Reconfigurable Context-Sensitive Middleware for ADS Applications in Mobile Ad Hoc Network Environments

Stephen S. Yau and Fariaz Karim
Computer Science and Engineering Department
Arizona State University
Tempe, AZ 85287, USA
{yau, karim}@asu.edu

Abstract

The emergence of inexpensive and low-power wireless communication hardware and various handheld, wearable, and embedded computing technologies is making computing and communication devices more mobile and ubiquitous. Due to the mobility and high-density of network-enabled devices, short range mobile ad hoc networks (MANET) are instantaneously and autonomously formed to facilitate exchange of information. In MANET, interactions among the devices are driven by constantly changing contextual and environmental conditions, rather than by the applications resident on the devices. This trend makes Autonomous Decentralized Systems (ADS) a desirable architecture for facilitating ad hoc communication among mobile devices. In this paper, Reconfigurable Context-Sensitive Middleware (RCSM) is presented to facilitate ADS applications in MANET.

Keywords: Autonomous Decentralized Systems, ad hoc networks, ubiquitous computing, middleware, context-aware applications

1. Introduction

Improvements in wireless communication technologies, and increased use of handheld, wearable, and embedded devices are making computing and communication devices increasingly mobile and more ubiquitous. The underlying computing infrastructure of these devices is usually based on MANET. A MANET is a collection of autonomous mobile nodes [1], such as wearables, handhelds, and other embedded devices. The nodes are free to move arbitrarily (e.g. a wearable device moves with a living carrier), usually have bandwidth and energy constraints, and are equipped with multiple sensors. The topologies in MANETs are dynamic, and usually no dedicated network connectivity devices, such as routers or a GSM base station, exist to assist the nodes in data communication. This characteristic distinguishes MANETs from other wireless networks with fixed routing infrastructure, such as PCS and GSM cellular networks. Applications of MANETs include mobile command and control centers, public meeting places, ubiquitous computing environment [2], especially in cases where a fixed network infrastructure is either neither available nor feasible, and a rapid communication setup and termination is necessary. Based on physical environmental conditions and other stimulus, nodes in this environment form numerous webs of ad hoc short-range wireless networks to exchange information, and react in a transparent fashion. Devices operating in MANETs have memory and processing constraints, need real-time performance, and have close coupling with specific hardware components. In addition, the characteristics of applications executing in these devices are:

1. Context and resource awareness: Applications use various data about the user and the surrounding environment to adapt their behavior and interaction.
2. Impromptu and volatile distributed interaction: Communication channels among applications tend to be instantaneously established and terminated due to changing contexts and node mobility.
3. High-density: As the number of mobile computing nodes increase, application interaction tends to increase exponentially.

A high-degree of correlation can be drawn between applications with the above characteristics and ADS. Although the ADS concept is rooted in manufacturing and factory automation systems, the properties of ADS are also applicable to devices operating in MANETs and ubiquitous computing environment. The properties are equality, locality, and self-containment [3].

The broad heterogeneity of embedded devices in MANETs demands the need for using object-oriented middleware implementations, since they are useful to rapidly develop and effectively provide interoperability among distributed software over heterogeneous platforms and environments. However, existing ADS architectures [4-6] do not support the characteristics, such as context-awareness and ad hoc interaction among applications. Moreover, except few ADS approaches [5,6], most

0-7695-1065-5/01 $10.00 © 2001 IEEE

Currently, ADS implementations do not take advantage of object-oriented middleware implementations. On the other hand, industry standard middleware specifications, such as CORBA, COM, and TINA-C, and their derivatives, such as Real-Time CORBA and minimumCORBA [7,8], are not sufficient to support ADS applications in MANETs due to the similar reasons mentioned above. In addition, the emergence of MANET transport and routing protocols, such as Bluetooth, IrDA, HomeRF, PicoNet, Personal Area Networks, AODV, DSDV [9-14], etc. make most existing middleware implementations inadequate.

To address the limitations mentioned above, we will present the Reconfigurable Context Sensitive Middleware (RCSM) [15] to facilitate ADS applications in MANETs. RCSM is co-designed in software and reconfigurable hardware, and combines the power of abstraction provided by mainstream middleware specifications (e.g. CORBA) with the performance and economy of hardware to facilitate ad-hoc communication among devices in MANETs.

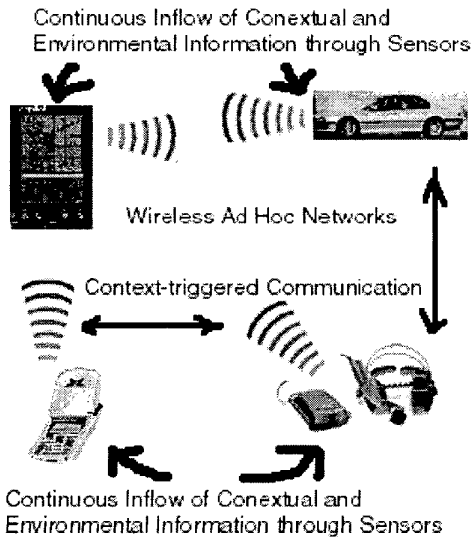

Figure 1: ADS Applications in Mobile Ad Hoc Environment

2.Our Approach

The RCSM has two key features, which facilitate the development and runtime operations of ADS applications in MANETs:

(1) RCSM provides the facilities for *context-triggered* interaction among ADS applications. This type of interaction is initiated based on specific environmental conditions, rather than by explicit application invocation. Figure 1 shows an example of context-triggered interaction among devices that run ADS applications.

(2) Unlike other middleware implementations, the performance-intensive parts of RCSM are implemented in Field Programmable Gate Arrays (FPGA) to achieve high-performance for embedded devices. FPGAs are a kind of reconfigurable hardware components, which can be reprogrammed after fabrication to achieve flexibility and customizability.

As shown in Figure 1, mobile nodes use short-range (some nodes may be capable of both short-range and long-range communication) wireless communication and on-board sensor hardware to detect the surrounding environmental and contextual data. Individual devices analyze these data according to the application requirements to transparently establish communication to enable autonomous exchange of information.

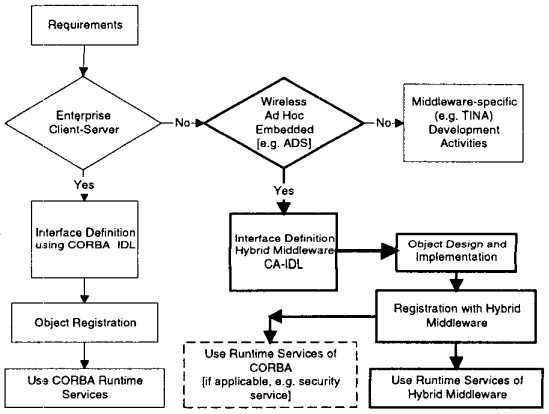

Figure 2: Development Methods using CORBA and RCSM

Figure 2 simultaneously shows the development processes using CORBA and RCSM, where the rectangles with heavy outlines represent the development process using RCSM. The main contrast is in the interface definition phase, where additional information about the target application is used. From developers' perspective, RCSM is architecturally compliant with CORBA specification. As such, existing CORBA services can be supplemented by RCSM to achieve more benefits.

Although RCSM has various capabilities, we focus in this paper on how its capabilities can be used from

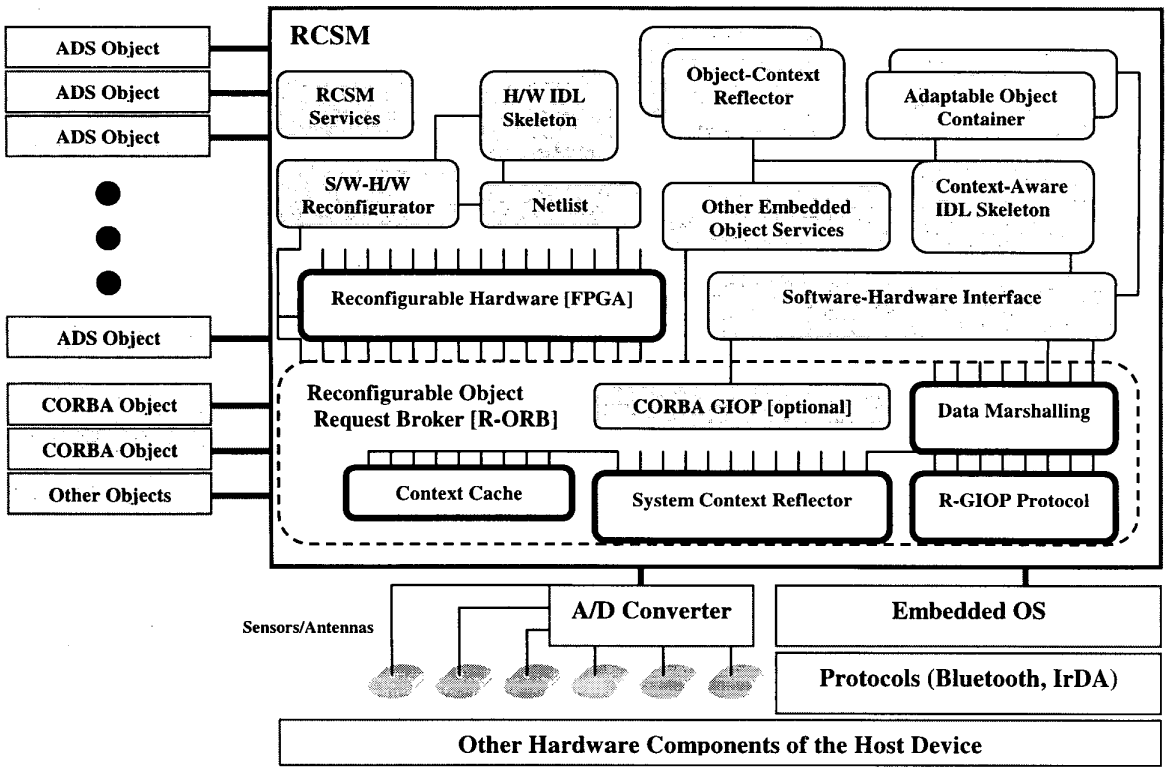

Figure 3: High-Level Architecture of RCSM

two different aspects – 1) during ADS application development, and 2) autonomous communication establishment in MANETs.

During ADS application development for MANET environments, RCSM is used in the following ways, as shown in Figure 2:

1. Specify the object interfaces with associated contexts based on which the individual methods will be activated remotely. The interfaces and necessary contexts are identified based on application requirements.
2. Generate a Context-Reflector for each ADS object.
3. After the application is developed, register with the RCSM.

To autonomously establish communication among ADS applications, the following services of RCSM are used:

a. Provide Context-aware Interface Definition Language (CA-IDL) for specifying context relationships and associating them with individual ADS object methods.

b. Provide a *context-reflection* facility to enable application objects to manage their context-awareness dynamically.

c. Use RCSM Object Request Broker (R-ORB), which uses dynamically changing contextual information instead of application-initiated actions to establish impromptu ad hoc communication among remote objects.

We will provide a brief overview of RCSM and elaborate the development steps and corresponding middleware capabilities in the following sections.

3. Overview of RCSM

As shown in Figure 3, an RCSM [15] is co-designed in software and hardware, and combines the power of abstraction provided by mainstream middleware specifications (e.g. CORBA) with reconfigurable hardware to provide adaptive object services for MANET environments. Specifically, RCSM is an embedded middleware with the following characteristics:

a. Uses dedicated reconfigurable hardware to provide core middleware services.

b. Provides a context-based reflection and adaptation triggering mechanism suitable for devices in MANETs.
c. Provides a programming abstraction to allow software-hardware interface specifications and reconfiguration.
d. Provides an Object Request Broker and Inter-ORB protocol that are context-sensitive and invokes remote objects based on contextual and environmental factors.
e. Can be used to supplement existing middleware specifications, such as CORBA and COM.

In the following, we will give the rationale for taking a hybrid approach to designing and implementing this middleware:

High-Volume Communication Among Networked Embedded Devices: Embedded devices currently constitute 98% of all the computers in the world [16]. The number is expected to grow as embedded devices are increasingly networked. As the trend continues, the volume of peer-to-peer communication will increase due to the ubiquity of these devices. To address this, RCSM uses dedicated hardware to implement R-ORB, which is the Object Request Broker (ORB) of RCSM. In most cases, Object Request Brokers are critical to provide predictable services for performance-intensive applications [17].

Increased use of Reconfigurable Hardware: Reconfigurable hardware units, such as Field Programmable Gate Arrays (FPGA) can currently have up to 500,000 gates, which usually doubles in number in every 18 months. Due to the flexibility of reconfiguration, these units are useful for application-specific customization in communications, military, and intelligence applications that often reside in different embedded devices. Moreover, it is shown that reconfigurable processors are useful in computations that are regular [18]. Examples include signal and protocol processing, encryption and compression operations. RCSM uses reconfigurable FPGAs to provide improved performance and customizability of ORB-specific operations, such as data marshalling, connection management, and protocol adaptation.

Use of Software in Traditionally Hardware-Only Domains and Vice Versa: Recent development in *software radio*, *software-based DSP*, etc. has indicated increased software usage in special-purpose embedded devices, which have traditionally been implemented in completely special-purpose hardware [19]. On the other hand, there is an increasing demand for using hardware to implement reusable and performance intensive functionality.

4. Context-aware Interface Specification

Context is considered any detectable attribute of a device, its interaction with external devices, and/or its surrounding environment. For example, a device's current geographical location, its history of interaction with other devices, or its available battery power are some of its contexts. Context-awareness is the ability of a device to detect its current contexts and changes in any contextual data.

Since a device may have multiple ADS objects, it is necessary to associate objects with their preferred contexts for the purpose of activation and communication with other objects. The process involves addressing the following issues - context-representation, semantic interpretation of contextual data, relationship among multiple contexts, and relationships between object activation and contexts.

4.1 Context representation: We define a *context* as a tuple <ai, aj, ak, ... an, tm> of size n, where n is the number of unique contextual-data sources present in the device. Each variable *ai* in the tuple represents a value, which is valid for the corresponding type of context. The variable tm represents the time of the tuple creation time. To illustrate the use of the tuple, we consider a device with equipped with five different sensors. These sensors are responsible for location tracking, direction, mobility rate of the device, sensing the amount of light in the surrounding environment, and the amount of available power in the device. A tuple corresponding to this context created at time 06:45 PM can be represented as <(10,20), north, 3, 10, 10%, 18:45>.

4.2 Semantic interpretation: The semantic interpretation of the contextual data is application-specific, and hence should be part of the object implementation rather than a feature of the core middleware services. For example, a device with GPS location tracking capability usually receives its current location in terms of longitude and latitude coordinates. However, depending on the application, the coordinates need to be mapped to a map of a small neighborhood, city, or a state.

4.3 Relationships among multiple contexts: To be able to specify relationships among multiple contexts, we have generated several timed-regular expression operators to specify temporal relationships among multiple context tuples. Most of the operators, as summarized in Table 1, are semantically similar to the basic regular expression operators. The timed-

regular expression operators can be used to specify temporal relationships among different contexts. Due to space limitation, we do not provide the mathematical definition of these expressions in this paper.

Operator	Example	Description
Union: +	[(C1 + C2) t]	Either C1 or C2 is true in every discrete time interval t
Concaten ation: ^	[(C1^ C2) t]	Both C1 and C2 are true in every discrete time interval t
Repetition: *	[(C1*) t]	C1 is repeatedly true in every discrete time interval t
Precedence: ->	[(C1 -> C2) t]	C2 becomes true within t time units C1's being true

4.4 Object Activation and Contexts: To address this issue, we are developing Context-aware Interface Definition Language (CA-IDL). It is based on CORBA 2.3 Interface Definition Language specification. It uses additional language constructs to associate a method of an ADS object with a particular context. To accomplish this, the following two rules are applied:

- Each method of an interface should be annotated with either [incoming] or [outgoing] tags. An [incoming] tag signifies that the corresponding method should be invoked by an external entity. An [outgoing] tag signifies that the method should be invoked on the remote object.

- Each method of an interface should be annotated with [activate-at-context x] tag, where x represents a context-tuple or timed-regular expression for contexts, as described in Table 1. Note that individual objects can also change the value of x during runtime.

The remaining parts of interface definition are similar to CORBA IDL definition.

4.5 Context Reflection

Context-reflection (CR) is an object's ability to reflectively discover different contextual information of a device. The information can be used to perform adaptation, if necessary. In the RCSM, the context-reflection facility is provided in two levels through the *device-context-reflector* component and *object context-reflector objects.*

Device-Context-Reflector: The *device-context-reflector* component, which will be implemented in hardware, interacts with on-board sensors and antennas that periodically generate primitive contextual-data from the surrounding environment. The component provides a well-known data access interface, which provides different methods to enable an object to use these methods to analyze the context-awareness of itself or the host device.

Object-Context-Reflector: The *Object-Context-Reflector* (O-CR) is responsible for managing and analyzing object specific contextual data. As such, there is a 1-1 mapping between an O-CR and an object, which is registered with RCSM. It is responsible for triggering a context-match when the criteria specified in the interface definition of an object or through the *CR_ContextTag* method. The object implements *CR_ContextTag* and *CR_ContextRegister* methods. Although the implementations of these methods are same for every instance, the storage and the interpretation of the contextual-data is unique for every application object.

Synchronous Context-Tuple Creation: To facilitate context-reflection, it is necessary to always be aware of the changes in the surrounding environment. In case of RCSM, a new context tuple is created in a synchronous fashion. The activity consists of collecting data using the device-context-reflector component. However, a demand-driven mechanism for creating tuples would also be straightforward to incorporate by implementing additional interrupts.

5. Context-Triggered Object Interaction

One of the unique characteristics of RCSM is its ability to connect objects of different devices based on the changing conditions of the surrounding environment. This is referred to as context-triggered object activation. This method of communication directly facilitates the equality and locality properties of ADS objects. It is in sharp contrast with existing middleware, which establishes communication based on explicit request of an object (e.g. a client in CORBA).

In this section, we briefly provide an overview of Object Request Broker of RCSM - R-ORB, and particularly its R-GIOP protocol and associated messages that are relevant in context-triggered communication establishment.

R-ORB: The fundamental characteristics of the R-ORB are closely tied with the distributed application or information services in ubiquitous and context-aware communication. A device has only one

instance of R-ORB. Since it provides performance intensive and reusable functionality, a part of the R-ORB will be implemented in hardware.

Based on the requirements in Section 1, we focus on R-GIOP, which is the inter-orb protocol to enable communication between two instances of R-ORBs in two different devices.

R-GIOP Protocol: The R-GIOP is designed to operate any connection-oriented protocols of various kinds, which are emerging as protocols for devices in short-range ad-hoc mobile networks. Some notable examples of these protocols include Bluetooth, IrDA-Lite IrORBEX, Ir-TinyTP, PicoNet, HomeRF, and Personal Body Networks.

Symmetrical Communication: R-GIOP is a symmetrical communication protocol. Symmetrical communication implies that a communication channel, which can be established by either of the nodes. To clarify the concept, we use CORBA General InterOrb Protocol (GIOP) as an example of asymmetric communication protocol, which defines two roles – client and server. The client object creates a request, which is processed and replied by the server through the GIOP. Clients usually have reasonably complete information about the servers. Moreover, most enterprise-oriented distributed applications can be easily implemented by adopting a client-server communication model.

Unlike CORBA GIOP, R-GIOP does not define any distinct roles with respect to a connection between two remote applications. Impromptu connection establishment is a norm rather than an exception in MANETs due to the reasons described earlier in the paper. Applications running in ad hoc environment are completely autonomous, and thus it is hard to characterize them as either clients or servers.

ORB Connection Establishment: ORB Connection establishment is the process of setting up a communication channel between two remote ORBs. The communication channel is used to exchange information between the objects hosted by the ORBs. In RCSM a connection is established based on the following three conditions:

- C1: Two devices are within each other's wireless range.

- C2: A *context-match* (Section 4) event occurs in both devices, which means each device has at least one object, which can be activated in the current contexts.

- C3: An *identity-match* event occurs in either device, which means two remote objects are suitable to exchange information with each other. By suitability, we mean matching of the object interface signatures, Radio Frequency Identification (RFID) matching, etc.

Based on these conditions, the process of connection establishment can be described as follows:

Consider two hypothetical devices P and Q.
1. P and Q establish a low-level connection upon discovering each other using a connection-oriented protocol, such as Bluetooth or IrDA Lite.
2. The R-ORBs of P and Q exchange the identifiers of the objects resident in their storage.
3. If Conditions C2 and C3 are true, ORBs notify each other.
4. Both ORBs in P and Q activate the particular objects in their respective devices.

ORB Connection Termination: A connection can be terminated if one of the following occurs:

- The devices are no longer within each other's range due to factors, such as mobility.
- At least one of the devices is not in a context, suitable for its objects to communicate.
- The exchange of information is complete between the objects.

6. An Example

In this example, we illustrate the techniques used in RCSM by showing how ADS applications in three devices autonomously exchange information under different environmental conditions. The requirements of the system are given below:

R1: The system is a collection of PDAs, mobile phones, and other information sources, with wireless capabilities.
R2: It is a part of wide area information service.
R3: Devices have capabilities, such as exchanging business cards of their owners, downloading shopping information, etc.
R4: The functionality described in R3 is activated autonomously when the devices are within each other's range, and in specific contextual and environmental conditions.

The example is divided into two parts. First, we describe individual devices along with built-in ADS objects, their interfaces, and their preferred contexts for activation. The first part also shows how context-aware interfaces can be specified using the CA-IDL of RCSM. Second, we describe two scenarios, and

show how respective the RCSMs of the devices facilitate autonomous context-based communication.

- Overview of the Devices:

Device A: It is a Personal Digital Assistant (PDA) with on-board location sensor, a light sensor, and a noise sensor. As such, the device is capable of detecting three different contextual data. Among different software packages, we consider the following objects:

- Object A1

```
//Name: Business Card Holder
//Define a context variable
context a (<<"-","high","high">>*)60;
//Method 1: activate when context a is true
[incoming] [activate at a]
        get_business_card (in string, ...);
//Method 2: activate when context a is true
[incoming] [activate at a]
  give_business_card (in string, ...);
```

- Object A2

```
//Name: Shopping Agent
//Define another context variable
context b (<<"A", "high", "-">>*)5;
//Method 1: Activate when context b is true
[incoming] [activate at b]
  download_sales_info (in ..., out ...);
```

- Object A1 has two methods. Both methods should be activated when a similar object is in range (possibly in another person's PDA) anywhere as long as it is daytime. It means, if the owner of Device A is nearby another person for 60 seconds and is in conversation (i.e. noise level is high), their business cards should be exchanged.
- Object A2 has one method. The *download_sales_info* method is activated when it is in location "A" and it is daytime. We assume the location information has been programmed in by user due to his knowledge about a mall in location "A".

Device B: It is a mobile phone with an ADS application, which also has Object A1 as shown above. The sensor capability of this device is the same as Device A.

Device C: It is a stationary beaconing device, which is installed near a shopping mall to broadcast information about ongoing sales in the mall. The device has the following ADS object:

- Object C1

```
//Name: Sales_Info_Distributor
//Define a context variable
context c <<"-","high","high">>30;
[outgoing] [activate at c]
  distribute_ current_sale (out string, ...);
```

- Object C1 continuously distributes ongoing sales information as long as it is daytime and noise level is high. Hence the object does not distribute sales data at night (when the mall is probably closed) or when the noise level is low (means no potential customer is in the vicinity).

- **Ad Hoc Interaction Scenarios**

Scenario1: Person 1 meets Person 2 in a conference at 2:00 PM. They engage in a conversation for several minutes. At 2:10 PM Person 1 leaves the conference. The following activities occur autonomously during the conversation:

Between 2:00 PM and 2:10 PM:

- Both Devices A and B detect each other's presence, and establishes a low-level communication channel, using protocols such as Bluetooth. The respective devices are notified.
- The R-ORBs of both devices establish a connection, and exchanges the object identities.
- A *context-match* event and an *identity-match* event occur in both devices. In Device A, the *context-match* event occurs since it matches context a, as defined in the interface of Object A1. Similarly, in Device B, matching of context a triggers a *context-match* event.
- The get_new_card and the give_business_cards are activated by the respective R-ORBs.
- The communication channel is terminated between the respective R-ORBs.

After 2:10 PM: [Person 2 is still in the conference]

- A *context-match* event continues to occur in Device B. However, since there is no device nearby (i.e. the person is alone), it is ignored and no method is activated in Device B.

Scenario2: Person 1 leaves the conference. The route to his next destination is near a mall, which is located at "A". Person 1's car is now at the vicinity of the mall, but Device A and Device C are not within each other's range.

- A *context-match* event occurs in Device A since it is in location "A", and it is daytime. However, no object is activated due to the absence of an *identity match* event, since there is no other device nearby.

Devices A and C are within each other's range.

- Both Devices A and C detect each other's presence, and establishes a low-level communication channel, using protocols such as Bluetooth.

- The R-ORBs of both devices establish a connection, and exchanges the object identities.
- A *context-match* event and an *identity-match* event occur in both devices. In Device A, the *context-match* event occurs since it matches context b, as defined in the interface of Object A2. Similarly, in Device C, matching of context c triggers a *context-match* event.
- The download_sales_info and current_sale methods are activated, and sales information is downloaded.

Device A is no longer within the range of Device C.
- The communication channel is terminated.

Both scenarios illustrate autonomous exchange of information among different devices based on environmental and physical situations.

7. Discussion

Currently, we are implementing the CA-IDL compiler (with Java and C++ mapping). Our current activities also include a hardware design model of R-ORB for Xilinx FPGAs using VHDL/Verilog to analyze the performance and different hardware-software interface issues of RCSM.

In this paper, we have addressed how ADS architecture can be supported in MANETs through RCSM. In particular, we have discussed context-aware interface definition, context-reflection, and autonomous context-based object interaction based on continuously changing environmental and contextual conditions. Further research needs to be done on hardware-software co-design of RCSM Middleware, adaptive object services through reconfigurable hardware, and providing deterministic R-ORB communication for real-time applications.

References:

[1] IETF, "Mobile Ad Hoc Networking (MANET)", March 1998, http://www.ietf.org/1id-abstracts.html.

[2] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing, *Comm ACM*, Vol. 36, No. 7, July 1993, pp. 75-84.

[3] K. Mori, "Autonomous Decentralized Systems: Concept, Data Field Architecture and Future Trends", *Proc. Int'l Symp. Autonomous Decentralized Systems (ISADS 93)*, March 1993, pp. 28-34.

[4] K. Mori, "Applications in Rapidly Changing Environments", *Computer*, Vol. 34, No. 3, April 1998, pp. 42-44.

[5] S. Yau and S. Mao, "A Framework for ADS Application Software Development based on CORBA", *Proc. Int'l Symp. Autonomous Decentralized Systems (ISADS 97)*, April 1997, pp. 113-120.

[6] S. Yau, N. Dong, and F. Karim, "Autonomous Decentralized System with Event Service for Information Services", *Proc. Int'l Symp. Autonomous Decentralized Systems (ISADS 99)*, March 1999, pp. 182-189.

[7] Object Management Group, "Real-time CORBA", 2000, http://www.omg.org.

[8] Object Management Group, "minimum CORBA", 2000, http://www.omg.org.

[9] Bluetooth Consortium, "Bluetooth Protocol Architecture", 1999 http://www.bluetooth.com/.

[10] The Infrared Data Association, "IrCoMM Protocol", http://www.irda.org/.

[11] HomeRF Working Group, "Shared Wireless Access Protocol", http://www.homerf.org/.

[12] F. Bennett et al, "Piconet – Embedded Mobile Networking", *IEEE Personal Communications*, October 1997, http://www.uk.research.att.com/ pen/.

[13] T. Zimmerman, "Personal Area Networks", *IBM Systems Jour.*, Vol. 35, No. 3&4, 1996, pp. 609-617.

[14] E. Royer, et al, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks", *IEEE Personal Comm.*, April 1999, pp. 46-55.

[15] S. S. Yau and F. Karim, "Hybrid Middleware for Embedded Devices in Wireless Mobile Ad-Hoc Networks", *OMG Workshop Real-time and Embedded Dist. Object Computing*, July 2000, http://www.omg.org/news/meetings/realtime/.

[16] D. Tennenhouse, "Proactive Computing", *Comm ACM*, Vol. 43, No. 5, May 2000, pp. 43-50.

[17] D. Schmidt et al, "Evaluating Policies and Mechanisms for Supposrting Embedded Real-Time Applications with CORBA 3.0", *Proc. IEEE Real-Time Technology and Applications Symposium (RTAS 00)*, June 2000, pp. 188-197.

[18] A. DeHon, "The Density Advantage Configurable Computing, *Computer*, Vol. 33, No. 4, April 2000, pp. 41-49.

[19] G. Borriello and R. Want, "Embedded Computation Meets the World Wide Web", *Comm ACM*, Vol. 43, No. 5, May 2000, pp. 59-66.