DISTRIBUTED WIRELESS COMPUTING WITH MULTIPE DOMAINS

Sabares Moola , Carlos Aguayo Gonzalez, Carl Dietrich, Jeffrey Reed (Wireless@VT, Virginia Tech, Blacksburg, VA, USA; sabares, caguayog, cdietric, reedjh@vt.edu).

ABSTRACT

The motivation for distributed computing across multiple software defined radios (SDRs) evolved from Component and Service based Radio Architectures like the Software Communication Architecture (SCA). This architecture provides an opportunity for value-added services and collaborative communication through its inherent ability to provide distributed data processing across multiple SDRs. We demonstrate a simple proof-of-concept for constructing a software framework for opportunistic signal processing and distributed data processing across SDR nodes. We build this framework on top of the Open source SCA implementation::Embedded (OSSIE), developed by Virginia Tech, for demonstration purposes. A preliminary demonstration implements opportunistic processing of data collected from SCA-based SDR nodes that are connected through a wired or fixed wireless backbone network, laying groundwork for advanced applications like parallel data and signal processing on both wireless infrastructure and Mobile Ad-hoc Networks (MANETs).

1. INTRODUCTION

In recent years, Distributed Computing has gained prominence in the wireless research community [1]. Recent advances in radio technologies have given leverage for this research, towards the goal of achieving optimized usage of power and bandwidth. In addition to this, they provide advantages like sharing of resources, fault tolerance, and dynamic adaptability to the environment.

The emergence of software defined radio (SDR) and cognitive radio has necessitated the convergence of Wireless Distributed Computing (WDC) towards these technologies. The requirement of dynamic reconfiguration of wireless environment, transparent allocation of resources, reliable exchange of data among the mobile nodes, etc, has been satisfied by these radio technologies.

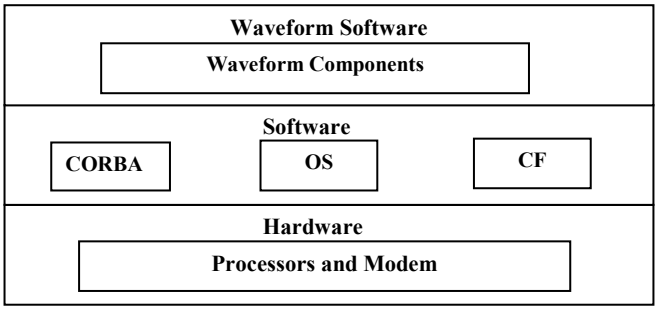
The SDR has its functionality developed in software rather than hardware. This approach allows for rapid deployment, ease of upgrades, etc. In order to make this radio terminal technology widely accepted and compatible with legacy radios, an open architecture called the Software Communication Architecture (SCA) was published by Joint Tactical Radio System (JTRS) Joint Program Office (JPO)[2]. Its goals include enhanced interoperability among different SCA-compliant radios, reduction in development and operation cost, ease of development and deployment of waveforms, and portability among different SCA implementations. The SCA specification relies on Object Oriented Programming Concepts, Common Object Request Broker Architecture (CORBA) middleware and CORBA Component Model (CCM).

The SCA in its specification of Operating Environment for SDRs defines interfaces for different software components. Such interfaces provide common infrastructure for distributed application deployment. We took advantage of these interfaces to build a network of nodes across a wired backbone, to access the services offered by the individual nodes, and to control the deployment of waveforms across the nodes. This involved obtaining the references of individual remote nodes over the CORBA transport protocol. Since the SCA specifies object-oriented development of components, obtaining the object references of SCA components will enable an application to interact with these components to form a larger distributed application irrespective of its location.

To illustrate this concept, we created a simple testbed using multiple nodes running the OSSIE framework. We used a C++ implementation and some shell scripting in order to automatically scan the network and identify nodes available to share resources for distributed computing.

The paper is organized as follows: Section 2 gives a brief description about how to build a distributed architecture based on the SCA architecture specification. The section also explains the need for following a multi-domain approach to achieve WDC. Section 3 discusses the proposed implementation details using the CORBA middleware specified by the SCA architecture. Finally, Section 4 gives the results and presents possibilities for future work.

Figure1. SCA Architecture Abstraction Levels



2. DISTRIBUTED ARCHITECTURE

In this section we describe concisely the various features involved in creating the distributed architecture.

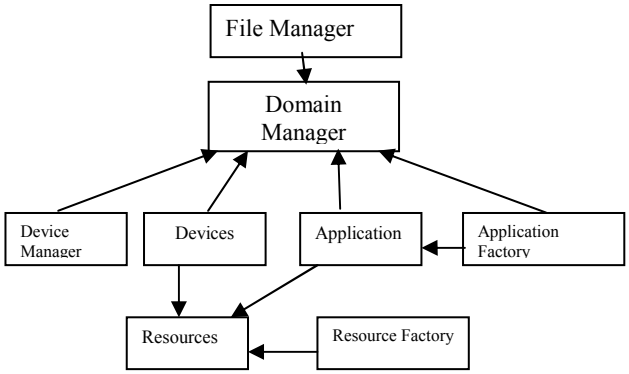
2.1. SCA Architecture

The SCA architecture is an implementation-independent software framework for creating JTRS compliant SDR nodes. The SCA tries to decouple the hardware from the software. Since the SCA deals mostly with specification of Software Architecture, we focused on a software-based approach for realizing distributed computing over wireless networks. The effective re-configuration of components along with real time exchange of information across the multiple SDR nodes, distributed throughout the environment, is made possible by this approach.

The SCA addresses the issue of modularity, scalability and compatibility with different radios including legacy radio. The architecture provides the flexibility of including all the modules in the software, which, replacing dedicated hardware for a wide variety of radio applications and functionality. The issue of inter-operability between different SDR nodes residing with multiple parties (e.g., Homeland Security, Navy, Air Force, etc.) is also addressed by the SCA. The Figure 1 shows the break down of the SCA. Among these features CORBA, specified as middleware by the SCA, enables distributed deployment of radio components. We further extended the concepts learned from the SCA to create a framework for realizing distributed applications across different individual SCA-based radios. We describe in this paper the work done on top of the existing framework to realize this.

The SCA defines an Operating Environment (OE) that aims to separate the applications and platforms. The Operating Environment is responsible for defining interfaces and providing a mechanism for deploying and controlling

Figure2. Simplified Relationship between Domain Manager and other SCA services



the applications and its components across hardware platforms. The SCA defines object oriented (OO) model development for defining various features associated with the OE. This model modularizes both software and hardware base components associated with the OE. The associated interfaces required between the modules are also defined in the OE. We took advantage of these OE features to create an environment for building distributed applications.

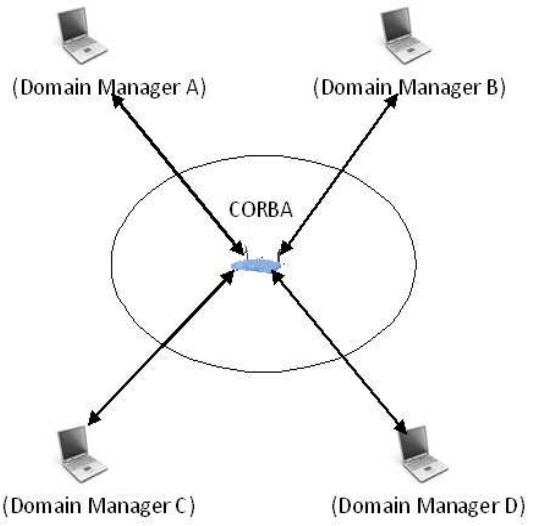
2.2. Core Framework and Domain Manager

The Core Framework (CF) provides a set of interfaces that are responsible for inter-connection of software components in distributed communication systems. These interfaces are responsible for deployment and running of applications. Through these interfaces, the CF provides complete abstraction for underlying Hardware and software layers. The interfaces are described for Applications, Control, Framework Services and Domain Profile. We use these interface to create a federated structure using Domain Profiles from different SCA-based SDRs to create a system for distributed waveform applications. Though the specification by itself enables the distributed computing, we demonstrate through this paper , a simple proof-of-concept for realizing advanced signal processing applications that will reach down to Physical Layer. The applications include distributed MIMO, cooperative spectrum sensing, cooperative relaying, etc.

SCA component, both hardware and software, have to register with the Domain Manager in order to form single entity as a radio. The Domain Manager is responsible for controlling and managing all the devices, resources, and applications under a single radio. Since it is necessary to

register all the resources available under a radio to a Domain Manager, it can act as a gateway by which a remote client/radio can access all these resources. The instantiation of DomainManager creates a naming context in the CORBA Naming Service. In the subsequent sections, we describe the use of CORBA and the necessity of the CORBA Naming Service in the Distributed Computing.

Figure3. Multi Domain Approach Based on SCA Framework



2.3 Multiple Domain Managers

A Domain Manager acts a repository for all the services offered by a single radio. We mention here services such as DeviceManagers, FileManagers, Resources, Applications, and Application Factories that are registered to the DomainManager during their instantiation. Therefore, obtaining a reference to the DomainManager is like getting a reference to each of the resources and services offered by the radio.

By doing so, any individual radio formed by the collection of components and resources under a single DomainManager will be able to interact with other DomainManager-specific SDR radios. With this interaction among different Domain Managers, each radio will have the capability to look into the services and resources available from radios in its vicinity. Thereafter, the radios can share data among multiple nodes using a reliable transport mechanism. Because of the component based model of the SCA, we could set up a communication system model using individual components from different nodes. With such a system model we can configure and control the properties of components of remote SDR nodes. In the next sub-section we describe about the mod of realizing the structure for multiple Domain Managers.

2.4 Federated Structure

Our objective is to create a federated structure from the services offered by DomainManagers to realize the goal of WDC. This requires obtaining the list of available SDR nodes. The mechanism of identifying nodes can be referred to as service discovery. We rely on CORBA to obtain the Naming Reference for DomainManager objects from different radios/network nodes. The details about the implementation are explained in the subsequent sections.

The above mentioned method of achieving our goal of wireless distributed computing became feasible because of the extension of the OSSIE open source SCA implementation [5]. In the next section, we describe about the implementation details of the architecture that was outlined above.

3. IMPLEMENTATION

In this section, we describe in brief the CORBA middleware and how to utilize the services offered by it to implement distributed computing.

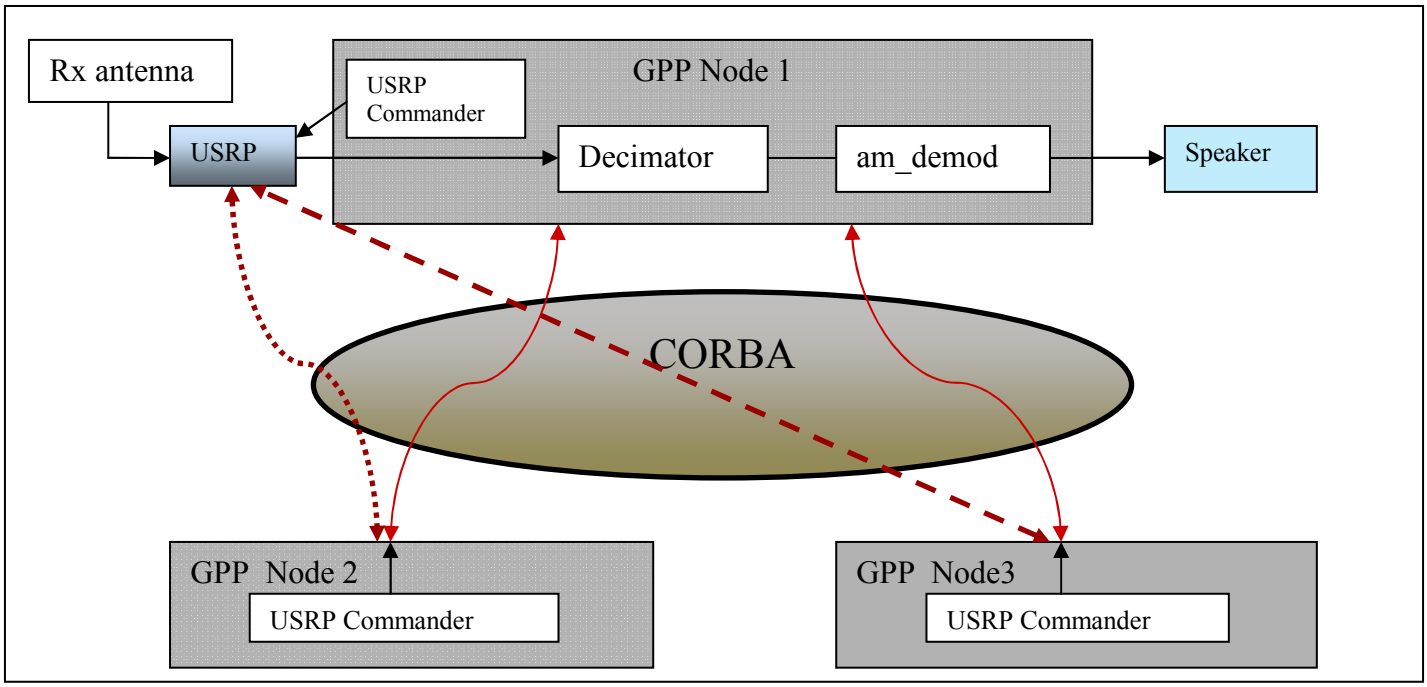
3.1 CORBA Middleware and Distributed Systems

CORBA middleware is targeted for heterogeneous networks with varied platforms and operating systems. It is well suited for distributed computing over heterogeneous networks. Therefore, SCA architecture specified it as communication language for the varied components distributed within the SDR radio. It is also targeted for wider portability among different hardware platforms. We created a simple proof of concept for demonstration of WDC using SCA-based radios. CORBA is used in the SCA Core Framework and it acts as interface for passing messages between components. We assume that there exists a reliable transport mechanism across the nodes. In the following subsections, we describe the procedure followed for implementation of the distributed framework.

3.2 Obtaining Object References

Our first task is to obtain Initial References for the Naming Service from SDR nodes. These references are essential for establishing communication between different nodes. Generally, object references are stored in the standard format called Interoperable Object References (IOR). These references are transported to other nodes explicitly by e-mail or by storing it some other file. The initial references acts a kind of bootstrapping mechanism for establishing contact with the other nodes.

Figure4. Simple Distributed Application based on OSSIE Framework



For our proof of concept, we considered a real-time scenario, by creating a local network with three to four nodes running the OSSIE framework, in which there is no information available about a node apart from its Internet Protocol (IP) address (IPv4). We used the IP addresses to obtain the initial Naming reference from different nodes. We devised a Linux-based approach to scan the entire network in order to obtain the IP address of the SDR nodes connected to the local network. From the list of IP addresses, we were able to obtain the initial references.

Before a node can interact with another node, it has to initialize an Object Request Broker (ORB) between those two nodes. This has to be done for multiple nodes in order for their resources to be used for computing purposes. In general the initial reference to the Naming Service either by bootstrapping mechanism or by explicitly specifying the mode of obtaining reference like underlying transport mechanism, address of the location. In the open source omniORB CORBA implementation used in our work [8], when an ORB is initialized between any two nodes, by default it takes the reference for bootstrapping from a configuration file, e.g., omniORB4.cfg. Instead, we passed the initial reference as argument to the code that initialized the ORB.

In order to ensure interoperability between different ORB products, we used a bootstrapping mechanism based OMG defined URL called corbaname. This URL takes the following form:

-ORBInitRef NameService=corbaname::<IP Address>.

The above URL forms a command line argument for ORB initialization. Thus, we perform ORB initialization as soon as we have the list of IP address. Once we obtain the initial reference to Naming service, we use the ORB::resolve_initial_references() function to further narrow down to other references. For every SDR node in our network, we have a DomainManager that makes a unique context under every initial Context. Therefore, once we obtain the reference to the initial Naming Context we can obtain reference for a radio's DomainManager, which then acts as a gateway for all the components and resources available under that radio.

3.3 System Setup

We used two nodes with Intel Centrino Processor and two nodes with an Intel Atom Processor. All these nodes ran the Ubuntu 8.04 OS and had OSSIE v0.7.3 installed. These nodes are connected to form a local network with a wired backbone.

3.4 Distributed Application Model

In this section, we describe a simple distributed application that can built from the re-configurable components from the OSSIE [5] framework. The system description is shown in figure4.This application involves a simple SDR radio that acts a AM radio receiver whose frequencies can be controlled by other SDR nodes. The application uses OSSIE Components like USRP_Commander, Decimator, am_demod. The audio can be played through Sound card or played through ALF speaker.

The Node1 acts a simple AM receiver. Its initial Naming reference is obtained by other two nodes Node2 and Node3 that can control the AM receiving frequency of the Node1. The same system can be implemented by making the components like Decimator and am_demod operating in other nodes Node2 and Node 3 respectively. The audio can then be heard in Node1. In this system, every transport of data is handled through CORBA transport layer.

4. RESULTS AND FUTURE WORK

In this paper, we demonstrated a simple proof of concept for distributed computing using the OSSIE framework. Through this demonstration, we are able to prove that enhancement to already available service based radio architectures like the SCA can enable opportunistic signal processing. This processing can take for advantages of both infrastructure and mobile ad-hoc networks, which provide the more challenging opportunity for wireless distributed computing (WDC). These distributed processing capabilities can be used for applications like parallelization of data and signal processing, data reduction and data fusion. Advanced applications include position location, distributed MIMO and automated frequency planning in ad-hoc networks.

For the future work, we aim to develop self-configuring software that will be used on top of existing open source SCA implementation called OSSIE. This will become target for advanced distributed applications that will reach down to physical layer in both static and dynamic wireless environment.

5. REFERENCES

[1] S.Zhou, M. Zhao, X.Xu, J.Wang, Y.Yao, "Distributed Wireless Communication System: a new architecture for future public access", IEEE Communication Magazine, Vol 41, Issue 3, March 2003.

[2] Software Communication Architecture Specification, Version 2.2.2.

[3] J.Bard, Vincent J.Kovarik Jr., "Software Defined Radio – The Software Communication Architecture', John Wiley & Sons Ltd, England, 2007.

[4] C.A. Gonzalez, C. Dietrich, J.H. Reed, "Understanding Software Communication Architecture", IEEE Communication Magazine, To be published.

[5] OSSIE Website, http://www.ossie.wireless.vt.edu.

[6] M. Henning, S.Vinoski, "Advanced CORBA Programming with C++", Addison-Wesley Professional Computing Series, USA, 1999.

[7] D. Grisby, S. Lo, D. Riddoch, "The omniORB version 4.1 User's Guide".

[8] omniORB 4.1, http://omniorb.sourceforge.net.