A Resource Design Methodology and Adapter's Implementation for Core Framework for Software Defined Radio

Jin-Ho Son[1] · Tai M. Chung[2]

Abstract

Software Communication Architecture(SCA) has been made to reduce the development period of new waveform applications by reusing design modules. Software Defined Radio(SDR) forum adopted SCA specification and follows its design concept. However, SCA specification does not describe interfaces and deployments of resources in details, which are software components, and how to implement them as well as how to communicate each other are another missing points in specification. In this paper, we propose a resource design methodology and details of adapter implementation for Core Framework(CF) that is a core set of open interface and services providing an abstraction of the underlying software and hardware layer. We also present SDR prototype system which communicates with resources implemented in different programming languages and processors in legacy environment through our design methodology.

Key words : SDR, SCA, Software Component, Core Framework.

Ⅰ. Introduction

The Software Defined Radio(SDR) is composed of a wide range of software modules including radio frequency(RF) part which is processed in high speed Digital Signal Processor(DSP) and Field-Programmable Gate array(FPGA) chips. SDR technologies support over-the-air upload of software modules(i.e. multi mode and multi band or service functions, etc.) without hardware replacement, in which radio functions are included such as networking infrastructure equipments and subscriber terminals as software modules running on a generic hardware platform[1].

Software Communication Architecture(SCA) specification has been published by the Joint Tactical Radio System(JTRS) Joint Program Office(JPO). SCA has been structured to reduce the development period of new waveform applications through the ability to reuse design modules and to build on evolving commercial frameworks and architectures.

In this paper, we propose resource design methodology and details of adapter implementation for Core Framework(CF). In addition, we present SDR prototype system in communication with resources implemented in different programming languages and processor in legacy environment through our design methodology.

The rest of this paper is organized as follows. In Section II, we provide an overview of SCA Software. Section III describes resource design methodology and details of the Adapter's implementation for CF. In Section IV, we investigate our SDR prototype. The paper concludes in Section V with suggestions for future work.

Ⅱ. SCA Software Overview

SDR system is an embedded waveform application system which is not built-in software of radio applications but reconfigurable software for different services according to user request without replacement of service platform in real time. Therefore, SDR needs an object-oriented structure based middleware technology[2],[3] and serves variable system modules and information between module/control interfaces and Application Program Interface(API)[4]. Application software is downloaded by ATM, Smart card or over-the-air.

SCA using Object-Orient Technology for the JTRS[5] and SDR forum[8] is defined by industry participants. SCA includes two hierarchical structures. One is Application layer; the other is Operating Environment (OE). OE is divided into CF and Commercial-Off-

Manuscript received June 18, 2003 ; revised September 3, 2003.

[1] DNT Group Information Technology Lab., LG Electronics Institute of Technology, 16 Woomyeon-Dong, Seocho-Gu, Seoul 137-724, Korea.

[2] School of Information & Communication, Sungkyunkwan University, 27328 Engineering Building II, Suwon, Korea.

The-Shelf(COTS). The SCA structure of software is illustrated in Fig.1. However, SCA specification does not describe interface and deployment of Resources in details as well as implementation issues and communications process. The Software structure of SCA is illustrated in Fig. 1.

Ⅲ. Resource Design Methodology and Adapter Implementation

In Section 3-1, the design concept of the SCA is reviewed and, based on this, we propose the design methodology of Resource and Resource adapter in Section 3-2. In Section 3-3, we describe how the duplexing of communications process is implemented using the shared memory mechanism for Resource.

3-1 Resource Concept in SCA

The design methodology of resources, which are software components, is a significant factor in SDR system design, since the system operation is mainly based on dynamic link between software resources and hardware resources using the equivalent downloaded resources of systems.

Resources can inherit and encapsulate common API types of Base Application Interfaces(Message Registration, Message, Lifecycle) for control, configuration and message path of a software component. Our SDR prototype extends these definitions by creating Protocol resource and Modem resource. The design of resource's internal functionality is not dictated by SCA and it is left to the application developer. The model of a resource for our SDR prototype is depicted in Fig. 2.

3-2 Main Characteristic of Proposed Design Methodology

It is not easy to set communications process with resources developed in different environments. So, we need a resource adapter that helps to transfer messages among resources. A resource adapter makes it possible to transfer data/messages to or from The Common Object Request Broker Architecture(CORBA)[9] resources and non-CORBA resources. Since a resource adapter is linked with non-CORBA resources and converts non-CORBA resources into CORBA resources, it supports CORBA resource attributes such as Message Registration, Message, and LifeCycle. A structure of a resource adapter is shown in Fig. 3.

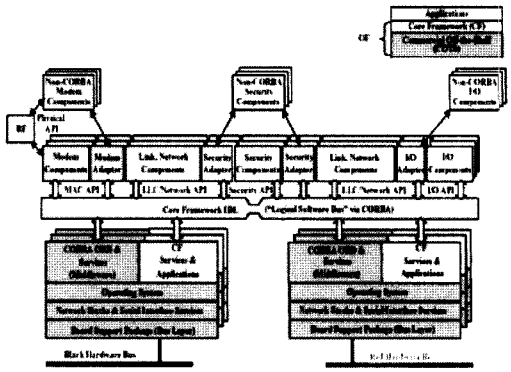

Fig. 1. Software structure of SCA.

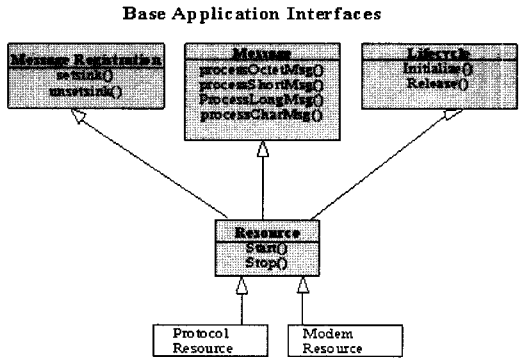

Fig. 2. Model of resource in SDR prototype.

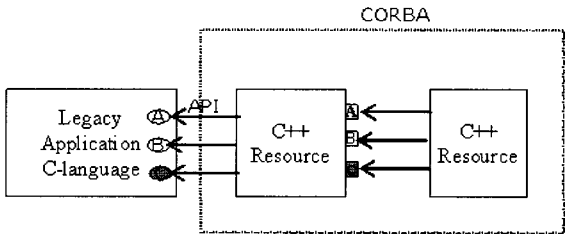

Fig. 3. The structure of resource: Adapter.

A protocol resource can be developed with C/C++ /Java. Therefore, a protocol adapter is needed to make links among resources. The resource adapter uses Function calls because the protocol is programmed with Specification and Description Language(SDL) in SDR Prototype, and in case it is developed with other languages, it uses wrapping with Shard Memory Mechanism for protocol adapter using API(C language). Shared Memory Mechanism is mainly used for performance enhancement, where speed is a priority. Therefore, the messages are transferred to modem adapter by CORBA API call methods in a protocol adapter.

Also, the message of modem adapter should be sent to DSP, while it is not possible to transfer messages directly into the DSP hardware. Therefore, the modem adapter is used to handle both CORBA messages and direct hardware accessing messages at the same time. Fig. 4 shows Modem Adapter and Protocol Adapter structure.

For the SDR prototype, the domain manager is designed to manage the resource states and the configuration of the virtual connections with the CORBA Interface.

3-2-1 Protocol: Adapter and C API

The protocol adapter transfers messages and data to/from a modem adapter using ProcessOctetMsg. These received messages are transferred to "Radio_Interface_Protocok_task" by calling a "Receive_Data_From_MA()" function, and send messages to Modem Adapter by calling a "Send_Data_to_ Modem_Adapter( )".

Fig. 5 shows that a protocol creates a "Protocol" resource as an instance and registers it in an Object Request Broker(ORB). After the protocol resource binds with a modem adapter, it transfers messages and data to/from the modem adapter. When a modem adapter calls SendOctetMsg, the protocol automatically calls processOctetMsg and transfers data.

3-2-2 Modem: Adapter and HPI

A modem adapter receives messages from CORBA resources(Protocol resources), non-CORBA resources, Interrupt Service Routine(ISR), and Domain Manager. The basic role of the modem adapter is to write to

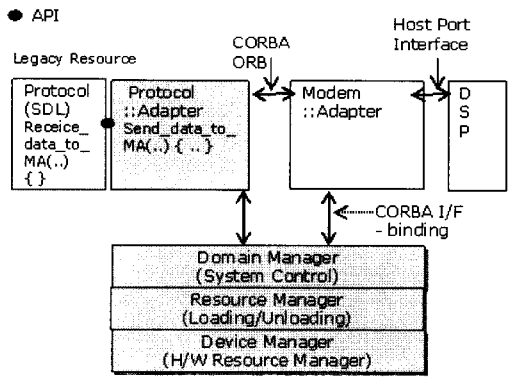

Fig. 4. Modem adapter and protocol adapter in SDR prototype.

```
Class ProtocolImpl:public ResourceImpl
{
 public:
 void processOctetMsg(...)
 {
   FromAntenna:
       ...
   ToAntenna:
       ...
 }
}
```

```
void Protocol(void)
{
  ProtocolImpl resource("Protocol");
  boa → object_is_ready(&resource);
          ...
  boa→ impl_is_ready();
}
```

Fig. 5. Code of protocol adapter.

specific addresses in DSP hardware according to the message/data header received from the protocol resources using Host Port Interface(HPI) in DSP. When the modem adapter gets an interrupt from DSP, it reads data from Rx_Data address in DSP and transfers data/messages to the protocol resources. Fig. 6 shows that modem adapter creates "Modem Adapter" resource as an instance and registers it in an ORB.

3-3 Resource Deployment and Duplex Connection

SCA defines the name connection point as Port, which is simply name connection point, however this is just conceptual and there're no details of the implementation.

We consider ports that actual resources providing connection functionality. In CORBA, the duplicate () function is available for making a copy of a resource references for duplex connections between Protocol resource and Modem resource, while the release () function is used to free the local memory used by a pointer. Fig. 7 shows Registration of resource via the proposed design methodology.

```
Class MaImpl:public ResourceImpl
{
 Public:
 void processOctetMsg(...)
 {
  switch((int)_options[0].value)
  {
   FromAntena:
      // Send MSG(Frame) to DSP
   ToAntena:
      //sendOctetMsg(...);
 }
}
```

```
void MA(void)
{

 MAImpl resource("MA");
 boa->object_is_ready(&resource)

 FOREVER
 {
  if (shared_stuff->written_by_any)
  .....
 }

}
```

Fig. 6. Code of modem adapter.

```
void ResourceImpl::setSink(CF::Resource_ptr _pushSink,
const CF::ResourceID_Type& _destinationResource)
{
 if (_destinationResource.direction ==
      CF::FROM_ANTENNA)
 {
   fromANT = CF::Resource::_duplicate(_pushSink);
 }else {
   toANT = CF::Resource::_duplicate(_pushSink);
 }
}
```

Fig. 7. Registration of resource.

A Download agent then downloads resources from a server and saves them into a domain Profile, which next sends "COMPLETE_MSG" to a Domain Manager. After receiving a complete-message, the domain manager parses the domain profile, and it deploys downloaded objects and then makes virtual duplex circuits. Fig. 8 shows virtual duplex circuits function.

A resource manager loads these resources and the domain manager copies the pointer of resources in binding order. It then sends "COMPLETE_MSG" to User Interface(UI).

Therefore SDR prototype is designed for all resources to communicate without the interrupt of a domain manager or other managers once the virtual circuit is made.

3-4 Simulation Evaluation of Communication Time of between Each Resource

We measured the interval time between protocol and DSP memory. Our experiment is to use a one-way operation passing 2340 bytes/sec of data.

- System Components for Timing Measure
- CORBA: VisiBroker-RealTime in Borland

```
void DomainManagerImpl::createVirtualCircuit ( const
CF::ConfigurationRequestType& _configurationRequest )
{
....
// make connection , direction is FROM_ANTENNA
for ( j = 0 ; j < i-1 ; j++ )
{
   rsc[j]->setSink(rsc[j+1],
  (CF::ResourceID_Type&)destResource) ;
}
...
// make connection, direction is TO_ANTENNA
for ( j = i-1 ; j != 0 ; j-- )
{
   rsc[j]->setSink(rsc[j-1],
   (CF::ResourceID_Type&)destResource) ;
}
....
}
```

Fig. 8. Virtual duplex circuits function.

- RTOS: VxWorks 5.4 in Windriver
- CPU: MPC 860 in Motorola
- DSP: TMS320c6202 in Texas Instruments.

[Simulation Results]

Assuming that the frame rate is 2340 bytes per second, the average value of data transmission was measured to be 252886 ns (0.252886 ms), 125000 ns as the minimum, and 375000 ns as the maximum. It took 0.25 ms for transmission from Protocol to DSP Memory through MPC 860 using Host Port Interface and CORBA ORB. Simulation time is shown as below in Fig. 9. And time interval between Protocol adapter and Modem Adapter was measured to be less than 0.007 ms, of which the value is negligibly small and, as a result the design structure is considered to be efficient.

The result of the simulation guarantees SDR prototype that is appropriate to the real-time communication using our design method.

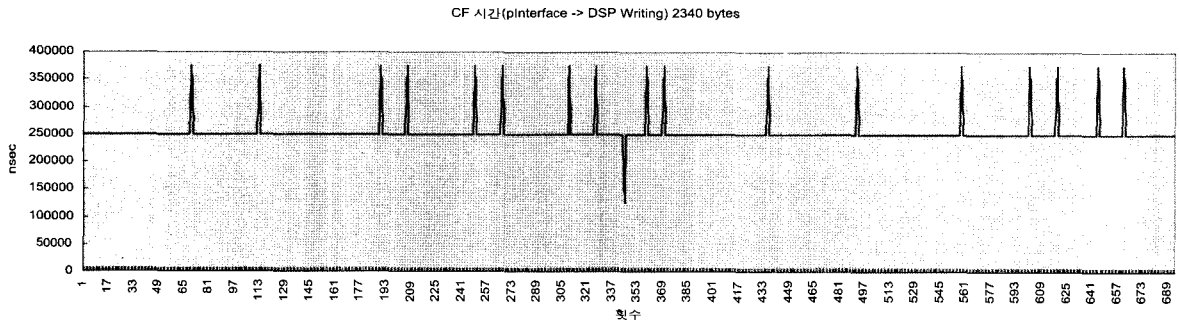

Fig. 9. Simulation time.

Ⅳ. SDR Prototype

In this Section, we describe our SDR-prototype system.

4-1 Core Framework

CF is the essential("core") set of open application-layer interfaces and services to provide an abstraction of the underlying software and hardware layers for software application designers[10],[11]. Our CF of SDR Prototype System consists of:

Three Base Application Interfaces(Message, Life-Cycle and resource) that can be used by all waveform applications.

Framework Control Interface(Domain Manager resource Manager, Device Manager, Download Agent) provides control of system via Real-Time CORBA[12]. Domain Manager manages the software Applications that a type of resource and Device Manager control system's internal hardware device.

Framework Services Interfaces(File Manager, File System and File) support core and non-core applications via Real-Time CORBA and POSIX.13 complaint RTOS(Vxworks)[13]. A Domain Profile(Hardware Profile, Software Profile) describes properties in radio system. Domain Manager handles domain Profile using Extensible Markup Language(XML) in addition. Total message flow diagram for start up from Download Server to DSP is illustrated in Fig. 10.

4-2 Protocol Download Stack and Software

4-2-1 Access Protocol

The physical layer of SDR prototype system supports only a dedicated transport channel for data transfer. We do not implement Radio resource Control(RRC) stack since there is no additional channel to make a RRC connection. To provide seamless data transfer from higher layer to physical layer, the Medium Access Control(MAC) and Radio Link Control(RLC) stacks are implemented in the SDR Prototype System. The RLC stack receives data packets from PPP stack and passes them to the MAC stack. The two stacks(RCL & MAC) perform the packetizing and depacketizing for the Service Data Unit(SDU) of Dedicated Channel(DCH) and Dedicated Traffic Channel(DTCH), respectively. In addition, MAC calculates the number of PDUs to be handed to Physical layer in each TTI. In the Next SDR system, only the minimal functions of each stack will be implemented as described in the following subsections. Data flows through layer 2 are characterized by the applied data transfer modes on RLC(acknowledged, unacknowledged and transparent transmission) in combination with the data transfer type on MAC, i.e., whether or not a MAC header is required. The case where no MAC header is required is referred to as "transparent" MAC transmission. Both the acknowledged and unacknowledged RLC transmissions require a RLC header. As explained above, the SDR prototype system has only a DTCH logical channel mapped to DCH for continuous DTCH data stream. The continuous DTCH data stream is segmented into transport blocks on RLC. When no multiplexing of DTCH on MAC is applied, both RLC and MAC sub layers are transparent, i.e., Fig. 11 is applicable. Thus, in the SDR prototype transparent RLC and MAC layers are implemented.

4-2-2 Download Protocol

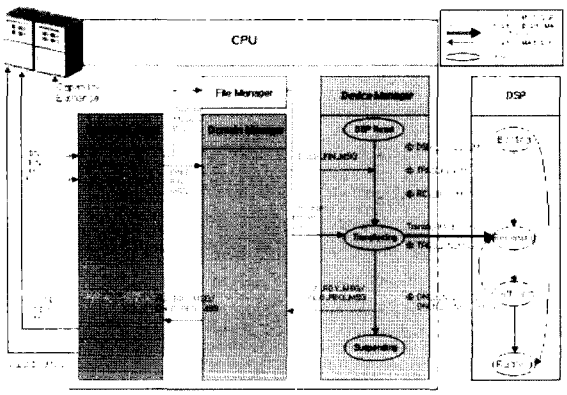

Fig. 10. Message flow diagram for start up.

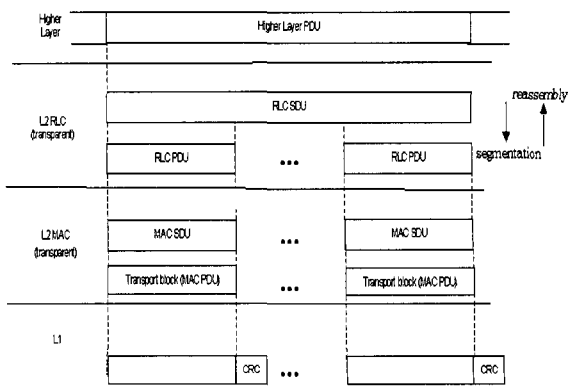

Fig. 11. Data flow for transparent RLC and MAC.

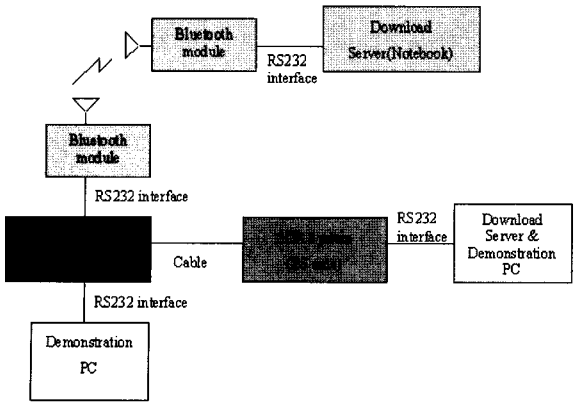

Fig. 12. Software download architecture.

This Section presents the architecture of software download for the SDR prototype H/W platform, where the downloaded software works as the modem software in the form of executable file with relocatable information and is loaded into memory[15]. The software consists of two parts, DSP and CPU, which run in the correspondent processor, respectively. The Fig. 12 shows the architecture of software download process.

As can be seen, we use Bluetooth module for a communication link between the download server and SDR prototype[16],[17]. Bluetooth is one of the most possible solutions for download service in near future since most handset will have the Bluetooth communication link[6],[7]. We need a communication link to download a waveform application, but we cannot use the waveform link prior to the downloading. Thus, a popular standard link such as Bluetooth is necessary for the software download.

There are four modules for downloading, which are modem adapter, DSP main/DSP data, protocol adapter, and protocol. These four modules are compressed to be saved in flash memory for downloading before reconfiguration along with decompression. The compressed sizes of each module are as follows:

Modem Adapter: 58 KB(decompressed size: 208 KB) DSP Data: 160 B(492 B), DSP Main: 13 KB(29 KB), Protocol: 188 KB(803 KB). It was measured that it took less than 30 seconds for downloading with Bluetooth. And it took less than 20 seconds for reconfiguration of downloaded modules after messaging, which is a highly efficient process.

4-3 Hardware and Modem

DSP chip plays a role of operational processing of 3GPP modem part in a wide range of functions, in which TMS320c6202(2000MIPS) from TI is used. In timing signal generator, both slot and frame sync signals can be achieved at chip clock speed of 3.84 MHz×8, which consists of four counters such as scaling down counter(SDC), 1/8 chip counter(CC), slot counter(SC), and frame counter(FC).

In SDR prototype systems, one DSP chip is used for baseband signaling process, where the current DSP operation speed[18] does not meet the chip rate(3.84 Mcps) for 3GPP applications[14]. Thus, in Phase 1 Systems the chip rate is scaled down by 1/10 and the number of operations is then decreased. The SDC is for reduction of chip rate in systems, which can decrease the speed of chip clock by 1/10, 1/20 or 1/50. As in Fig. 13, the length of chip, slot or frame is scalable by varying the value of SDC. On the other hand, the SDC is not needed for 3GPP specifications, and hence it will not be used in systems for next phase. FPGA is also used to create radio frame and slot timing.

To speed up communication process between CPU and DSP, the following schemes are utilized in implementations

- Shared memory is shared between CPU and DSP (shared memory is allocated in each module's memory map)
- Command, Status, Data areas are shared
- All transfers are recognized by interrupt and memory flag

We used MPC860 CPU of the Motorola in order to

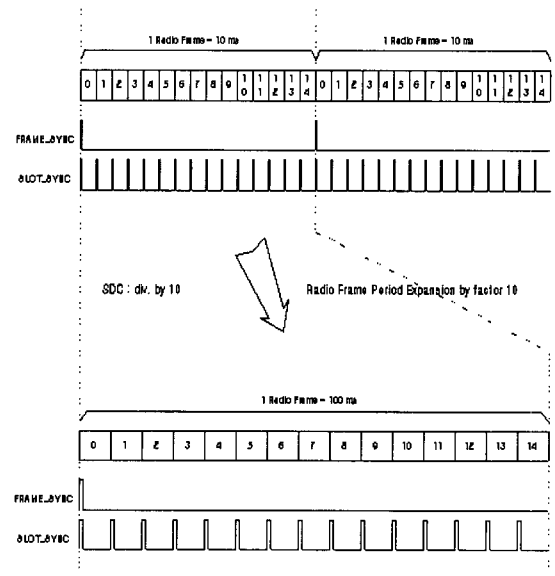

Fig. 13. Functionality of scaling down counter.

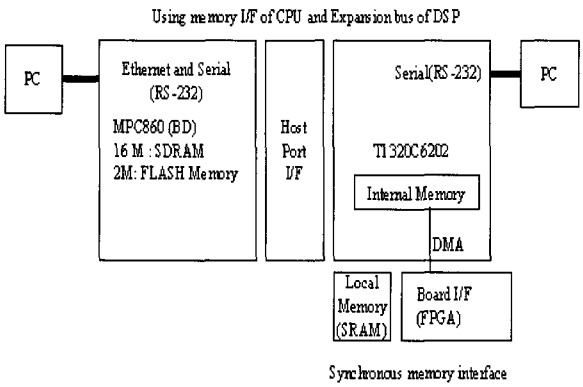

Fig. 14. Configuration of hardware system.

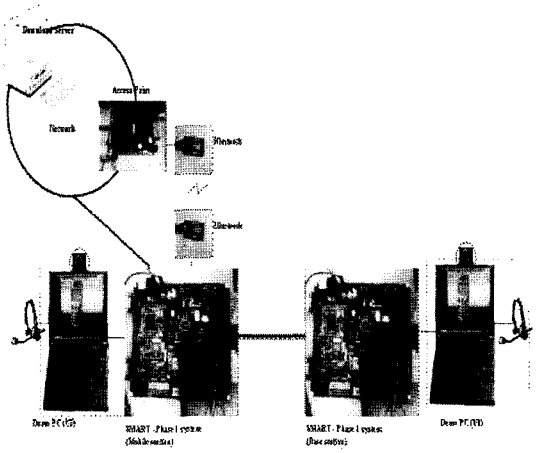

Fig. 15. Demonstration environment of SDR prototype.

run CORBA in middleware Software, CF and RRC, RLC, MAC layer Protocol. Fig. 14 shows configuration of Hardware System.

4-4 Total Framework in SDR Prototype

Fig. 15 describes the demonstration environment in the view of protocol side. Once the waveform application(3GPP FDD modem application) is installed in the MobileStation(MS), the transparent mode data transfer between two RLC peer entities can be performed whenever the demo program sends packets to RLC layer. As can be seen, the data generation from demo program in the Demo BaseStation(BS) PC is transferred to the SDR BS Platform through serial interface in the form of PPP packets. The arrived packets are converted to RLC packets by PPP to RLC converter and come down to the modem layer. The modem transfers modulated signals to SDR MS Platform through hard wired line. The arrived signals are changed into data packets and go up to RLC layer in the reverse way done in SDR BS Platform.

V. Conclusion

We have presented resource design methodology and details of Adapter implementation for CF and, SDR prototype systems which interact with resources implemented in different program languages and processors in legacy environments through our design methodology. The resource objects were to be controlled and managed by Domain Manager and Device Manager. Protocol Adapter and Modem Adapter are reconfigured by downloading from server using "download agent" which is implemented by the SDR download protocol. We used a Bluetooth for downloading software modules from Download Server.

We measured the time interval between protocol adapter and DSP. The simulation results show that our proposed system is appropriate to the real-time communications.

As a next step, the full function of SDR prototype using CF and download protocol will be implemented based on SCA 2.2 specification. In addition, Reconfigurable Engine(RC) will be used in the future work. The above implementation work will be adapted to a base station as well.

References

[1] *Modular Software-Programmable Radio Consortium Software Communication Architecture Specification,* MSRC-5000SCA V2.2, November 2001. http://jtrs.army.mil/pages/sections/technicalinformation/fset_technical_sca.html

[2] J. Mitola, "The Software Radio Architecture", *IEEE Communication Magazine*, vol. 33, no. 5, May 1995.

[3] Joseph Mitola, *Software Radio Architecture-Object-Orient Approaches to Wireless Engineering*, John Wiley & Sons, Inc. SDRF Technical Report 2.1, November 1999.

[4] Byron Tarver, Annamarie Miller, *Software Defined Radios(SDR) Platform and application programming interface(API), IEEE*, 2001.

[5] Joint Radio Tactical System(JTRS). http://jtrs.army.mil

[6] Bluetooth Special Interest Group, *Specification of*

the Bluetooth System, Version 1.1. http://www.bluetooth.com

[7] Jaap Haartsen, ERICCION Radio System B. V. "The Bluetooth Radio System", *2000 IEEE Personal Communications.*

[8] SDR Forum. http://www.sdrforum.org

[9] *The Common Object Request Broker: Architecture and Specification.* Object Management Group. http://www.omg.org/

[10] Software Defined Radio Forum Mobile Working Group, "Distributed-Object Computing Software Radio Architecture", vol. 1.2, September 16, 1999.

[11] *SDR Forum,* Technical Report 2.1, November 1999.

[12] Borland® VisiBroker®-RT. http://www.borland.com/visibroker_rt/

[13] Tornado and VxWorks. http://www.windriver.com/

[14] *The 3rd Generation Partnership Project(3GPP).* http://www.3gpp.org/

[15] C. Noblet, A Aghvami, *Assessing the Over-The-Air Software download For Reconfigurable Terminal,* IEE Library, 1998.

[16] Klaus Moessner, Seiamak Vahid and Rajim Tafazoli, "A Minimum Air Interface Implementation for Software Radio based on Distributed Object Technology".

[17] M. Cummings, S. Heath, "Mode Switching and Software Download for Software Defined Radio: SDR Forum Approach", *IEEE Communication Magazine.*, vol. 37, Aug. 1999.

[18] Tim Hentshel, Gerhard Fettweis, "Sample Rate Conversion for Software Radio", *IEEE Communication Magazine,* Aug. 2000.

Jin-Ho Son

Chief Research Engineer Information Technology Lab. LG Electronics Institute of Technology Feb. 1993-Present. Ph.D. candidate in School of Electrical and Computer Engineering, Sungkyunkwan University, Korea Mar. 1999-Present. M.S. Mar. in School of Electrical and Computer Engineering, Sungkyunkwan University, Korea. Mar. 1991-Feb. 1993. B.S. Mar. School of Mechanical Engineering, Sungkyun-kwan University, Korea Mar. 1987-Feb. 1991.

[Research Interest]

Software Defined Radio: Software Communication Architecture

Personal Area Network: Clustering, Scatternet, Mobility, Qos in Zigbee, Bluetooth, WLAN

Home Network: PLC Communication, Middleware(UPnP, Jini, Havi)

Tai M. Chung

Professor School of Electrical and Computer Engineering SungKyunKwan University Sep. 1995-Present. Ph.D. Electrical and Computer Engineering Purdue University Aug. 1990-Aug. 1995. M.S. Electrical Engineering and Computer Science University of Illinois at Chicago Sep. 1984-Dec. 1987. B.S. Mathematics and Computer Science University of Illinois at Chicago Sep. 1982-Jun.1984 B.S.Electrical Engineering Yonsei University Mar. 1977-Feb. 1981.

[Research Interest]

Security Management Technology: Integrated Network Security Control System, Intrusion Detection and Control Systems, Grid Security

Network Management Technology: Virtual Private Network Management, Accounting and Billing Management Active Network Management

Ad Hoc Networks: Ad-hoc Network Protocols, Network Mobility