DISTRIBUTED PROCESSING APPLICABLE TO EMBEDDED SYSTEMS

STANISLAV SLIVA, VILEM SROVNAL
Department of Measurement and Control
Faculty of Electric Engineering and Computer Science
VSB-Technical University of Ostrava
17. listopadu 15, 708 33, Ostrava-Poruba
CZECH REPUBLIC

Abstract: - This paper describes several possibilities of using distributed computation in an embedded environment. Distributed processing plays an important role in applications whose parts (procedures) are executed outside the local node in remote nodes distributed in a network. A major part of the paper is focused on a description of web services and related protocols (SOAP, XML-RPC). QNX Transparent Distributed Processing is mentioned as an alternative to standard distributed processing based on RPC (Remote Procedure Call).

Key-Words: - embedded system, remote procedure call, Qnet, SOAP, XML, network, TCP/IP.

1 Introduction

One of the motivations of using distributed processing mechanisms in designing embedded system is to extend the possibilities of such a system to have an access to the resources and services of other systems in a distributed environment. The aim of future work consists in the right choice of distributed system, which will be able to implement into resource limited embedded systems and which will provide an environment for remote procedure execution regardless of the platform used of a remote service provider. An example of such functionality should be the execution of a resource intensive algorithm on some data which are generated by an embedded system. This algorithm may be too difficult to its local implementation in an embedded device or its implementation needs too much memory. Such an algorithm would be implemented in other more powerful equipment like a network server in the form of service. This service would be called by an embedded system if necessary.

The next section addresses the description of several existing distributed processing mechanisms.

- RPC (Remote Procedure Call)
- Qnet (QNX Networking)
- Web services

2 Remote Procedure Call

RPC is an easy and popular paradigm for implementing a communication model client – a server of distributed environment. Briefly described, the RPC mechanism is initiated by calling – client, which sends the request message to a remote node (server) to execute a certain procedure using sent arguments. The resultant message is returned to the caller. Lots of various implementations of RPC exists, some of them use different protocols and due to it they could be incompatible with each other. In order to allow servers to be accessed by different clients, a number of standardized RPC systems have been created. Most of these use an IDL (Interface Description Language) to allow various platforms to call the RPC. Sun RPC (ONC RPC), Distributed Computing Environment (DCE), Microsoft DCOM, CORBA are examples of different implementations of a RPC mechanism. The actual trends in distributed processing are web services, which use XML (eXtended Markup Language) instead of IDL and which use Internet protocols such as HTTP or SMTP as transport protocols. An example of such a RPC mechanism is SOAP (Simple Object Access Protocol) or its predecessor XML.

3 Qnet – QNX Networking

Qnet is native network distributed processing mechanism used in the real-time operating system QNX. This scalable operating system is aimed not only at powerful workstations and servers but also at embedded systems based on various microprocessors. This distributed processing mechanism, also called Transparent Distributed Processing, is realized using passing message. The message passing mechanism is a fundamental part of Inter Process Communication (IPC) of QNX system. The core of this system consists of a microkernel surrounded by modules (managers) and they

communicate with each other mainly by using message passing. Qnet extends the capabilities of message passing to a local area network (LAN). Qnet also provides transparent access to all Qnet nodes resources in a local network. Qnet messages could be encapsulated into network protocol IP (TCP/IP) in the case of distributed processing beyond the local area network.

To understand how a network-wide message passing works, consider two processes who want to communicate with each other: a client process on one node named *lab1* and a server process (serial port manager /dev/ser1) on another node named *lab2*. A client simply calls *open()* function [2].

```
fd =open("/net/lab2/dev/ser1",O_RDWR....);
/*Open a serial device on node lab2*/
```

The Figure 1. shows message passing communications necessary for the successful opening of a remote serial port.

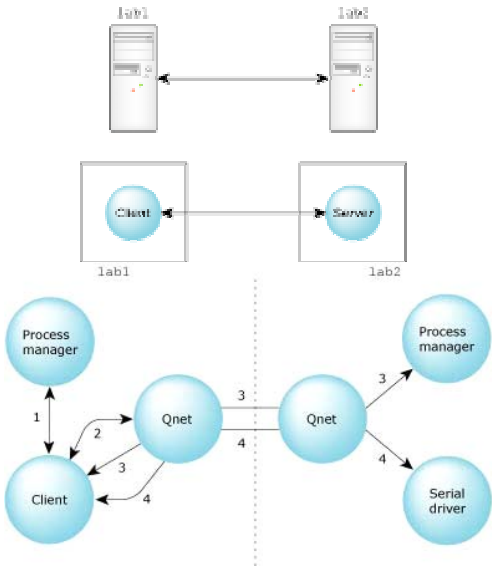

Fig.1 Message passing between client (application) and server (serial driver)

Here are the interactions:

1. A message is sent from the client to its local process manager, effectively asking who should be contacted to resolve the pathname */net/lab2/dev/ser1*.
   The process manager returns a redirect message, saying that the client should contact the local network manager.
2. The client then sends a message to the local network manager, again asking who should be contacted to resolve the pathname. The local network manager then replies with another redirect message, giving the node descriptor, process ID, and channel ID of the process manager on node `lab2` -- effectively deferring the resolution of the request to node `lab2`.
3. The client then creates a connection to the process manager on node `lab2`, once again asking who should be contacted to resolve the pathname. The process manager on node `lab2` returns another redirect - node descriptor, channel ID, and process ID of the serial driver on its own node.
4. The client creates direct connection to the serial driver on node `lab2`, and returns a connection ID that can then be used for subsequent message-passing operations. After this point, from the client's perspective, message passing to the connection ID is identical to the local case.

The main advantage of this type of distributed processing is high efficiency. It's because a message passing mechanism is supported by the microkernel and the message transfers between a client and server are binary.

The disadvantage of Qnet is its tight couple to the QNX operating system so its implementation to other platforms is very difficult. Qnet isn't an open standard and that's also the reason why it's used only in a QNX operating system.

4 Web services

Web services (WS) were created because of the need to have standard way of accessing and using a distributed mechanism regardless of the platform used. A web service is any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any one operating system or programming language [1]. The application in model of web services consists of small building blocks – functions regardless of their locations in Internet servers or their particular implementations. Such an application is easy to create, change and dynamically modify. The web services as a model of distributed processing and is very popular due to looser couples in comparison to traditional models such as DCOM, CORBA, etc. Web services are based on three standards (Fig.2):

- Data exchange is realized by SOAP protocol or XML-RPC both based on a XML standard
- WS provide uniform methods of service interface description. This description is usually provided in the form of an XML document using WSDL (Web Services Description Language). A user must be able to create a client application by using a WSDL description.

- WS are registered so the other users can find them easily using UDDI (Universal Discovery Description and Integration)

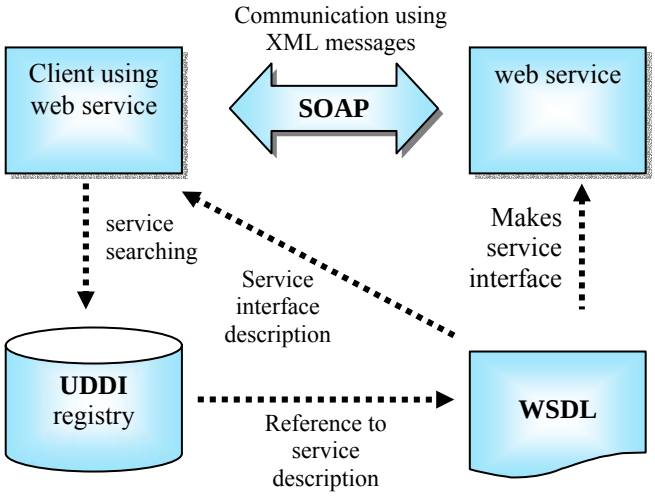

Fig.2. Relations between WS standards

Fig.3. reflects four steps which should be done in the case of service requestor perspective and Fig.4. shows five steps which should be done in case of service provider perspective.

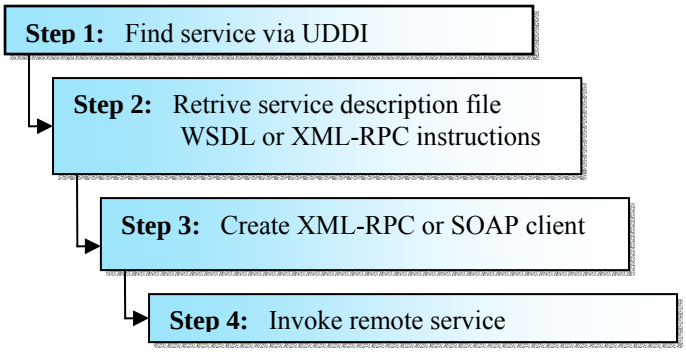

Fig.3. Service requestor steps

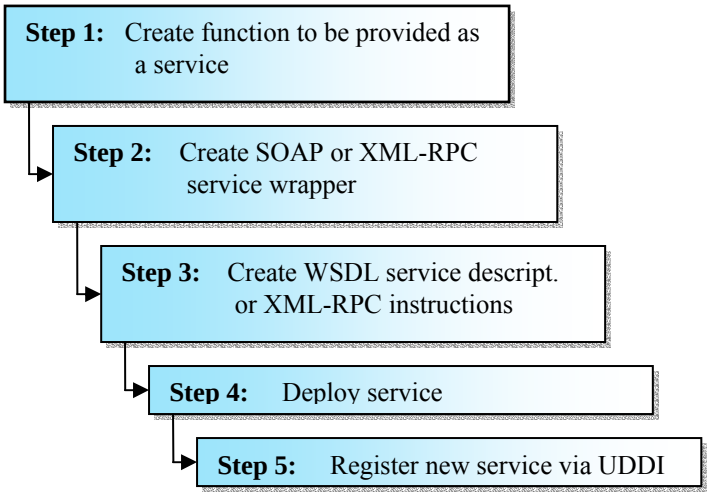

Fig.4. Service provider steps

To summarize, a complete web service is any service that [1]:

- Is available over the Internet or private (intranet) network
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

4.1 XML-RPC

XML-RPC is a very simple concept with a limited set of capabilities used to remote procedure invocation. Those limitations are in many ways the most attractive feature of XML-RPC, as they substantially reduce the difficulty of implementing the protocol and testing its interoperability [1]. Procedure call request is encoded into XML form and transferred via HTTP POST. After execution of a remote procedure, the results are returned via HTTP reply also in XML form. In cases where a wide variety of different systems need to communicate, XML-RPC may be the most appropriate lowest common denominator. XML-RPC is in many ways easier than SOAP so it's easier to implement it, but on the other hand XML-RPC does support neither service description WSDL nor service searching using UDDI. The data model of XML-RPC consists of six basic data types and two compound data types that represent combinations of types. All data types are represented by simple XMP elements. Here is an example of a string and integer element:

```
<string>Hello !</string>
<int>31</int>
```

4.2 SOAP

SOAP is a base protocol of web services. SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML [1]. Other types of frameworks – CORBA, DCOM or Java RMI provide similar functionality like SOAP, but the difference is that SOAP messages are represented in XML form so they are platform independent. SOAP doesn't define a program model or implementation. Instead of this, it defines modular packaging model end encoding mechanisms to code data in modules. Other standards (WSDL, UDDI....) were created later after the release of SOAP and extend SOAP possibilities. SOAP specification consists of three main parts:

- SOAP envelope specification – rules for encapsulation of data transferred between computers. Called method names, their

parameters or return values are specified there. Envelope specification also includes information about who should process the content of envelope and how to encode error messages.

- Data encoding rules – set of conventions (based on XML) for encoding different data types
- RPC convention - SOAP can be used in a variety of messaging systems, including one-way and two-way messaging (Fig.5). For two-way messaging, SOAP defines a simple convention for representing remote procedure calls and responses.

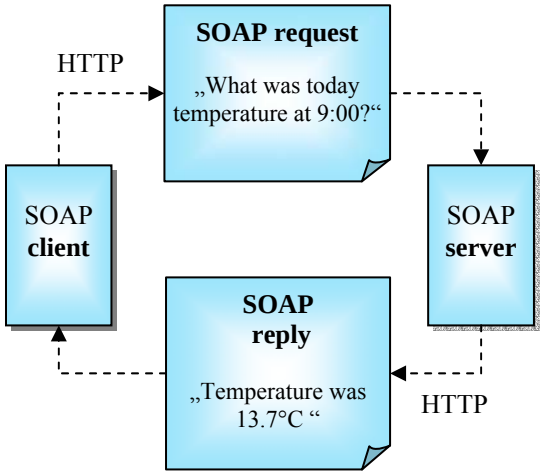

Fig.5. Example of SOAP conversation

One-way message, client request or server response, is referred to a SOAP message. Every SOAP message has mandatory and optional elements (Fig.6).

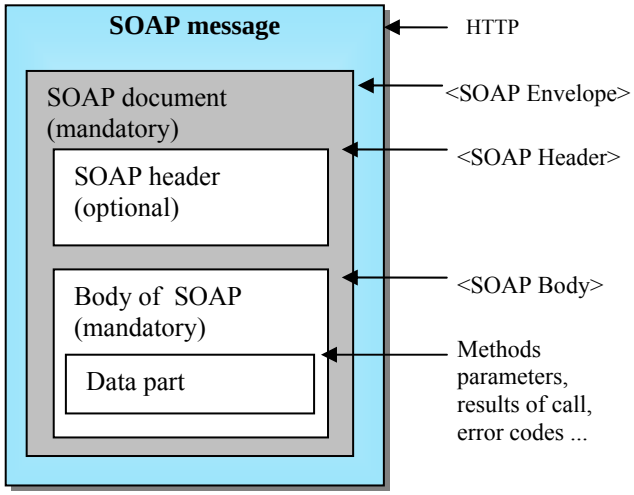

Fig.6. SOAP message main elements

4.3 Web services and embedded devices

Embedded systems in contrast of workstations dispose of limited resources. They usually have less memory and processor performance. Implementation of web services means additional load from the perspective of resources.

The necessary component of every embedded device supporting web services must be TCP/IP stack. The stack provides standard transport mechanisms (TCP, HTTP, SMTP...) for web services. An efficient TCP/IP stack is essential for building web services infrastructure especially on small 8 and 16 bit systems. SOAP protocol isn't fixed only to HTTP protocol, SOAP can be used in conjunction with other internet protocols. Direct insertion of a SOAP message into a TCP segment can save some necessary computing and memory.

XML is a text-oriented language and its advantage is platform independence. On the other hand, XML data transfer is less effective than a binary transfer. Therefore, it's necessary to implement larger incoming and outgoing buffers. The important part of the system supported by WS is XML parser/generator and the SOAP message encapsulation/extraction service. Well-pared XML parser that fully supports SOAP can be under 20KB in size [4]. This could be still too much for integration to 8 or 16 bit systems. In this case, it's possible to use web services without SOAP protocol. You can use a simpler XML-RPC mechanism, which needs less memory and it's easier to implement.

5 Conclusion

The paper describes some essentials about several distributed processing mechanisms. The possibility of using web services in branch of an embedded system was discussed here. Our future plan will be the implementation of some mentioned features of web services into an embedded system represented a by microcontroller. First prototype based on Rabbit microcontroller will be able to understand XML-RPC conversations.

Acknowledgement:
The work and the contribution were supported by the project GAČR102/05/0571 – Architectures of embedded system networks.

References:

[1] Cerami.E., Web Services Essentials,O'Reilly, ISBN 0-596-00224-6, 2002.

[2] QNX Developer support centre, Native networing (QNET), www.qnx.com.

[3] Short S., Building XML Web Services for the Microsoft .NET Platform, Microsoft Press, ISBN 0735614067, 2002

[4] Canosa J., Introduction to Web Services, www.embedded.com, 2002.