VOL. 10, NO 22, DECEMBER, 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

WEB SERVICES ORIENTED ARCHITECTURE FOR DYNAMIC EVOLUTION OF COMMUNICATION WITH EMBEDDED SYSTEMS

Smt. J. Sasi Bhanu[1], A. Vinaya Babu[2] and P. Trimurthy[3]
[1]Department of Computer Science Engineering, KL University, Vaddeswaram, Guntur District, India
[2]Department of Computer Science and Engineering, JNTU, Hyderabad, India
[3]Department of Computer Science and Engineering, ANU, Guntur, India
E-Mail: sasibhanu@kluniversity.in

ABSTRACT

A HOST is a Computer system that needs to be situated at a remote location preferably connected through internet for communicating with an embedded system which is meant for monitoring and controlling a safety or mission critical system. The communication between the HOST and the embedded system must be dynamically evolvable if the changes to the communication system must be effected while the embedded system is up running. The main issue that must be addressed while attempting to implement dynamically evolvable embedded system is to ensure that the entire system must be light weight due to the availability of limited resources with the embedded systems. Implementing the ES software related to syntax, semantics, online testing and communication components along with components that are required to make the entire system dynamically evolvable is the issue that can be addressed through use of WEB services related technologies. Architectural frameworks are required to explain how the entire dynamic evolution system can be implemented through use of WEB services oriented technologies. This paper is aimed at determining various web services oriented architectures and selecting the best that suits the dynamic evolution of communication with an embedded system.

Keywords: dynamic evolution, communication systems, embedded systems, web services, safety and mission critical systems.

1. INTRODUCTION

The overall architecture of dynamic evolution of embedded from the point of view of evolution of syntax, semantics and online testing has been shown in the Figure-1. The first layer in this model is the communication sub-system. Communication between the HOST and the TARGET is initiated from the HOST through commands strings which follow the UNIX like standard. The physical connection between the HOST and the embedded system can be achieved in many ways. Sastry [1, 2] have presented the way peer to peer communication between a Target and HOST can be achieved using wireless communication methods which include Wi-Fi and Bluetooth which have distance limitations.

Embedded systems can also be connected to a HOST using SPI and interfacing the same with a I^2C, USB, RS425, CAN or Multi port interface [3, 4, 5, 6, 7, 8, 9]. A HOST communication with the TARGET using serial port communications systems such as I^2C, USB, RS425, CAN or mulit port is limited by the distance to a maximum of 1200 Meters.

An embedded system can be connected to an Internet so as to increase its distance from the HOST. Use of Ethernet interface and use of TCP/IP or UDP based communication helps establishing communication between the HOST and the Target. Both HOST and the TARGET can be situated at longer distances, thus meeting the primary requirement of Safety and Mission critical systems.

Communication between the HOST and the Target using the internet can be achieved through implementation of an email extension server, WEB server or a WEB service server using TCP/IP communication protocol. The way the communication between the HOST and the TARGET can be archived through implementation of a WEB service server within the target system has been the main focus of this paper as it allows communication using the OPEN standards which help in dynamic evolution of communication system.

Communication between the HOST and the Target can be effected by several protocols such as TCP/IP, HTTP, UDP, X.25 etc. The target can host several services and a different communication protocol is necessary for utilizing a WEB service. Thus the communication system must evolve dynamically based on the type of protocol needed by a service which is initiated by the HOST for execution by the TARGET.

The implementation of dynamic evolution models requires fairly more memory resources and faster processor. The modern Microcontroller based systems no doubt can run a dynamically evolvable embedded system due to the availability of more resources. However, low powered microcontroller systems may not be able to take much load. Sometimes it is worthwhile idea to move some of the processing load to the HOST and keep the embedded system light weighted. The dynamic evolution of the communication system however is needed so that communication with the WEB services can be done as per the protocol used by it.

Use of web services technology addresses both the requirements of dynamic communication system and the lightweight embedded system. Implementation of dynamic evolution using the WEB services Technologies is necessary as they support OPEN standards which is the true means of evolving dynamic evolvable systems.

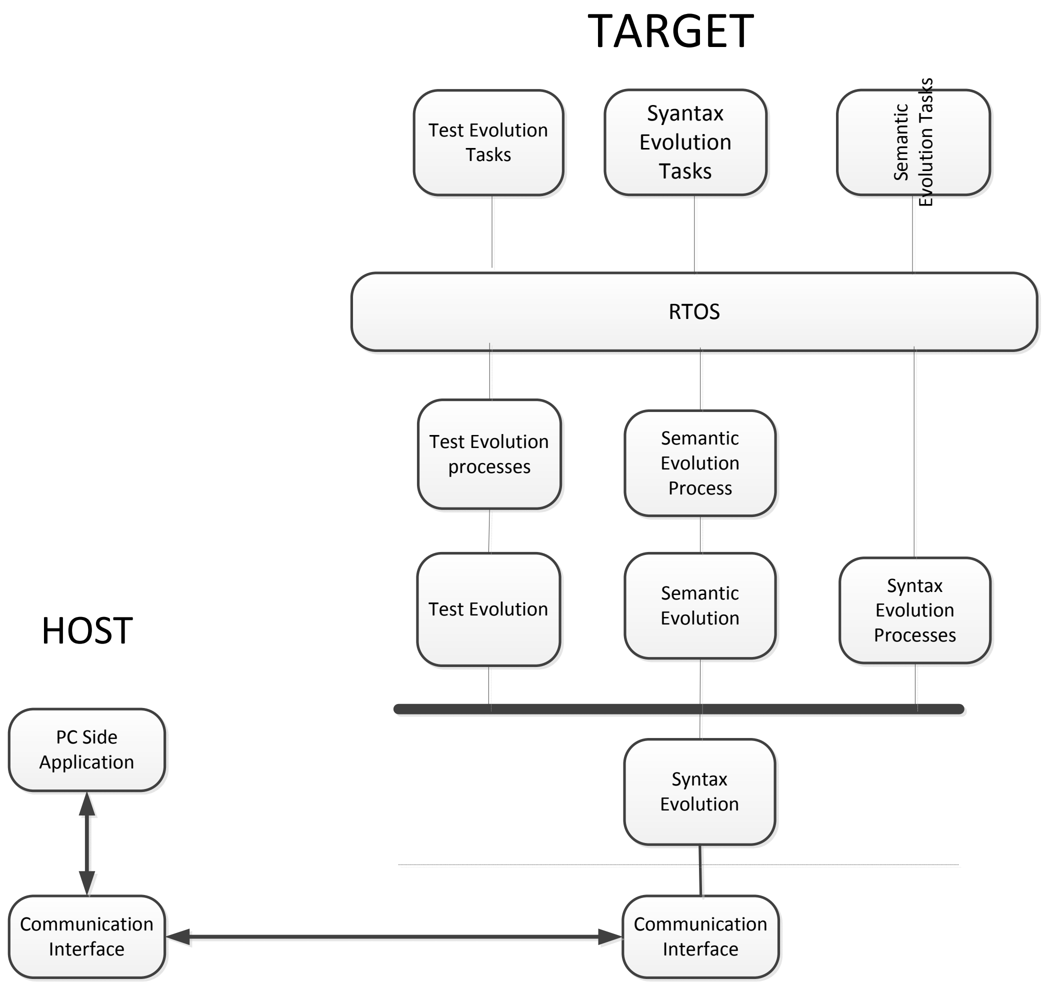

Figure-1. Overall architecture of dynamic evolution of embedded systems.

There are many architectures that support dynamic evolution using the WEB services. It is necessary to find the best of the architectures that support dual purposes of OPEN communication and dynamic evolution. Dynamic evolution is the ability to make changes to any of the software components while the system is up and running.

The general web service architecture is shown in the Figure-2. There are two ways to view the WEB service architecture. The first is to examine the individual roles of each WEB service actor; the second is to examine the emerging Web service protocol stack.

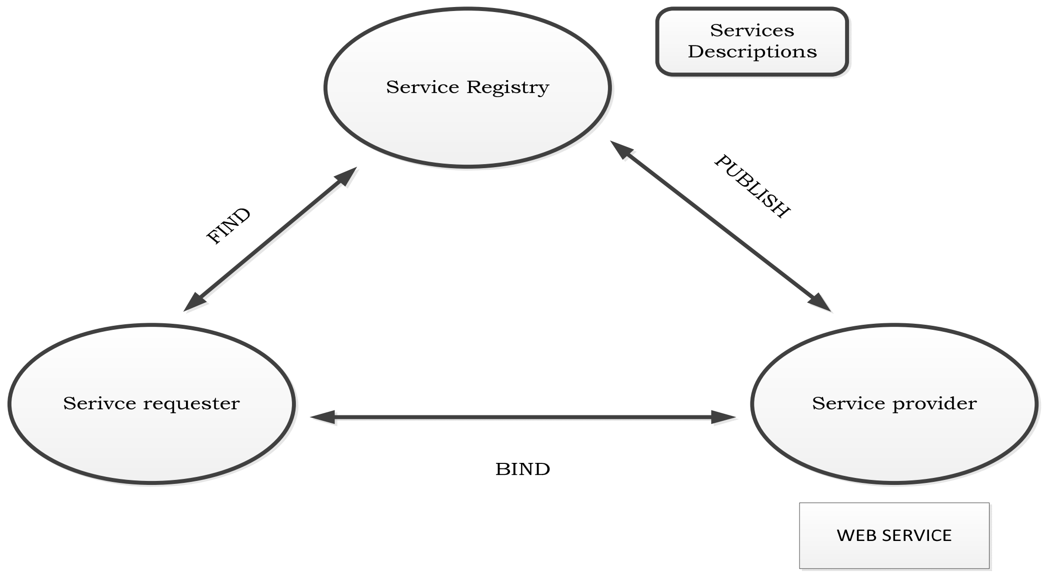

Figure-2. WEB service architecture.

There are three major roles within the WEB service architecture which include service provider, service requester, and service registry. Service provider provides the service on a server which is connected to the Internet by using standard protocols. Service requestor uses the WEB service for implementing various types of applications. The requestor utilizes an existing Web service by opening a network connection and sending an XML request. Service registry is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearinghouse for companies and their services.

A second option for viewing the WEB service architecture is to examine the emerging Web service protocol stack shown in Figure-3.

The stack is still evolving, but currently has four main layers. The first service is Service transport layer. This layer is responsible for transporting messages between applications. Currently, this layer includes hypertext transfer protocol (HTTP), Simple mail transfer protocol (SMTP), file transfer protocol, such as Blocks Extensible Exchange Protocol (BEEP) etc. In the second layer, XML messaging is used for encoding messages in a common format so that messages can be understood at their end. Currently, this layer includes XML-RPC and SOAP. The third layer is service description layer which is responsible for public interface to a specific WEB service. Currently, service description is handled via the WEB service description language (WSDL). Service discovery is the fourth layer which is responsible for centralizing services into a common registry, and providing easy publish/find functionality. Currently, service discovery is handled via Universal Discovery, and Integration (UDDI).

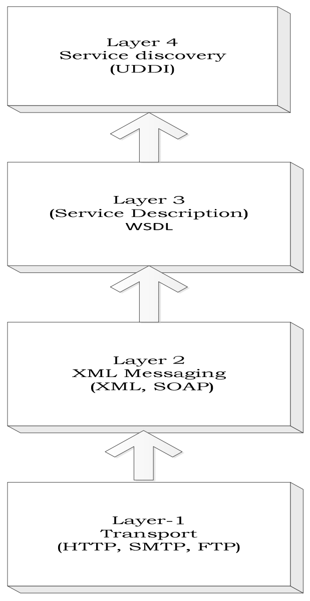

Figure-3. Web service protocol stack.

Several technologies are being used for implementing WEB services which include UDDI, XML, SOAP and WSDL. UDDI (Universal data discovery interface) forms part of the discovery layer within the WEB service protocol stack. UDDI is a technical specification for building a distributed directory of business and Web services. Data is stored within a specific XML format. The UDDI specification details API for searching existing data and publishing new data. UDDI is implemented through building a business registry which fully implements UDDI specification.

XML is a language that is described over open standard and enables diverse computer system to share data most easily, regardless of operating system or programming language. There are dozens of XML tools, including parsers and editors that are available for nearly every operating system and every programming language, including java, C#, C, C++ etc.

XML-RPC is a simple protocol that uses XML messages to perform RPCs. Requests are encoded in XML and sent via HTTP POST. XML responses are embedded into the body of the HTTP response. Because XML-RPC is platform independent, it allows diverse applications to communicate. For example, a java client can speak XML-RPC to a perl server.

SOAP is XML based protocol for exchanging information between computers. Although SOAP can be used in variety of messaging systems and can be delivered via a variety of transport protocols, the initial focus of SOAP is remote procedure calls transported via HTTP. SOAP therefore enables client application to easily connect to remote services and invoke remote methods. For example, a client can immediately add language translation to its features set by locating the correct SOAP service and invoking the correct method. SOAP specification defines three major parts:

WSDL is a specification meant for describing WEB services in a common XML grammar. WSDL describes four different aspects which include Interface information describing all publicly available functions, Data type information for all message requests and message responses, Binding information about the transport protocol to be used and addresses information for locating the specific server.

In a nutshell, WSDL represents a contract between the services requestor and the service provider, in much the same way that the java interface represents a contract between client code and the actual java object. The crucial difference is that WSDL platform is language independent and is used primarily to describe SOAP services. Using WSDL a client can locate a WEB service and invoking any of its publicly available functions. With the WSDL-aware tools, one can also automate this process, enabling applications to easily integrate new services with little or no manual code. WSDL therefore represents a cornerstone of the WEB service architecture, because it provides a common language for describing services and a platform for automatically integrating those services.

UDP and TCP are standard, well-supported protocols for computers that need to send and receive messages with in local network or on the internet. Many application protocols transfer information using UDP or TCP. For example, a computer that sends a request for an IP address to a DNS server places the request in a UDP datagram. A request to a server for a Web page and the page sent in response both travel in TCP segments. UDP and TCP can also be used to transfer messages of any type, including information in application-specific formats. In general, UDP is a simpler protocol to implement but has no built-in support for acknowledging receipt of messages, determining the intended order of messages, or flow control.

Communications that use UDP, TCP, or other Internet protocols must use IP addresses to identify the sender and receiver of the communications (with the exception that a UDP datagram doesn’t have to specify a source address). In addition, sending a message using IP may require a net mask value, the IP address of a gateway, or router, and the IP address of a domain-name server. The device firmware may specify these values, or the device may request the values from a DHCP server. Many standard application-level protocols also use TCP or UDP when exchanging information. One of the most popular of these is the hypertext transfer protocol (HTTP), which enables a computer to serve Web pages on request.

Every embedded board can be provided with data related to IP address, network mask, logical port number stored in its EEPROM. The Ethernet port can be binded with the TCP/IP address and suit a TCP/IP function calls can be made available as callable functions as regular functions that can be called from an ES application

2. PROBLEM DEFINITION

Implementing WEB services server within an embedded systems leads to extensive automation and reuse. Implementation of WEB services requires huge amount of storage area processing power and porting of the technologies which are required for interacting with internet world. Embedded systems are low in resources and therefore throw a challenge to implement light-weight embedded WEB services that can cater for the user requirements. The main problem is to develop simple WEB services architecture that requires very less resources on the embedded system side which is expected to host all the services required to complete the tasks initiated by the HOST and returns the results achieved out of execution of the tasks.

Many tasks which together forms embedded system software (TARGET) are required for implementing dynamic evolution of syntax, semantics, and online testing. The tasks are to be activated based on the commands initiated from the HOST and the results obtained out of execution of the tasks are to be returned to the HOST which gets connected to the TARGET through internet. Dynamic evolution is the ability to make changes to the software components while the embedded system is up and running.

The commands initiated from the HOST can be enclosed into a SOAP message which is formatted in XML language and transmitted. The SOAP message is received and parsed and the command is retrieved. Based on the command received, one of the services related to Syntax Evolution, Semantic evolution, online testing or communication system can be initiated.

An efficient WEB services oriented architecture is required to implement a communication system that is dynamically evolvable and also be able to execute the tasks which are, kind of services that can be invoked and executed in seamless manner. On the embedded system side, all the components related to syntax, semantics, online testing and those components that are responsible for dynamic evolution of the operating components must be co existent with proper interfaces. The components and the interfaces between them must be light weight and must be operable within the limited resources.

Thus the problem is to find an efficient WEB services oriented architecture which is simple to implement and that requires very few resources. One important way is to explore pushing much of the processing to the HOST and just the services are implemented on the target side. The implementation must be done using the platforms that require few resources. To the extent possible, it is necessary to develop open architecture so that the overheads that one has to face when technologies like TOMCAT, WEBLOGIC SERVER, JBOSS etc. gets completely eliminated.

3. LITERATURE SURVEY

Michael Sig Birkmose [10] mainly described the applicability of WEB services in distributed embedded systems environment. The first and foremost thing that one needs to find is, how the overhead associated abstractions provided by WEB services can be minimized. The extra overhead is caused by the extra size of the messages that must be transferred, when compared to the actual size of data carried within such message. Stanislav Sliva [11] stated that distributed processing plays a major role in applications whose parts (Procedures) are executed in local nodes and in remote nodes distributed in a network. Distributed processing applicable to embedded systems describes several possibilities of using distributed computations in an embedded environment. The major part of the distributed processing applicable to embedded systems is focused on a description of WEB services and related protocols like SOAP, XML-RPC.

Kevin J [12] have presented their observations in integrating a WEB service infrastructure into a Simple Network Management Protocol (SNMP). SNMP has been used predominantly for the development of networking equipment which are developed using embedded systems. WEB services based approach allows enhancing their existing applications with XML/SOAP interoperability, SSL/TLS security and the potential to migrate both application and protocol layers to encompass future extensions and WEB browser accessibility. The difficulty with SNMP, and many other legacy networking protocols, is that they are all installed on legacy hardware. Their vision for a WEB service-based SNMP has led to non-intrusive extension to the existing protocol. They have proposed a scheme for extending and improving the existing SNMP v2 infrastructure through the use of WEB services at the transport level using an XML encoded SOAP message encapsulation and bound to HTTP for SNMP transport. They investigated two options for implementing these extensions which include, using standard WEB server and java tool set, and another being using light weight HTTP/SOAP stack. Both were integrated with an existing SNMP daemon and tool set. Their main aim was to achieve interoperability between the two approaches, as well as maintain interoperability with legacy systems. Each scheme has unique performance and feature characteristics, and both provide SNMP with the benefits of the WEB Services.

Nikolay Kakanakov [13] mainly discussed the possibility of adaption of WEB service architecture (WSA) for implementing distributed embedded systems. The WSA integrates the best aspects of component-based development and World Wide Web. According to R. Pallavi [14] a service is: “a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the WEB services in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.” Applications access WEB services via common WEB protocols and data formats. The most common used protocol for transferring data in the Internet is HTTP (Hypertext Transfer Protocol) and it is the key transport protocol in the WSA. The universal schema in the Internet is the one that codes the data is XML (Extensible Mark-up Language). The current structure of Internet is based on program-to-user interaction and the WSA is based on program-to-program interaction [15]. The WSA is based on some key standards: XML for data representation; SOAP for accessing services; WSDL for describing services; UDDI - registering and discovery of services.

Daniel Schall [16] have mainly discussed the capabilities of embedded devices such as smartphones that provide flexibility of data access and collaboration with other smart phones around while being mobile. From the distributed computing point of view, fundamental issues in mobile computing include heterogeneity in terms of varying device capabilities (i.e., operating systems and various hardware platforms), performance characteristics and real-time behaviour, and the ability to discover and interact with peers seamlessly. WEB services are a family of XML based protocols to achieve interoperability among loosely coupled networked applications. They proposed the use of WEB services on embedded devices in order to solve interoperability issues in distributed mobile systems. They discussed various toolkits available for embedded devices and investigate performance characteristics of embedded WEB services on smartphones. The goal is to guide the design of WEB services based applications on mobile devices, and provide estimates of performance that can be expected.

Guilherme Bertoni Machado [17] have presented that the Embedded applications, which were originally built on standalone devices, nowadays require a growing integration with other systems through their interconnection with TCP/IP networks. WEB Services, which provide a service oriented distributed architecture for the interconnection of systems through TCP/IP networks, have been widely adopted for the integration of business applications, but this sort of integration is still not provided by embedded applications. They demonstrated the feasibility of using WEB Services for the integration of embedded applications running on heterogeneous architectures. This is achieved through the provision of a support for the development and deployment of WEB services on embedded platforms. The feasibility of this approach is demonstrated by developing an application deployed on an embedded platform - the SHIP board - which is then integrated with a distributed enterprise application.

Mitko Petro Shopov [18] have said that one of the most promising trends from the recent years - the service oriented architecture (SOA) is now emerging in the domain of distributed embedded systems. One of the most important benefits of such a shift is the possibility to replace traditional vendor specific solutions with popular open standards for communication and to satisfy the arising need to connect distributed embedded devices within the network of enterprise systems. They have presented test-bed experiments for evaluation of WEB services implementation for ARM-based embedded system running embedded Linux 2.6. The gSOAP WEB services generation toolkit optimized for embedded devices is used. Two WEB services are developed for the experiments: Echo and Temperature. The WEB services are tested as a standalone application which Apache WEB server using CGI Interface. The services are tested with gSOAP and .NET WEB services clients.

David E. Culler [19] have explained that integrating diverse sources of information takes place at three levels. The bottom is the communication media that allows information and control actions to be exchanged. Given such an interconnection, the communicating participants must agree on how information is represented. The data required for transferring messages include format, data model, or object model. The information exchanges to be useful, it is necessary to discover the services that the devices are capable of delivering. Danilo J. Oklobdzija [20] pointed the need for new technologies for integrating embedded devices in heterogeneous distributed networks, and outline the perspectives opened by applying service-oriented paradigm for realizing interaction with embedded devices. As a foundation for interaction with embedded devices, XML and WEB services have been used. Strong integration power of XML and WEB services make the framework appropriate for unique approach to all resources of embedded device regardless of applied technology. WEB services based middleware, provides application designers with a high level of semantic abstraction, hiding the complexity of the applied technologies. Elmar Zeeb [21] introduced the WS4D initiative, a project that tries to provide a common open source platform for using DPWS (Device profile WEB services) in different environments. The usage of the Service Oriented Architecture (SOA) paradigm currently changes the view on many enterprise applications. SOA allows the creation of modular and clearly defined software architectures that ensure a high grade of interoperability and reusability. As even small, resource-constraint networked devices get more and more powerful it is common sense to try to adopt the SOA paradigms to embedded device networks. This idea is substantiated in the specification of DPWS, a standard that uses the primitives of the WEB services architecture (WSA) to create a framework for interoperable and standardized communication between embedded devices.

Risto Serg **[**22**]** have explained that most embedded systems are seldom used alone. Systems that communicate between each other are much more common in real world ubiquitous applications. If the embedded sensors and devices could directly work together and with other computing devices, they would add value to each other, and enable new consumer application. Present requirements of cyber-physical systems are usually too high for implementing them on single, non-networked units. Using service oriented architecture is one of the solutions to achieve interoperability and possible future scaling of the system. They have explained that a limited subset of XML WEB service protocols can be implemented in very limited environment. Surprisingly they found that limited XML WEB service implementation introduces only minimal overhead.

Yin-Wei-Feng [23] have said that engineering education necessitates the use of laboratories for measurement, data collection, analysis and design activities as well as for hands on experience of equipment, physical devices. Local laboratories are the traditional way of doing experimentation. Neither a virtual environment nor remote access can replace its function. But it has disadvantages like fixed time and place, limitation on the number of equipment sets and hence the number of students who can use them as the broadband connectivity to the Internet becomes common, Web based e-Learning have come to play an important role in self-learning, where learners are given much flexibility in choosing place and time to study. They have implemented remote laboratories. Remote laboratories are software environments that run experiments by interacting with real devices, which allow remote users to communicate with measurement devices and experiments set up to make experiments on a real system.

Elmar Zeeb [24] have presented that as the application of the Internet Protocol (IP) is no longer restricted to the internet and computer networks, future IP-based application scenarios require an enormous diversity of heterogeneous platforms and systems. Thereby emerging communication architectures, concepts, technologies and protocols must be capable of handling thousands of devices and communication endpoints on the one hand and be flexible and extensible enough on the other hand, to provide cross domain interoperability independent of platform specific constraints. The Devices Profile for Web Services (DPWS) is as such a cross domain technology. They provide an overview of DPWS and existing DPWS implementations and toolkits with special focus on the Web Service for Devices (WS4D) initiative. Therefore, features and capabilities of DPWS are described in detail by referring to the open source WS4D implementations. The target platforms are ranging from resource rich server platforms down to highly resource constrained embedded devices.

3.1 Drawback of existing methods

When WEB services oriented technologies are used, extra overhead is caused by the extra size of the messages that must be transferred, when compared to the actual size of the data carried within such messages. This includes abstractions on the underlying network protocols and serializations of the data. The complexity and verbosity of XML Web services protocols creates a whole new set of design trade-offs and issues for developing Web services applications for embedded systems. Embedded systems rarely have enough memory and processing power to run Web services. On the other hand, current Web services implementations do not adequately apply to the embedded processing. While that being the cases many of the embedded systems behave like server engaged in collecting the process data and process the same before the same are transmitted to a remote location. Some of the embedded systems can be designed to collect the environment data and transfer the same to those we need such data. A client can make a request to an embedded server which in term makes available the process data as a service to the client. Many applications exist that needs a model as above. The embedded systems suffer from lack of resources which are required for implementing the platform required to launch the services. Therefore there is a need to design a service oriented architecture that is light weight and still be able to support the entire platform required for implementing the WEB services.

4. INVESTIGATIONS AND FINDINGS

Different types of architectures are possible for implementing the WEB services for execution of software components within embedded systems. A review of possible architectures is needed to find the suitability of architecture for implementing dynamic evolution within the embedded system. Most efficient architecture that completely suits to an embedded system that implements dynamic evolution of syntax, semantics and online testing is needed. Just recommending architecture is not sufficient. An implementation mechanism that is feasible and implementable has to be determined and then must be applied to a alive application to determine proper working of the WEB services installed within the embedded systems. Different kinds of WEB service architectures that can be deployed are presented and then a comparative analysis is made to find the best suitable architecture to implement dynamic evolution of the embedded systems.

4.1 Embedded system as WEB service server

Figure-4 shows the architecture which includes all the components related to WEB server, SOAP processor, component for scheduling the services, WSDL scripts, and the application tasks whether real time or normal. In this case all the WSDL files are stored in the client in the HOST. SOAP messages are extracted from the WDSL files and sent to the embedded systems which are servers in this case. Client sends its XML/SOAP messages according to the required service type (previously known through the WSDL file located in the Embedded System and/or in a UDDI repository) through the HTTP protocol.

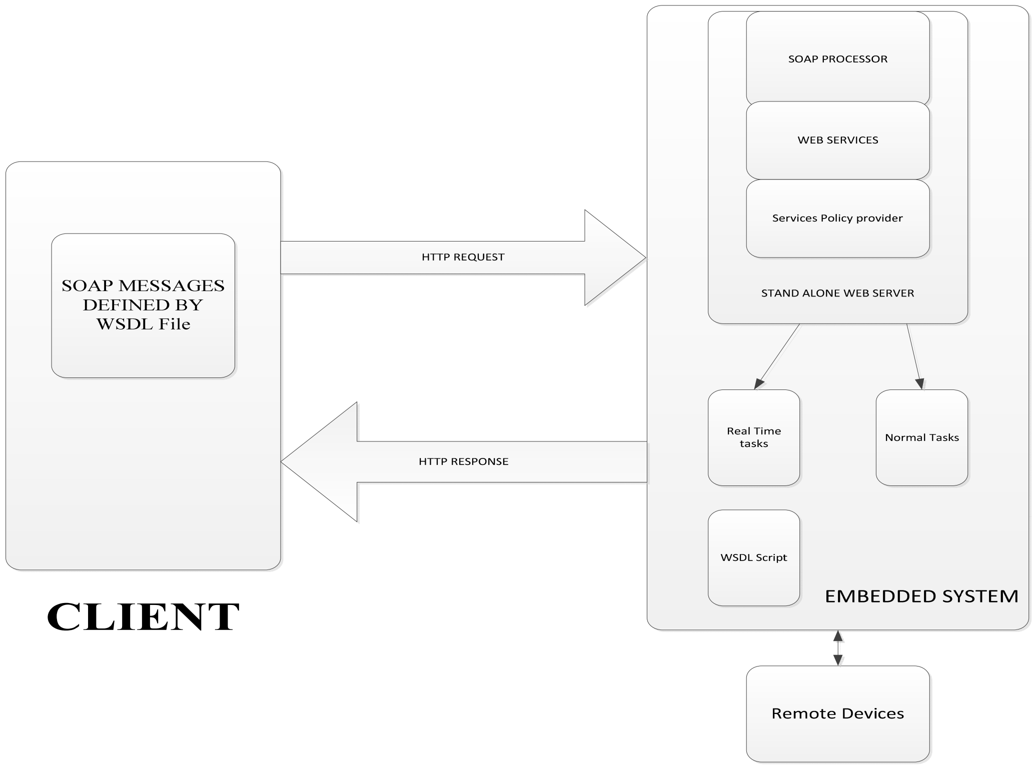

Figure-4. WEB services architecture.

In this architecture, the WEB service occurs in a pre-processing stage, for efficient scheduling of HTTP requests. The process that deals with "scheduling policy" tries to offer a fair chance that every request will be served based on the priority associated to each service. Then these messages will be processed by the microcontroller. After the execution, a reply is generated by the service which is returned to the client through http response. In this architecture the WEB server that hosts the web services is located within the embedded system. Thus requiring many resources. Even WSDL files are also to be stored within the embedded systems which require more storage area.

4.2 HOST as WEB service server

Figure-5 shows the architecture. In this case it is the HOST that is loaded with a web server under which the WEB services are deployed. Embedded systems keeps getting the inputs from the environment that it is controlling and makes a HTTP request to a HOST based server for undertaking a function such as updating a database with the sensed data, or informing a medical assistant, doctor or any other concerned using different types of gadgets such as a PC, Mobile phone either through direct connections or through internet.

In this architecture it can be seen that the embedded system only makes the request to the server for doing a service. An embedded system in it implement a UDDI registry to get the WSDL through an XML enquiry, parse the same and then makes a HTTP request through a SOAP message. In this architecture HTTP response is not received by the embedded system and as such embedded system is not providing any service to the remote host.

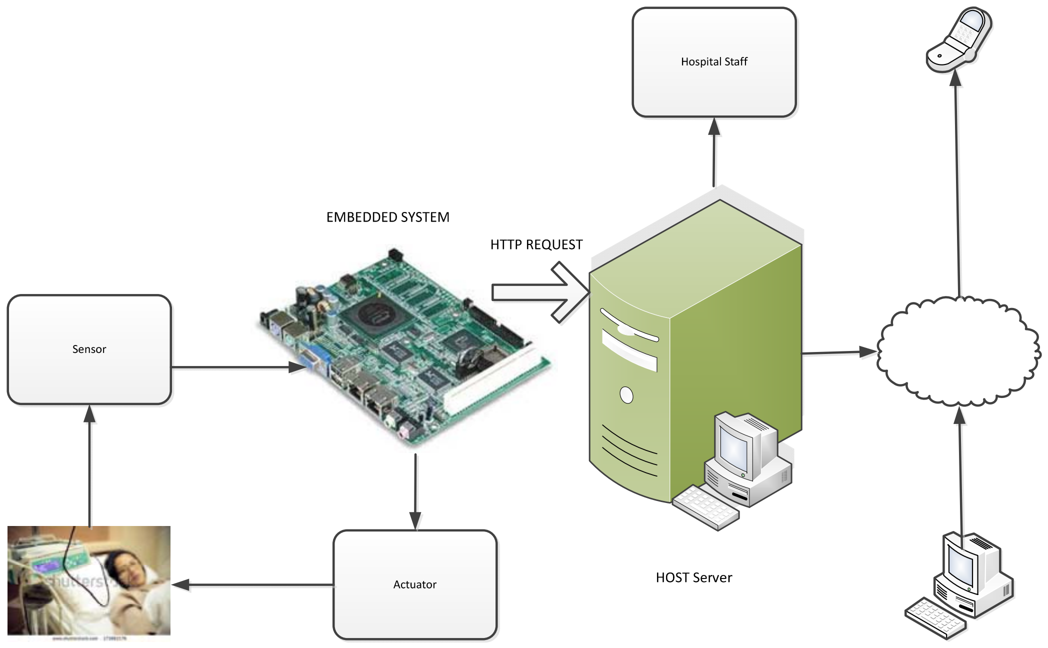

Figure-5. Typical implementation of WEB services.

The role of the application server is to store and distribute the diverse information coming from different sets of Embedded devices. The great advantage of this architecture is that, since the HOST is providing the WEB Services, the rest of the distributed environment has handled just the SOAP messages. Concerns regarding integration with legacy systems, database structure, the kind of software the client is using, etc. are irrelevant in this case as the communication between the embedded systems and the HOST is based on SOAP message which is open standard.

4.3 A three tier WEB services architecture

WEB Services present a way to interconnect applications through Internet among computational systems. Besides, its eminently open and standardized architecture provides WEB Services a great potential use in distributed computation. Therefore, nothing more natural than proposing the deployment of a WEB services in an embedded systems provide the integration of applications running on the embedded platform with other systems in a distributed environment. WEB Services as middleware to Embedded Systems integration helps especially to tie-up with heterogeneous issues. Besides, web services add to QoS support in WEB Services, because QoS requirements and its use policies are not still well consolidated in this technology.

A three tyre WEB services is shown in the Figure-6. In this WEB service architecture, each information source or computational element describes itself in a WEB Services Description Language (WSDL) file. As illustrated in Figure 6, a requestor of the service first obtains its WSDL, either directly from the service, as in device discovery, or indirectly from a repository. The WSDL is a complete machine readable description of the service, including how information is represented and what behaviours are provided. It allows the requestor to bind to the provider of the service and establish any necessary translations in a fully automated fashion. Then, data sharing and behaviour invocation are conducted efficiently through the established service interface. XML and WSDL provide an automated framework for defining objects and operations. The application domain determines the specific objects and operations that are included in the services.

VOL. 10, NO 22, DECEMBER, 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

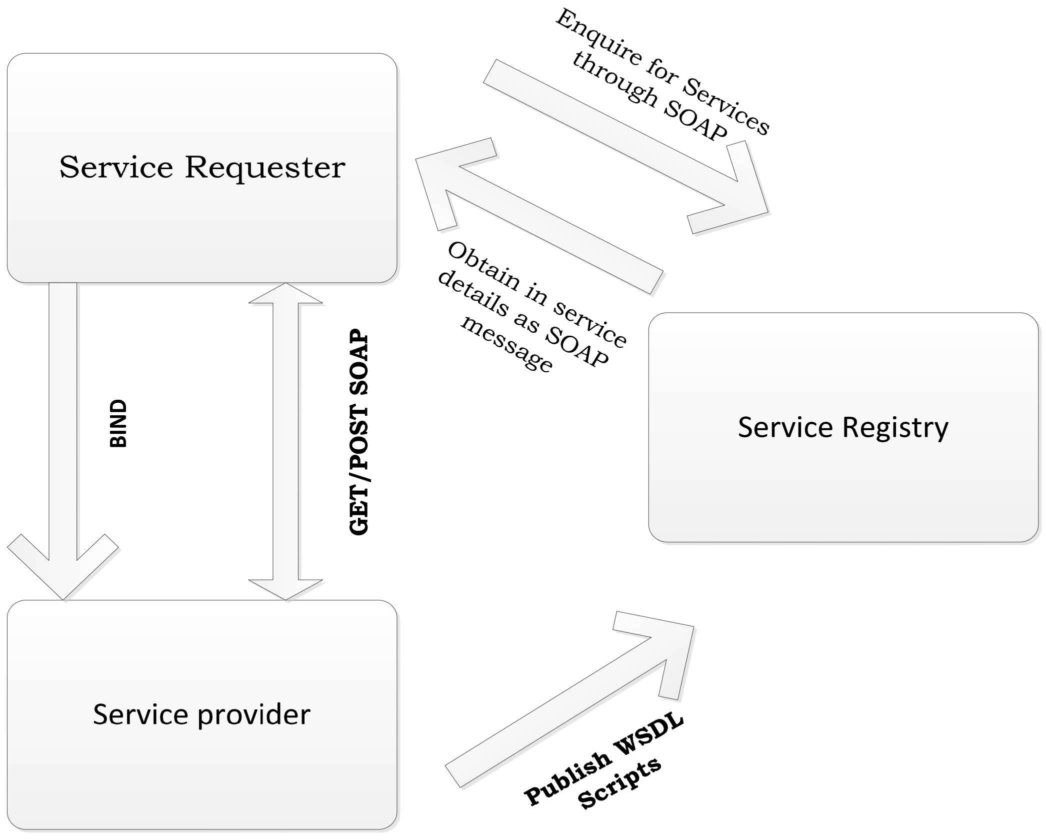

Figure-6. WEB service discovery, bind and utilize.

The increasing intelligence, ubiquity and diversity of sensors have made paramount the need for an interoperable, robust means of communicating with sensors and interpreting their output. In this architecture, three individual entities exist. Service provider develops the web service and hosts the same within its own server. The WSDL specification is developed at the service provider side. Service registry provider maintains the services provided by different service provider through maintaining a registry of services. It implements an enquiry response and a publish interface. A request is made by the service provider through publish interface for registering the service.

The clients bind itself with the service provider using a protocol of its implementation and then starts communicating by posting the messages. The services provided runs the service requested and transmits the messages back to the client. As said earlier the messages are formatted using XML and the messages are encapsulated into SOAP messages which are also developed using XML language.

4.4 A middleware based WEB services architecture

Figure-7 shows a middleware based architecture using which WEB services can be implemented. A middleware kind of architecture would be required when several embedded systems are to be used in the development of an application. Several embedded systems with heterogeneous platforms may have to be used for the development of the embedded applications which is truly a distributed application that either distribute the HW or distribute the ES software. One can implement middleware in each of the embedded system and the embedded system that provides networking services. This kind of architecture is complicated while individual embedded systems runs their respective WEB services, but in the front end, a middleware component works which is an additional burden on the part of the embedded system.

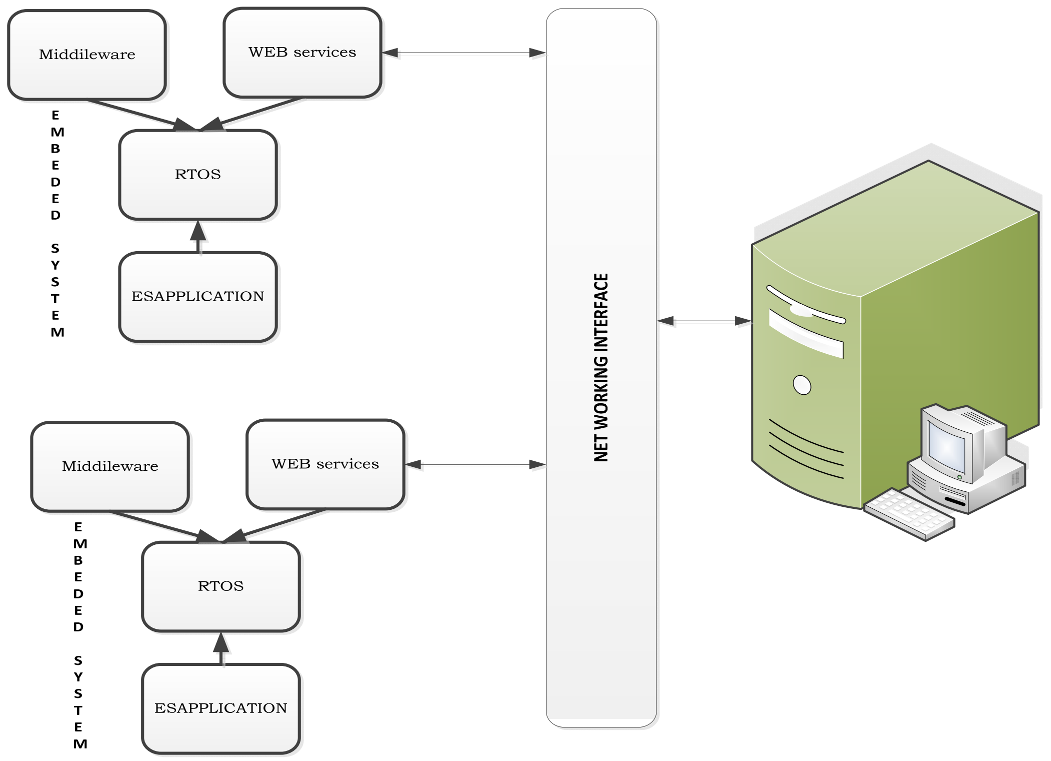

Figure-7. Middleware based WEB services architecture.

This architecture is suitable for use in many domains, independent of embedded devices physical realization or network characteristics. The middleware takes care of heterogeneous issues by marshalling the data either way.

4.5 A four tier WEB services architecture

Web services technology is to be used to provide services for multi-user sharing purposes. Figure-8 shows four tier service architecture. This architecture is useful for the applications that use distributed resource management using the embedded system as the primary means for communication. The Client application can be WEB application rendered through a WEB browser. Client application calls for WEB services for want of using a resource may be for learning or conducting an experiment. The application server hosts all the services that run a part of the application. The WEB service server provides the interface required for communication, messaging through SAP and parsing for getting the actual message and to invoke a service component that has been deployed on the application server. The application conducts the experiment required using the hardware and send back the experimental results back to the client application through a WEB browser or through a specially written application that uses the WEB services. The client applications is used by the user for the purposes of experimenting for using any of the services the way they are required by the client.

4.6 WEB services architecture suitable for dynamic evolution

The most important WEB services architecture which is suitable for implementing dynamic evolution of embedded system is what is required with the dual purpose of implementing dynamic evolution of the communication system through which a HOST communicates with the embedded system in an open manner and also reduce the computation load as much possible. The computation load on the embedded system can be reduced in a way to the HOST. The dynamic evolution system can be made to be resident at the HOST. All the evolution components can be converted into the services and can be made to be working as tasks under the influence of an RTOS. Two of the dynamic evolution system related web service architectures can be implemented.

4.6.1 Dynamic evolution using web services architecture - Alternative -1

This architecture is shown in the Figure-9. This is essentially two tier architecture. Tier-1 is the HOST which is the client itself. At the client, UDDI registry is implemented, thus eliminating the requirement of another server. This also reduces the communication delays quite drastically. The client application uses the UDDI for registering the WEB services through publishing the WEB

VOL. 10, NO 22, DECEMBER, 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.
www.arpnjournals.com

services into the registry. The UDDI is an application by itself that implements the UDDI protocol through use of SOAP/XML messaging. The client application uses UDDI interface both for publishing and enquiring the details of the WEB services. WSDL scripts are written for all the WEB services and the same are published into UDDI

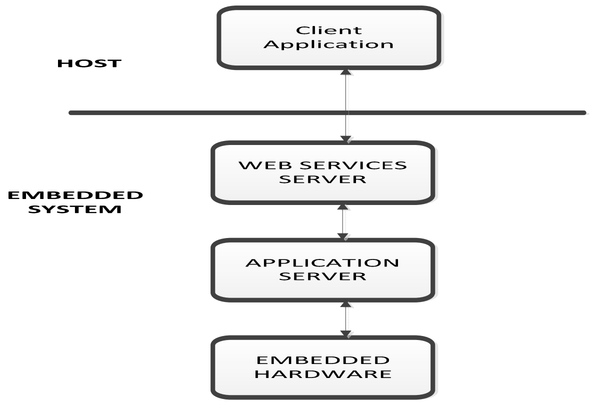

Figure-8. Tire WEB services architecture.

The client application parses the details of the WEB application after obtaining the WSDL scripts from UDDI. The command string that the client needs to transmit is queried from the data base or obtained through a user interface is developed into a XML script and the same is encapsulated into SOAP message and transmitted to the Embedded systems. On the embedded systems side, different types of evolution components are recognized as tasks operating under the influence of a Real time operating system. The tasks are treated as WEB services. However the service execution is undertaken through tasks that together accomplish a command initiated from the HOST as a HTTP request. From the architecture diagram it can be seen that service execution through tasks is always achieved through the syntax evolution task.

This architecture implements all the dynamic evolution models and also archives dynamic evolution of communication system. The communication is undertaken through SOAP messages into which the commands that are coded as XML messages have been encoded. Even the results are also sent as XML messages, thus requiring the parsing to obtain the actual results to be presented using the GUI implemented by the client. The architecture is excellent as long as the embedded system has as much resources as required. This is definitely a light weight web services system as the embedded system is not burdened with the need for a WEB server or Application server which is definitely a major advantage.

The Communication block implements the Communication related web services tasks. The Communication related WEB services task receives the message, parse the message to retrieve the command, validate the command and passes the same to the command processors which validates the command for its semantics and parses the command to its related task which is executed as a service. The service task places the output in global variables and the results are returned by the web service oriented communication services by retrieving the data from the global variables, forming into a XML script, enclose into SOAP message and transmit.

On the Clint side, the Web services oriented communication application receives the SOAP messages, extracts the XML message and parse the same into data which is displayed either on the user interface or the same is used to update the database.

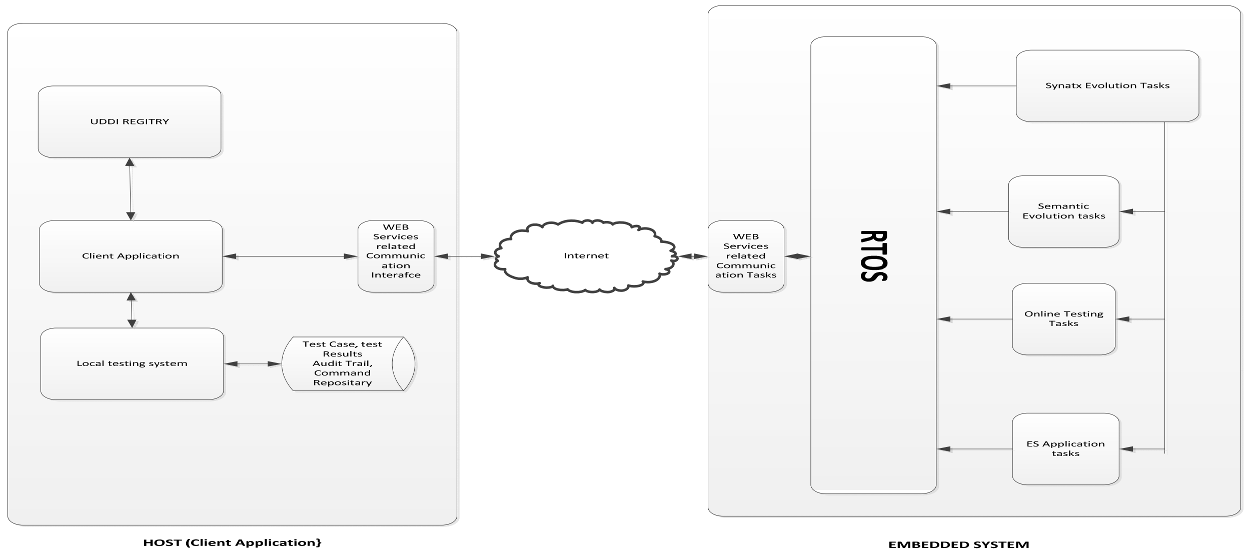
UDDI REGITRY
Client Application
WEB Services related Communic ation Interafce
Local testing system
Test Case, test Results Audit Trail, Command Repositary
Internet
WEB Services related Communic ation Tasks
RTOS
Synatx Evolution Tasks
Semantic Evolution tasks
Online Testing Tasks
ES Application tasks
HOST (Client Application)
EMBEDDED SYSTEM

Figure-9. Dynamic evolution of embedded system oriented - Web Services Architecture - Alternative-1.

4.6.2 Dynamic evolution using web services architecture - Alternative -2

Another WEB services oriented architecture that suits to the dynamic evolution of the embedded system that is implemented on a Low powered low storage embedded system is shown in the Figure-10.

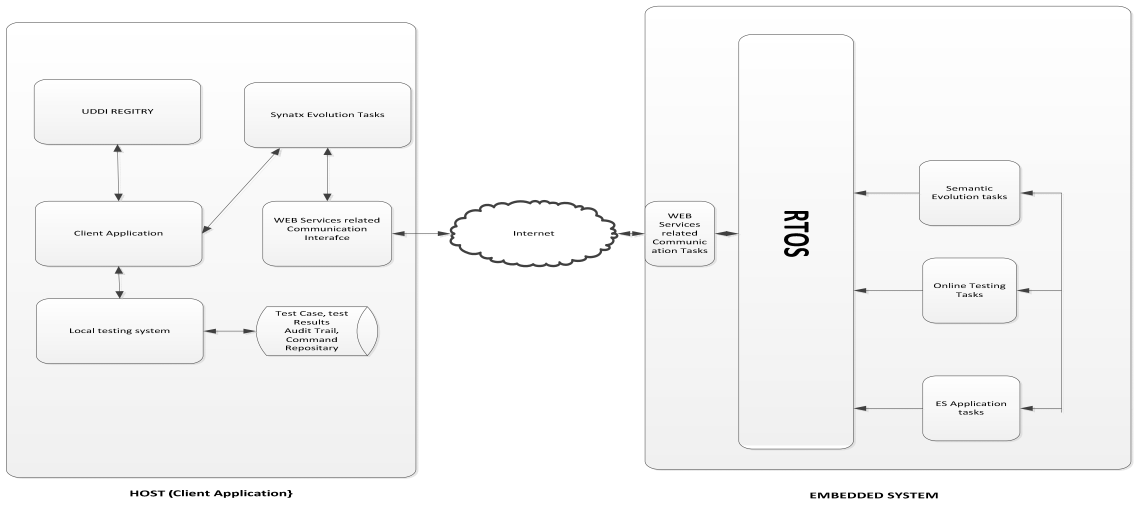
UDDI REGITRY
Synatx Evolution Tasks
Client Application
WEB Services related Communication Interafce
Internet
Local testing system
Test Case, test Results Audit Trail, Command Repositary
HOST (Client Application)
WEB Services related Communic ation Tasks
RTOS
Semantic Evolution tasks
Online Testing Tasks
ES Application tasks
EMBEDDED SYSTEM

Figure-10. Dynamic evolution of embedded system oriented - Web Services Architecture - Alternative-2.

In this architecture the entire block related to syntax evolution of the embedded system is shifted to the HOST. The command language evolution, syntax and sematic verification when the commands are inputted from the user screen are carried on the HOST. If the commands are directly fetched from the database, the syntax and semantic evaluation are bypassed leading to processing of the commands. This architecture implements web services as further light weight services as no web server or application server is used for the implementation of dynamic evolution of the embedded systems. The syntax evolution module now placed in the client takes care of all the tasks/processes that are related to syntax evolution after finding the correctness of the command, hands it over to WEB service oriented communication system. The web service oriented communication system converts the command string into XML specification which in encapsulated into a SOAP message and the same is sent to the embedded server where the SOAP is messaged is received parsed and handed over to the service oriented task that runs under the influence of the RTOS.

5. COMPARATIVE ANALYSIS OF WEB SERVICES ORIENTED ARCHITECTURES

Table-1 shows the comparative analysis of various architectures presented in the previous sections from the point of view of their suitability to implementation of Dynamic evolution of the embedded system. It can be shown that architectures presented as alternative-1 and alternative-2 is best suited for implementing dynamic evolution of the embedded systems.

Table-1 shows the comparison of WEB services based architectural models. Alternative-1 and Alternative-2 architectures are same by all means except for the reasons of load balancing between the client and the embedded system. Alternative-1 is quite suitable if the microcontroller used for implementing the embedded system is high powered backed with huge memory and the alternative-2 is quite suitable for the embedded systems that are built with microcontroller which is medium powered and support medium size of the memory. One of the Alternative-1 or Alternative-2 can be chosen based on the power of the microcontroller that is provided for implementing the embedded system.

6. CONCLUSIONS

Implementation of WEB services for the dynamic evolution of the embedded systems helps in allowing any client using a different communication standard to communicate with the embedded server. The implementation of web servers to establish communication between the HOST and the server truly makes it possible to evolve dynamically with respect to communication. The HOST can be situated anywhere, thus true remoteness of the HOST from the ES server can be achieved which is the most important requirement to monitor and control safety and mission critical systems. Certain amount of load on the ES sever can also be shifted to client when WEB services oriented architecture is chosen for implementation of dynamic evolution of the embedded system.

WEB services oriented dynamic evolution helps one to implement true dynamic evolution which caters for all dynamic evolution components which include Syntax, Semantic, Online Testing and the HOST communication with the client. Task based service implementation does not require any web server or any other application server, thus making the ES based server light, making it possible to implement WEB services within the limited resources available with the embedded system.

Table-1. Comparative analysis of WEB services oriented architectural models for dynamic evolution of embedded systems.

Serial Number	Parameter	ES as server	HOST as server	Three tire (Middleware based)	Four Tier	ES as server (Alternative-1)	ES as server (Alternative-1)
1	Overhead due to WEB server	High	None	High	High	None	None
2	Overhead due to application server	None	High	None	High	None	None
3	Support of dynamic evolution through service chaining	None	None	None	None	YES	YES
4	Storage requirement	Huge	Huge	Huge	Huge	Small	Small
5	Memory requirement	Huge	Huge	Huge	Huge	Less	Less
6	Client and server balancing	None	None	None	None	None	Exists
7	Number of servers required	0	1	2	3	0	0
8	Command based communication	None	None	None	None	Yes	Yes

REFERENCES

[1] Sastry JKR, Venkataram N, Srinivasa Ravi K, Pradeep G, Reddy LSS. 2012. On Dynamic Configurability and Adaptability of Intelligent Tags with Handheld Mobile Devices. International Journal of Electronics, systems and circuits (IIJECS). 1(2): 114-123.

[2] Sastry JKR, Venkataram N, Srinivasa Ravi K, Pradeep G. 2012. Software Architecture for Implementing Dynamic Configurability and Adaptability of Intelligent Tags with handheld Mobile Devices. Research Journal of Computer Systems Engineering - RJCSE. 3(2): 393-398.

[3] Sasi Bhanu J, Sastry JKR, J Viswanath Ganesh. 2015. I^{2}C based Networking for Implementing Heterogeneous Microcontroller based Distributed Embedded Systems. Indian Journal of Science and Technology. 8(15): 1-10.

[4] Sasi Bhanu J, Sastry JKR, Sai Kumar Reddy. 2015. Networking Heterogeneous Microcontroller based Systems through Universal serial bus. International Journal of Electrical and computer Engineering.

[5] Sasi Bhanu J, Sastry JKR, Vijaya Lakshmi Machineni. 2015. Optimizing Communication between heterogeneous distributed Embedded Systems using CAN protocol. ARPN Journal of engineering and applied sciences.

[6] Sasi Bhanu J., Sastry JKR, Suresh A. 2015. Building Heterogeneous Distributed Embedded Systems through RS485 Communication Protocol. ARPN Journal of Engineering and Applied Sciences. 10(16): 6793-6803.

[7] Sasi Bhanu J, Sastry JKR, Mounica. 2015. On Testing Distributed Embedded Systems through Scaffolding. Journal of Embedded systems - Inder Science.

[8] Sastry JKR, Neeraja N, Naga Teja K, Devi Kavya Priya M, Vineela D, Immanuel K. 2012. An Approach towards Development of Communication Standard around CAN Protocol Suite for Networking Embedded Systems. International

Journal of Advances in Science and Technology. 4(2): 36-42.

[9] Sastry JKR, Neeraja N, Pushpa A.G, Satya Prakash G, Tejaswi G, Muni Kumari T. 2012. An Open Specification for Fire wire Standard to extend its suitability to diversified Applications. International Transactions on Electrical, Electronics and Communication Engineering. 2(2): 1-8.

[10] Michael Sig Birkmose, Lars Haugaard Kristensen. 2004. Producing Efficient Web Services for Distributed Embedded Systems. Thesis - AALBORG UNIVERSITY.

[11] Stanislav Sliva, Vilem Srovnal. 2005. Distributed Processing Applicable To Embedded Systems. WSEAS International Conference on Automatic Control, Modelling and Simulation.

[12] Kevin J. Ma, Radim Barto. 2005. Performance Impact of WEB Service Migration in Embedded Environment. IEEE.

[13] Nikolay Kakanakov, Grisha Spasov. 2005. Adaptation Of WEB Services Architecture In Distributed Embedded Systems. International Conference on Computer Systems and Technologies.

[14] R. Pallavi, C. Veeranna. 2015. Implementation of TCP/IP Ethernet WEB services On ARM7 LPC2148 for Embedded Systems. IJARCSSE. Volume 5.

[15] Kreger H, Austin, D., A. Barbir, C. Ferris, S. Garg. Austin, D., A. Barbir, C. Ferris, S. Garg. 2001. Web Services Conceptual Architecture (WSCA 1.0). IBM Software Group, www.redbooks.ibm.com.

[16] Daniel Schall, Marco Aiello, Schahram Dustdar. 2005. WEB Services on Embedded Devices. J. Web Infor. Syst. Vol. 1.

[17] Guilherme Bertoni Machado, Frank Siqueira, Roninson Mittmann, Carlos Augusto Vieira. 2006. Integration of Embedded Devices through Web Services: Requirements, Challenges and Early Results.

[18] Mitko Petro Shopov, Hristo Matev matev, Grisha Valentino Spaso. 2007. Evaluation of WEB services Implementation for ARM Based Embedded Systems. Proc of Electronics. Vol. 7.

[19] David E. Culler, Gilman Tolle. 2007. Embedded WEB services: Making Sense out Of Diverse Sensors. Arch Rock Corp.

[20] Danilo J. Oklobdzija, Bransislav T. Jevtovic. 2008. Using XML WEB services as Platform for Remote Access and Control of Embedded Systems. Vol. 28.

[21] Elmar Zeeb, Andre Pohl, Ingo Luck. 2008. WS4D: SOA-Toolkits Making Embedded Systems Ready for WEB services.

[22] Risto Serg, Johanner Helander. 2008. Using XML WEB services For Embedded Systems Interoperability World's Smallest WEB 2.0 Server Demo.

[23] Yin-Wei-Feng, Sun Rong-Gao, Wan Zhong. 2009. Distributed Remote Laboratory Using WEB services For Embedded Systems. CCISST.

[24] Elmar Zeeb, Guido Moritz, Dirk Timmermann. 2010. WS4D: Toolkits For Networked Embedded Systems Based on the Devices Profile for WEB services.