Service Oriented Pervasive Computing for Emergency Response Systems

Swaroop Kalasapur, Kumarvel Senthivel, and Mohan Kumar
Department of Computer Science and Engineering,
The University of Texas at Arlington, Arlington, Texas, USA - 76019
{kalasa, senthive, kumar}@cse.uta.edu

Abstract

Pervasive computing paradigm is characterized by the presence of a diverse variety of computing and communicating devices. The vision of pervasive computing is to enable effective support for user tasks by utilizing the devices carried by the users and those available within the infrastructure around the users. Emergency response is very critical to crisis situations. The process of emergency response can benefit greatly by the application of pervasive computing principles. In this paper, we describe a prototype designed to enhance the existing emergency response mechanism in attending to automobile accident victims. The prototype has been built using our event oriented middleware called Pervasive Information Communities Organization (PICO), and is built on top of a popular P2P framework called JXTA. By utilizing the middleware, we model the useful features of devices available within a car as services. Our proposed framework allows the creation, discovery and maintenance of continuous services such as crash detection. Further more, the framework allows the composition of complex services on occurrence of events by utilizing the available basic services with a goal to provide essential support in case of an accident. We present the design and implementation details of the prototype and snapshots of the working prototype.

1. Introduction

The presence of a wide variety of devices along with increased network connectivity provides an opportunity to develop applications that can support users in their everyday activities. Personal devices such as cellular phone, PDA, laptop computers, intelligent wrist watches, etc. can provide a wide range of additional features along with their basic functionalities. Creating customized services in such devices, make it possible to utilize the available specialized functionalities, in a cooperative manner.

In this paper, we describe a system aimed at assisting a victim in an automobile accident and provide necessary help by utilizing the techno-rich devices around the user. We employ our event oriented middleware architecture called Pervasive Information Community Organization (PICO) [3]. The constructs in PICO architecture enable us to build and operate services around the available devices. We have utilized an open source P2P framework called JXTA [6, 7], as a foundation on which we have built our PICO middleware. The uniqueness of our proposed approach is the ability to compose services in the presence of exceptions and faults in a seamless fashion by using the available resources.

2. Emergency Response Process

Emergency response systems are an important functional entity in our society. For any kind of public distress, the Emergency Medical Response (EMR) personnel are the first responders to the scene, and they provide immediate medical and psychological assistance. Some of the commercially available Personal Emergency Response Systems (PERS) enable the elderly and patients in need to call for much needed help in case of emergencies, just by the touch of a button. Some of the recent research initiatives towards assisted living environments are also geared toward monitoring the inhabitants and summoning the help of appropriate support personnel in case of emergencies. Similar solutions have been implemented to assist victims of automobile accidents [1, 2].

2.1. Motivating Scenario and current practices

Consider the following scenario. A traveler driving on a highway meets with an accident and is critically injured. Typically the driver calls for help by dialing 911, or if the driver is unconscious, other drivers on the

Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)
0-7695-2520-2/06 $20.00 © 2006 IEEE

highway usually report the accident. Once the accident report is made with the EMR, a paramedic unit is dispatched to the accident scene. When the paramedics arrive at the scene, they will typically assess the victim's condition by making the first examination, and if the victim is conscious, information such as allergies to certain types of medicines, known medical conditions such as diabetics etc are gathered. All the information is typically collected on paper based forms. Some of the advanced units already use pen based computers to capture the information in digital format. The victim is then transported to a medical facility, such as the ER in a nearby hospital. En route to the hospital, the paramedics communicate with the ER personnel in the hospital to inform them of the incident and update them about the medical condition of the victim. Once the victim reaches the hospital, the information collected by the paramedics (either on paper based forms, or in digital format) is transferred to the personnel in the ER, along with details about the treatment given by the paramedics. The ER personnel will then gather the victim's medical history by querying the medical information database. Based on the information given by the paramedics, and the prior medical history of the victim, appropriate measures are taken at the hospital to ensure victim's safety.

2.2. Possible Improvements

There are a number of additional improvements that can be brought into the process of emergency response by employing pervasive computing applications. First of all, the automobiles can be empowered with the presence of additional capabilities to monitor and detect abnormal events such as an accident, and report the event to EMS. The paramedics will be further assisted, if the medical history of the patient is known in advance, or if they can communicate with the victim, while they are driving (or flying) to the scene. Access to victim's medical history (e.g., diabetics) will assist the paramedics in preparing for emergency care. Further, the paramedics, based on the medical history of the victim, and the feedback they obtain from the scene, can identify the best suited hospital to treat the victim and can contact the hospital and make necessary arrangements to bring in the victim. All the above operations will prove invaluable in saving the victim's life, keeping in mind the critical time window available for response in such emergency situations. The medical history will again be needed by the doctors and nurses, once the victim is in the ER. The assessment made by the paramedic unit on the way to the ER, is also critical in treating the victim.

In building such solutions, first, there are some specialized structures that need to be maintained at different levels in the process. Specialized software and hardware are required by the paramedics to enable to assess the state of the victim, and prepare for viable treatment options. The paramedic's vehicle itself needs to be equipped with capable communicating devices, to allow different communication operations with the victim and with the related agencies such as the hospital. Within the hospital, computing systems need to be in place to locate and track personnel, emergency equipment and medicine. Specialized software capabilities need to be built into emergency response process within the hospitals to accept incoming communications from the paramedics to allow the dynamic operations explained above. The process of emergency response can be further enhanced by dynamically utilizing the devices within the vicinity of the victim. The devices available within the automobile, such as the voice recognition capabilities of the audio unit, the presence of a video camera, the cell phone carried by the user and the telematic and navigation unit present in the car, need to be better utilized to aid the paramedics in assessing the condition of the victim prior to their arrival at the scene.

2.3. Proposed Architecture

In this paper, we propose the use of service oriented pervasive computing architecture to support the above described mechanism. The proposed model for emergency response is built on the premise that different organizations make their services available over the network to be discovered and utilized. The different organizations also cooperate among each other in performing their day to day operations. Within organizations such as hospitals, there are a number of operational protocols in place for different operations. These protocols will then drive the dynamic combination of available services to result in successful task completion. The services of the organizations can be exposed by well defined service interfaces, similar to the web services [3].

The prototype described in the rest of the paper illustrates a part of the scenario described above. The PICO middleware itself has been implemented over the popular JXTA framework that provides the fundamental P2P infrastructure over which devices can communicate. The different devices present within a car are modeled as services through implementing software entities exporting their functionality as services to the other devices over the network. The specific scenario being illustrated within the prototype

is as follows. An automobile on a highway meets with an accident.
(i) The sensors within the automobile detect the crash and a software service built specifically to monitor the status of the vehicle will query the driver about their well being.
(ii) If the driver does not respond to the query, an emergency call is placed using the cell phone.
(iii) Information such as location, snapshot of the victim, personal details for medical history retrieval are transmitted to the ER center.
(iv)The emergency response center dispatches a paramedic unit to the crash scene.
(v) The resources within the car are utilized to participate in a dialogue with the emergency personnel giving additional information.

In supporting the above task, we employ the service oriented architecture generated through the use of PICO middleware. The features of different devices are identified and are network enabled by implementing software entities called Delegents [3]. In [9, 10], we have proposed schemes to combine the services available to meet the needs of a task at hand.

3. System Architecture

The prototype has been implemented by utilizing the event-oriented middleware called PICO [3]. The implementation of the middleware itself is based on JXTA [7], a popular P2P framework. JXTA is a protocol specification, which does not make any assumptions about the operating system, development language or network transport. The reference implementation provides support for HTTP and TCP/IP [7]. Heterogeneous devices with completely different software profiles can communicate with each other using the JXTA protocols. The reference middleware stack utilized for implementing the prototype is shown in Figure 1.

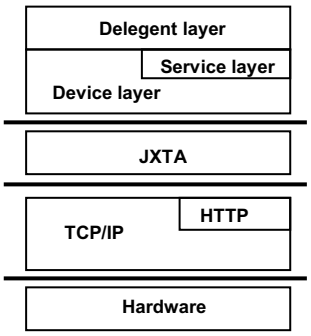

Figure 1. Reference middleware stack.

When a device layer is initiated, it instantiates a default peer group. Any other device in the local network which starts after the instantiation of the first peer group will search for the existing peer group and joins the group. After creating a new group or joining an existing group, the device layer tries to find the available rendezvous service on the local network. The device layer sends and receives the messages to and from other devices respectively, after it gets connected to them through the service layer. The received messages are processed by a message handler. The messages intended for the services are passed to the delegent layer's message handler.

The service layer helps the device layer to create and publish services. In order to create a service, it creates and publishes an advertisement for the service. It also creates a bidirectional pipe and listens for service requests. When the service layer receives a request from other peers, it accepts the connection and returns the new accepted pipe to the device layer, which waits on the pipe to receive messages.

A service or an application which accesses the service is created by implementing the interface provided by the delegent layer. When a service is started, it requests the device layer to create a service with the specified name. It also specifies the message handler to be invoked whenever it receives a message. When an application needs to find a service on the network, it calls the Service layer to look for the service.

Upon shutdown, services enter the *exit* state, which breaks the infinite execution of the state machine and requests the device layer to close the service which closes all the open pipes and requests the service layer to close the server pipe.

4. Prototype Details

A prototype for enhanced first response system is implemented by employing services developed using the PICO middleware over the reference implementations of JXTA. In the prototype, once a car crash is detected by monitoring the pressure sensor, a message is sent to the cell phone asking if he or she is ok. If the user does not respond within a specified time, it accesses a web service of the emergency service office and logs a request. The web service locates the available paramedic vehicle nearest to the location of the accident and sends the information to the cell phone, which in turn contacts the service on the paramedic vehicle to upload the user's medical records and helps in deciding the correct medicines to be administered.

4.1. Modeling the devices within the car

The devices within the car include a mote sensor that detects pressure at regular short intervals and sends the information to a computer which is capable of communicating to other mobile devices in the car. A service which monitors the readings from the pressure sensor and determines the occurrence of a crash is built and represented as the *crash detector delegent (D_{CD})*.

Figure 2 is the state diagram for the crash detector service. The ellipses are the states of the application. The text above the arrows are events and the text below the arrows are actions taken in response to the events to change from one state to another.

On initiation, the D_{CD} receives the pressure values from the sensor and determines whether a crash has occurred by comparing them to a threshold value. If a crash is detected, a discovery for a *UI* service within the vicinity is made until a predetermined time out. After timeout, D_{CD} tries to find the *Dialer* service, to contact the emergency services. If the *UI* service is found, it opens the service and waits for a confirmation message. If an error message is received, it tries till the maximum repeat count and then continues to find the *Dialer* service. If it receives success, then it sends "*Are you OK?*" message, to the *UI* service and waits for the response.

If the response is "*Yes*" or "*No*", D_{CD} continues to detect crash as the *UI* service will take over to contact the emergency services. But if there is no response until the preset time, then D_{CD} proceeds to find the *Dialer* service on its own, opens it and begins the "*Contact Emergency*" process.

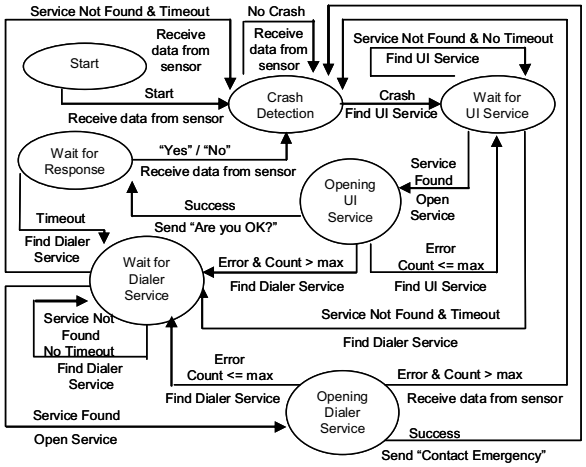

Figure 2. State Diagram for the crash detector.

The *UI* service (D_{UI}) can execute on a device with which the user can interact. In the current prototype, we have experimented with D_{UI} running on a wrist watch as well as on the cell phone emulators. The devices are Java enabled with J2ME, MIDP 2.0 profile and CLDC 1.1 configuration.

When the user fails to respond through the UI service, the emergency personal are informed through the *dialer* service. The *dialer* service is essentially a communication service, implemented on a cellular phone. The dialer delegent conveys accident information to the emergency response system through any of the available communication interfaces. The detailed state diagrams for all the delegents are not provided due to space limitations.

4.2. Modeling the Emergency Service Center

The emergency service center is an operational unit, where the incoming distress calls are handled. Within our prototype, we have implemented the emergency service center as a web service, and the designed service accepts incoming requests for emergency help via SOAP messages. In our implementation, along with the incoming request, the information about the victim, the current location of the incident is also made available. The Emergency service then locates a paramedic unit closest to the accident scene and dispatches it. Once the paramedic unit has been dispatched, it responds back to the requestor, with a message stating the dispatched information.

4.3. Modeling the devices within the ambulance

Typical paramedic units are equipped with an array of state of the art devices such as EKG monitors, defibrillators, oxygen monitors, etc. In the current implementation, the paramedic unit is also implemented as a web service, and it accepts messages from the emergency response center, and responds back with an acknowledgement indicating that the victim will be attended to in a short time.

The emulation of the user interface service on a wrist watch emulator is shown in Figure 3. Figure 4 shows snapshots from the dialer service at different time instances. The dialer service contacts the emergency service and will also display the response from the emergency service. We are currently extending the dialer service to facilitate live communication channel for the emergency personal.

Figure 3. UI Service on a wrist watch emulator.

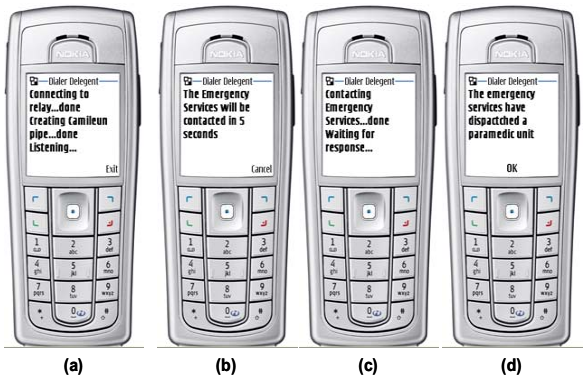

Figure 4. Different states in the dialer service.

5. Conclusions and Future work

In this paper, we have presented a prototype illustrating the use of pervasive computing principles in enhancing existing EMR systems. The described prototype is capable of continuously monitoring an automobile, and in the event of an accident, it uses the distributed capabilities of different devices present within the automobile to call for emergency help. We have also described extensions that would enable the emergency response process by creating a medium of communication between the EMR team and the accident scene, by utilizing the services available around the victim.

We are currently extending the prototype to include a possible report of the accident scene to the paramedics, automatic medical information retrieval based on the information stored in driver's personal devices. Techniques used to include the live reporting between the victim's car and the ambulance can directly be used to relay information from the paramedic's vehicle and the hospital.

We are also investigating the use of semantic information to empower the designed applications in obtaining timely support from services beyond the immediate vicinity of the user.

ACKNOWLEDGEMENTS: The work presented in this paper was completed under funding from the National Science Foundation Grant NSF-STI 0129682. We would like to thank Dr. Behrooz Shirazi, Dept of EECS, Washington State University, and Mr. Byung Sung, Dept of CSE, The University of Texas at Arlington for the valuable technical discussions and ideas. We also like to thank Mr. Ed McGinley, a paramedic by profession and MedStar, Fortworth for detailed insight into the practices and facilities in EMS.

6. References

[1] Thompson. C. "Everything is alive," Internet Computing, IEEE, Volume 8, Issue 1, Jan-Feb 2004 Page(s):83 – 86.

[2] Bretz. E.A. "The car: Just a web browser with tires," Spectrum, IEEE Volume 38, Issue 1, Jan. 2001 Page(s):92 – 94.

[3] Papazoglou, M. P. and Georgakopoulos, D. "Service-Oriented Computing –Introduction," Communications of the ACM, Vol 46, Issue 10 (Oct. 2003), pp 24-28.

[4] M. Kumar, B. Shirazi, S.K. Das, M. Singhal, B.Y. Sung, and D. Levine, PICO: A Middleware framework for Pervasive Computing, IEEE Pervasive Computing, volume 2, No 3, page 72-79

[5] Satyanarayanan. M. "Pervasive Computing – Vision and Challenges," IEEE Personal Communications, Aug 2001, Vol 8, Issue 4, page(s) 10 -17.

[6] L. Gong, "JXTA: a network programming environment," Internet Computing, IEEE, May-June 2001, Vol 5, Issue 3, page(s) 88-95.

[7] Web Resource "JXTA v2.0 Protocols Specification," http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.html

[8] Nokia 6230i Emulator, "The Series 40 Developer Platform: Introductory White Paper," http://sw.nokia.com/id/07dfcd93-4010-4ab9-8feb-c50c2799f154/Series_40_DP_White_Paper_v1_1_en.pdf

[9] S. Kalasapur, M. Kumar, and B.A. Shirazi. "Personalized service composition for ubiquitous multimedia delivery," In WoWMoM 2005. Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, 2005, pages 258 - 263, 2005.

[10] S. Kalasapur, M. Kumar, B. Shirazi. "Seamless Service Composition (SeSCo) in pervasive environments," to appear in The first international workshop on Multimedia Service composition, to be held in conjunction with ACM Multimedia 2005.