack devices connected to your computer. The IDs of detected devices are stored in the int array that was passed to the Web methods as parameter by reference, during the call procedure.

```
[WebMethod(Description = "Detect all
LabJacks")]
public int AllLabJacks(ref int[]
pera)
{
    int[] productIDList = new
int[127];
    int[] serialNumList = new
int[127];
    int[] localIDList = new int[127];
    int[] powerList = new int[127];
    int[,] calMatrix = new int[127,
20];
    int numFound = 0;
    int reserved1 = 0, reserved2 =
0;

    // Call the ListAll function.
    int result =
LabJack.ListAll(productIDList,
serialNumList, localIDList,
powerList, calMatrix, ref numFound,
ref reserved1, ref reserved2);

    if (result != 0)
        return result;
    // count how many we found and set
the array
//which will be returned to contain
valid IDs

    int i = 0;
    foreach (int id in localIDList){
        if (id != 9999)
        {
            pera[i] = id;
            ++i;
        }
    }
    // return that array
    return result;
}
```

Listing 1. Web method example

As host application for the LabJackWCF Service module the Internet Information Service (IIS) has been chosen. This allows access to the Web method from a remote location, which was one of the proclaimed objectives. During the testing period, students have developed a number of clients' applications in order to test these functions. Many of these applications were implemented by using ASP pages that support AJAX technology.

CONCLUSION

The framework, shown here, enables design of service oriented applications with loosely coupled software components which can be independently developed and implemented. Only basic knowledge about characteristics of the framework and implementation mode of the other components is needed for successful operation of the loosely coupled components. Realized system also allows the user to visualize functions of LabJack embedded device. Namely, in the space of embedded devices, visualization is a synonym for device characteristics abstraction. Simply put, virtualization is a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications or end users interact with those resources.

The authors have decided to choose the Windows Communication Foundation (WCF) as the base technology for the development LabJackWCFService module. The main reasons for this lie in the great possibilities of interoperability of these technologies, as well as the ease of its implementation for user friendly development environment. For middleware layers in distributed service oriented application WCF offer unification, great functionality and significantly reduced complexity.

Concrete implementations of the represented framework enable the usage of WCF as platform for successful interaction, not only

communication, with LabJack embedded devices. Realized Web methods allow full integration of LabJack embedded devices in the complex data acquisition systems, where the communication between users and devices perform at the level of "semantic" commands.

REFERENCE

- [1] Gernot Heiser, "Virtualization for Embedded Systems", Open Kernel Labs, Inc., Document Number: OK 40036:2007, November 27, 2007, Available: http://www.ok-labs.com/_assets/image_library/OK_Virtualization_WP.pdf [Accessed: January 2010].
- [2] Gernot Heiser, Embedded systems virtualization: Consider a Hypervisor, Embedded.com – The Official site of the Embedded Development Community, Available: <http://www.embedded.com/design/212902574> [Accessed: January 2010].
- [3] Dan Kusnetzky, "Does virtualization equal cloud computing?", ZDNet News & Blogs, November 25th. 2009, Available: <http://blogs.zdnet.com/virtualization/?p=1475> [Accessed: January 2010].
- [4] Early Look, "Introducing Windows Communication Foundation", David Chappell, Chappell & Associates, September 2005, Available: <http://www.davidchappell.com/introducingwcfv1.2.1.pdf> [Accessed: January 2010].
- [5] Branislav Jevtović, Danilo Oklobdžija and Jovan Oklobdžija, "APPLYING OF AJAX TECHNOLOGY IN INTEGRATION OF EMBEDDED DEVICES INTO A WEB ENVIRONMENT", Elektronika-Electronics, Elektrotehnički fakultet Univerziteta u Banjoj Luci, Vol 12, Number 1, 1. Jun 2008, YU ISSN 1450-5843, pp 12-15
- [6] Audon Electronics, "LabJack U12 USB Data Acquisition", Available: <http://www.audon.co.uk/labjack.html> [Accessed: January 2010].
- [7] Microsoft, "Visual Web Developer 2008 Express Edition", <http://www.microsoft.com/express/Default.aspx> [Accessed January. 2010].