PerSON: A Framework for Service Overlay Network in Pervasive Environments

Kumaravel Senthivel[1], Swaroop Kalasapur[2] and Mohan Kumar[3]

[1] Fujitsu Network Communications, Richardson, Texas, USA - 75082
[2] Samsung Information Systems America, San Jose, California, USA - 94086
[3] Department of Computer Science and Engineering,
The University of Texas at Arlington, Arlington, Texas, USA - 76019
kumar.senthivel@us.fujitsu.com, s.kalasapur@samsung.com, kumar@cse.uta.edu

Abstract. With the increase in the number of mobile devices and their network capabilities, users expect transparent access to available services in their pervasive environment. However, heterogeneity and interoperability issues persist in existing mechanisms. In this paper, we present a lightweight framework, PerSON (Service Overlay Network for Pervasive Environments), to abstract the details of service provision and utilization in a pervasive environment. PerSON constructs an ad hoc service overlay network in the pervasive environment based on service requests. PerSON achieves high efficiency by combining service discovery with route discovery. PerSON is especially suitable for resource constrained devices as minimal information about discovered services and the neighboring devices is maintained. In particular, the proposed framework provides the overlay network for the earlier developed middleware for pervasive computing. We also describe the implementation of a prototype for emergency response system (ERS).

1. Introduction and Motivation

In 1988, Mark Weiser envisioned that the computing devices will recede into the background in pervasive computing environments [1]. Even though there has been tremendous progress in many related areas, research is still being done to address many challenges [2]. The most common challenges to creating a pervasive environment are service provisioning and utilization. In a pervasive environment, there are a variety of devices such as desktops, laptops, PDAs, cell phone, etc. These devices may be connected to different physical and logical networks like the Wireless LAN, Bluetooth, etc. The services and applications hosted on different devices should be able to discover and communicate with each other over heterogeneous networks.

In keeping with Mark Wesier's vision PerSON attempts to address two important objectives of pervasive environments: i) A service developer's focus should be on the details of what to provide in the service rather than how to provide the service; and ii) An application developer's interest in the services that can be utilized by the application rather than how to access them.

In this paper, we present a simple framework called PerSON (Service Overlay Network for Pervasive Environments) for constructing a service overlay network. PerSON abstracts the complexity of the underlying heterogeneous networks and provides a simple application interface to the user for developing services and applications. We employ distributed query based architecture [3] [4] to discover services and dynamic source routing (DSR) [5] to discover routes across multiple networks. Service discovery is optimized by piggybacking the route messages in the service discovery messages. We also describe the implementation of a prototype for an Emergency Response System (ERS) [6] using the Pervasive Information Community Organization (PICO) [7] architecture.

2. Related Work and Limitations

Over the last few years, significant research has been done in the field of service provision and utilization. Almost all existing frameworks construct a service overlay network to provide additional services over the underlying network. Service Overlay Network (SON), proposed by the authors of [8], enhances the best-effort services in the Internet by providing QoS. An application overlay network, created in [9], detects and recovers from path outages and periods of degraded performance in the Internet within seconds. The peer-to-peer overlay networks like Kaaza and Gnutella provide content sharing. Content Addressable Network (CAN) [10], Chord [11] and Pastry [12] organize the overlay network based on the content attributes to optimize the search.

Recent research initiatives have led to the development of Konark [13] and JXTA [14] [15] that provide support for generic services in a dynamic pervasive environment. Though these frameworks have their own advantages, there are certain limitations when executed in a pervasive environment. XML is used for service description and other data messages in both Konark and JXTA. XML is flexible but not suitable for resource constrained devices. High processing power and huge memory is required to parse the XML messages. JXTA attempts to provide support for such devices by introducing a new version called JXTA for Java 2 Micro Edition (J2ME) or JXME. Konark is dependent on IP multicasting. The routing mechanisms provided by the underlying network are used to route messages. Bridging between different physical or logical networks is not supported. The services in one network cannot be accessed by the devices in a different network. Though JXTA supports different message transport bindings like TCP, HTTP and TLS, it requires special designated devices to discover routes and forward messages.

3. Architecture of PerSON

The services and applications developed using PerSON may be hosted on devices with different computing capabilities. The devices may be connected to different physical or logical networks. A service overlay network is created by the framework to ensure availability, accessibility and utility of services. The services in PerSON can

be discovered and utilized by external services and applications, regardless of the topology and complexity of the underlying network.

Suppose three devices (Fig. 1) are connected in two different physical networks. For example, the cell phone and the laptop are connected using RFCOMM in the Bluetooth network, whereas the laptop and the PDA are connected using TCP/IP in wireless LAN. Generally, a service available on the PDA cannot be accessed by an application on the cell phone. If the application and service are developed using PerSON, then they are connected by the service overlay network. The two different networks are bridged by the laptop using PerSON. The underlying network complexity is hidden by the overlay network and service discovery and connectivity are abstracted from applications and services.

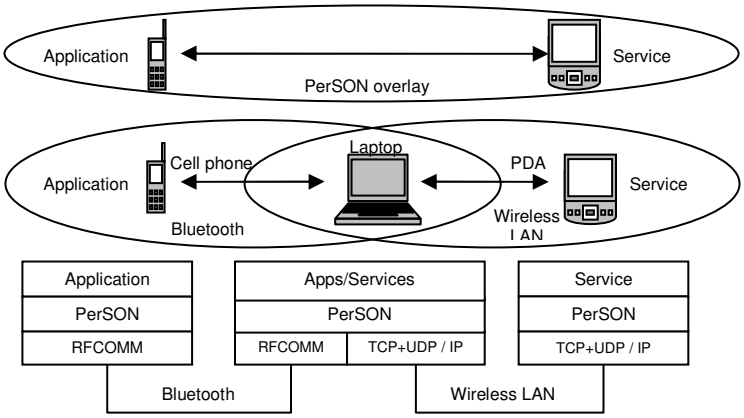


Fig. 1. PerSON Architecture

3.1 PerSON stack

The PerSON stack (Fig. 2) is implemented by every device that participates in the overlay network. PerSON stack comprises three layers – network, device and service.

3.1.1 Network layer

The network layer abstracts the various networks that connect the devices. A device is identified by a unique device identifier (DID). The addresses of the physical networks are masked by the DID. The network connections are made using the available transport on each physical network. The unicast and broadcast functions of the network layer are utilized by the device layer to exchange messages with the neighboring devices.

3.1.2 Device layer

The functions provided by the network layer are utilized by the device layer. The details of service creation, discovery and utilization are abstracted from the services by the device layer. The device component is central to the PerSON architecture. The

incoming messages are received by the device and delivered to the service connections, the resolver or the discoverer based on the message type. Whenever a message is received, the device table is updated with the physical address of the neighboring device from which the message is received. The device is also responsible for choosing the right network connection to send the outgoing messages. The services hosted by a device are registered in the local services. Each service is identified by a unique service identifier (SID). A new service connection is spawned when the device receives a service request. The connection is utilized by the service layer to communicate with the application that requested the service.

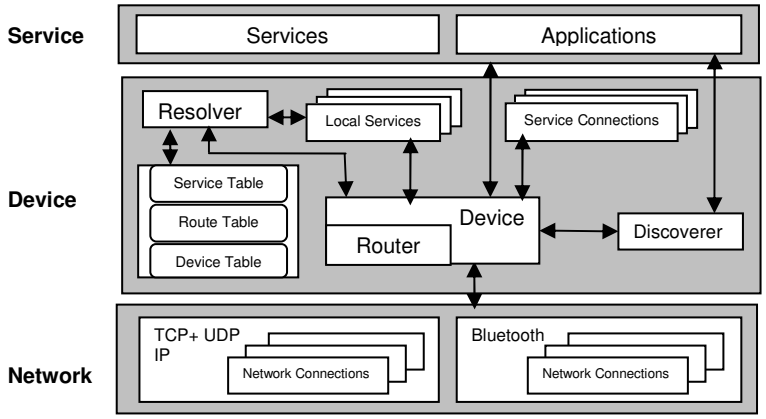


Fig. 2. PerSON Stack

The discoverer is responsible for finding the required services. When an application or service requests the discoverer to find a service, a query message is broadcast to other devices. The query contains the service description in simple text. The discoverer can be overridden to support user defined complex descriptions like XML format description. The scope of the discovery is restricted in terms of hop count.

The received query messages are processed by the resolver. If a matching service is found in the local registry then a result message is sent back by the resolver to the device that requested the service. The result messages received from other devices are also processed by the resolver. The SID of the requested service and the complete route to the device that hosts the service are contained in the result message. The service information is updated in the service table and the route information is updated in the route table. If the discoverer is overridden, then the resolver is also overridden to analyze the query message and search for service description.

The router is only used when the device is connected to multiple networks and the device is willing to act as a bridge between those networks. The router is not required for a device with only one network interface. When a message received is not intended for the device, the router forwards it to the next hop in the route.

3.1.3 Service layer

The user defined services and applications are included in the service layer. The functions provided by the device layer are utilized to create new services. The applications in the service layer request the device layer to discover other services and connect to the required services. Once a service connection is established by the device layer, an application and a service can communicate using the connection.

3.2 Physical network connections

In a pervasive environment, a device may be connected to more than one network. For example, a device may be connected to two IP networks supporting TCP and UDP transports and to one Bluetooth network using RFCOMM connections. The functionalities of the underlying network are utilized to broadcast query messages and unicast other data messages. IP and Bluetooth networks are supported by the current implementation of PerSON.

In IP networks, the query messages are broadcast using UDP transport. The device responding to the query, first establishes a TCP connection. In Bluetooth networks, when a device enters (or turned on) the network, the neighboring devices are discovered using the Bluetooth discovery mechanism. If a PerSON service is found, then a RFCOMM connection is established to those neighbors. Devices that can alternate between the master and slave modes (of Bluetooth) listen for incoming connections and also connect to other devices. Multiple connections with different devices are possible for such devices. Devices that are not capable of alternating between master and slave modes can have only one connection. Query messages are flooded using multiple unicasts.

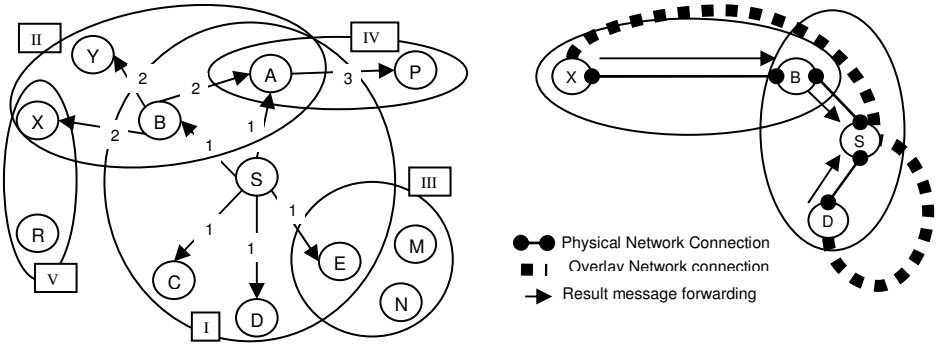
3.3 Overlay network

The overlay network in PerSON is constructed to facilitate service provisioning and utilization. By combining the service discovery with route discovery, high efficiency is achieved. The route information is piggybacked in the service discovery messages, so that additional route discovery messages are not required to discover the route between the service provider and the service consumer.

Distributed query architecture is used to propagate the service discovery queries. In a pervasive environment where only local scalability is considered the flooding mechanism, restricted by the scope is a simple method for service discovery. The devices do not depend on specific service directories to discover the available services. Each device broadcasts a query for the required service in its local network. All other devices in the local network process the query message, but only those devices that support routing to bridge different physical or logical networks will forward the queries.

Consider the network in Fig. 3(a) that comprises five different sub networks. Devices S, A, B, C, D and E belong to network I. Devices A, B, X, and Y belong to network II. Devices E, M, and N belong to network III. Device A and P belong to the network IV. Devices X and R belong to the network V. Only devices A, B and E have

two different network interfaces and belong to two different networks. The device S sends a query for a specific service with a scope of 2. The query is broadcast in the local network of device S. The devices A, B, C, D and E in this local network process the query. But only those devices that are willing to route the messages will propagate the query to other networks. For example, B propagates the query and acts as a bridge between networks I and II. E cannot propagate the query because it does not support routing. Duplicate queries are not propagated. Hence device A propagates the query to network IV only once, even if it receives the same query from devices S and B.



(a) (b)

Fig. 3. (a) Query propagation in PerSON, (b) Overlay network in PerSON

Whenever a query message (Fig. 4 (b)) is broadcast in a network, the DID of the device that propagates the query is added to the route information in the message. The device also specifies the physical network address through which other devices can be connected. The time for which the device can be expected to be available is also added to the message. When device B receives the query message from device S, it knows how to connect to device S if needed and similarly device X knows how to connect to device B. This information is cached in the device table (Fig. 4 (i)) and cleared from cache after the specified time. Each record in the device table contains the DID, available time and the physical address of the neighboring device.

Device X also knows the complete route to device S by reversing the route in the query message. The route information for device S is cached in the route table (Fig. 4 (j)), which is cleared when all the connections to the next hop are broken. The route information can also be cleared when any other device in the route sends a negative acknowledgement specifying that the route is longer valid. Each record in the route table contains the DID of the destination device and the list of all intermediate devices. A device may cache many routes to another device based on the route information specified in the messages it receives.

Suppose the required service is provided by device X, its result message (Fig. 4 (c)) to device S is sent via the resolver of device X. The resolver specifies the complete route in the message. The network layer chooses the first entry in the device table and tries to connect to the device if there are no previous connections. If the connection is successful, then the message is sent, else the network layer tries to connect using the next entry in the device table. If the message cannot be sent an error

message is returned to the device layer. All routes that include this device are cleared from cache. The device layer retries to send the message with an alternate route. The message is dropped if there are no more routes to the destination device. When a device such as B receives the message (sent from X to S), it simply routes the message to S via the router in the network layer.

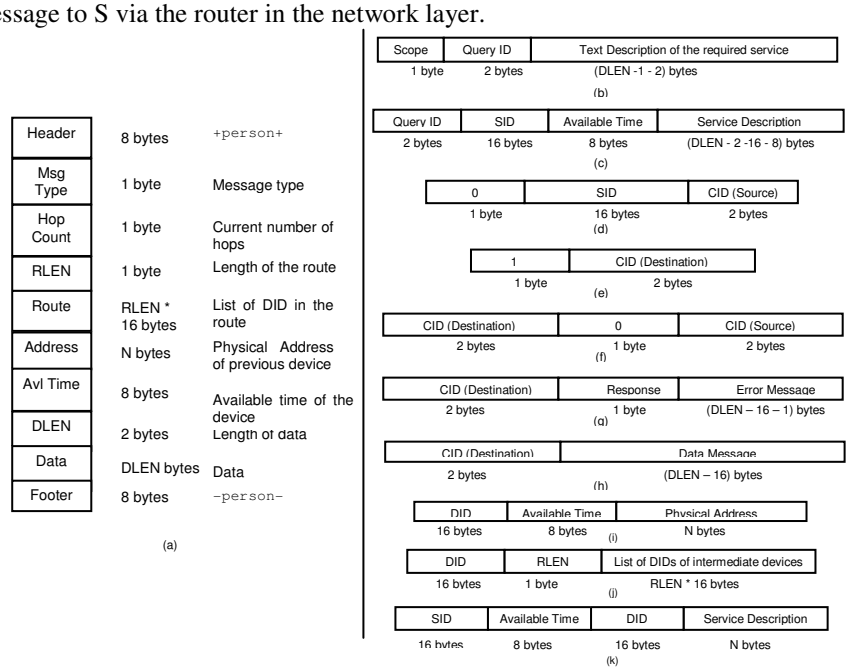


Fig. 4. (a) Message Format, (b) Query, (c) Result, (d) Connect Request, (e) Close Request, (f) Success Response, (g) Error, (h) Data, (i) Device table, (j) Route Table (k) Service Table

Suppose device D also provides the requested service then it sends the result message to device S. The overlay network (Fig. 3(b)) is formed only between the devices X and S and between devices D and S. A device that provides a service gets connected to a device that consumes the service. When the device S receives the result message, it caches the service information in the service table (Fig. 4 (k)) and the reversed route in the route table. Each record in the service table contains the SID, service available time, the DID of the device which provides the service and the service description.

An application on device S requests the device layer to connect to the required service by specifying the SID, The device layer retrieves the DID of the device X that provides the service from the service table. The first available route to device X is obtained from the route table. The device layer then requests the network layer to send the message to the DID of the next hop (B). Devices S and X exchange messages (Fig. 4), through device B, by using the connections previously established during the service discovery process.

4. Prototype Implementation

The PerSON framework is not dependent on a specific development language or operating system. The reference implementation of PerSON is developed using Java. The J2SE version of PerSON can be executed in powerful devices like desktops and laptops. The J2ME version can be executed in resource constrained devices like PDAs and cell phones. The J2SE version supports both TCP/IP and Bluetooth networks whereas the J2ME version supports only Bluetooth networks. The reference implementation provides simple Java APIs to create, discover and utilize services.

The implementation of PerSON is used to provide the overlay network for the PICO middleware for pervasive computing. The main components of PICO are *devices*, and intelligent agents called *delegents.* PICO creates mission oriented dynamic and static *communities* of delegents that perform tasks for the users and devices.

4.1 PICO implementation using PerSON

The device layer and the service layer of PerSON integrate with the device layer and the delegent layer of PICO middleware. When a device is started, the different networks supported by PerSON are initiated. The TCP/IP networks open a UDP server socket and wait for incoming query messages. The Bluetooth networks explore other devices in the vicinity and connect to the devices that implement PerSON. Additional functionalities like the definition of system characteristics are added by the device layer of the PICO framework. The delegent layer of the PICO middleware is similar to the service layer of the PerSON framework.

4.2 Emergency Response System

A prototype for enhanced Emergency Response System (ERS) is implemented by employing services developed using the PICO middleware framework. In the scenario for an ERS, when an automobile on a highway meets with an accident, a crash detector application running on the automobile's computer detects the crash through various sensors. The user may or may not require immediate help based on the intensity of the crash. In order to avoid false alarms the application has to confirm whether the user needs any help. So, it communicates with a UI Service that can inquire the user and confirm the crash. The UI service may be located on the user's watch or PDA or any device that can display the message and get the user's attention. If the user's response is negative or if the user does not respond, the computer assumes that a real crash has occurred. Now the application finds a Dialer Service that can request help from an emergency response center. The dialer service may be located on the user's cell phone or any communication device that can access the Internet. The crash detector application requests the dialer service to place an emergency call with information about the location of the incident and user's personal details for retrieving the available medical history. The emergency response center dispatches an emergency medical services (EMS) vehicle nearest to the location.

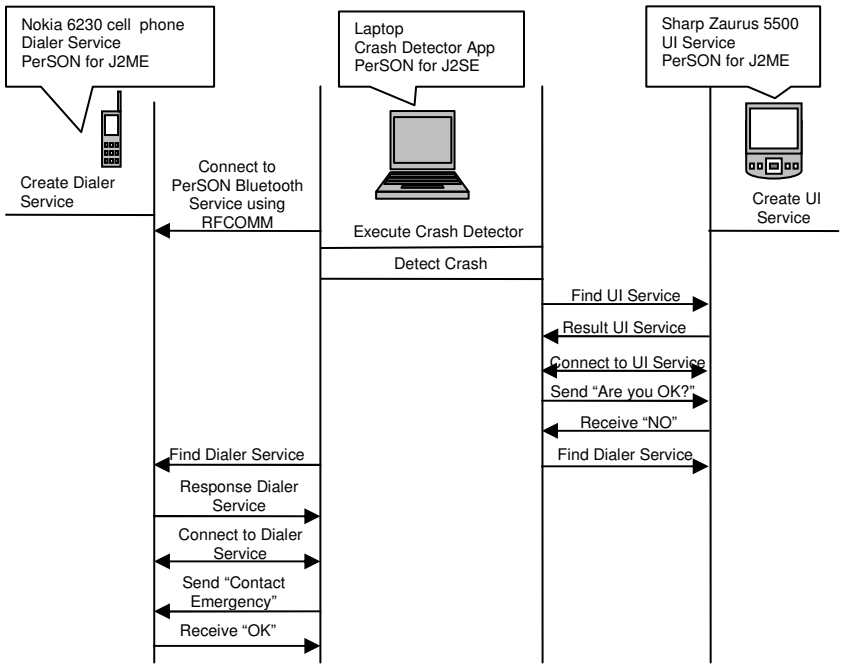


Fig. 5. Execution of ERS services using PerSON

In the prototype for the ERS, the crash detector application is executed on a Fujitsu laptop, equipped with a DBT120 Bluetooth dongle driven by Microsoft Bluetooth stack. The UI service is executed on a Sharp Zaurus 5500 PDA with a compact flash wireless LAN card. The Dialer service is executed on the Nokia 6230 cell phone. A light sensor is used to simulate the crash. The laptop and PDA communicate using an ad hoc wireless LAN. The cell phone and the laptop communicate using Bluetooth network.

The services and the application are initialized and executed as shown in Fig. 5. The PerSON framework is utilized by the PICO middleware to initialize the Bluetooth network in the cell phone and the laptop. A Bluetooth service of PerSON is created on the cell phone and the laptop. When the Bluetooth network is initiated in the laptop the Bluetooth discovery mechanism discovers the cell phone. The laptop is connected to the "PerSON" service on the cell phone using RFCOMM connection. The dialer service and UI service are created and registered only in the devices in which they are hosted. When the crash detector detects a crash, a query message is broadcast using UDP in the IP network and unicast to the cell phone using the RFCOMM connection. The query message is received by PerSON on PDA and a TCP connection is created between the laptop and the PDA. The response message for the UI service is unicast over the TCP connection. A service connection is established between the crash detector application and the UI service. The data messages are exchanged over this service connection. Similarly, the dialer service is discovered and utilized by the crash detector application.

5. Features of PerSON

5.1 Light-weight Framework

In PerSON, the directory of available services in the network is distributed. Each device stores only the information about the provided services. Other devices cache the information for the discovered services and the complete route to those services. Since there is no central repository, PerSON requires less memory to store the minimal service information, compared to the service-coordinator based approaches. PerSON uses simple text to describe services. The memory required to cache the discovered services is considerably reduced. A typical record in the service table requires 16 bytes for the service identifier, 8 bytes to specify the available time, 16 bytes the device identifier and say, 256 bytes for the service description. A total of 296 bytes is required to cache the information of a service. A typical record in the route table for a route with 3 hops requires 16 bytes for the destination device identifier, 1 byte for the length of the route and 32 bytes for the route. A total of 47 bytes is required to cache the route. A record in the device table that stores the information about another device on an IP v4 network requires 16 bytes for the device identifier, 8 bytes for the available time and 6 bytes for the IP address and the port number. A total of 30 bytes is required to cache the device information.

5.2 Heterogeneous Network Connectivity

The reference implementation of PerSON is capable of bridging IP networks and Bluetooth networks. Devices connected to only Bluetooth network can communicate with devices connected only to an IP network using any device that is connected to both networks. The bridging is done at the application layer above the TCP/IP stack and the Bluetooth stack. A device willing to route the messages should include the router component of PerSON stack. The router component simply forwards the message to the next hop in the route.

5.3 Dynamic Service Discovery and Routing

In order to support heterogeneous networks, the messages have to be routed in the overlay network. Dynamic source routing protocol is used in the overlay network for routing messages in the ad hoc environment. The route discovery is merged with service discovery. The route information is piggybacked in the service discovery query messages. A device receiving the query message knows the route to the device that requires the service. Similarly a device receiving the response message knows the route to the service provider. The service and route information are cached in each device. The complete route is specified in each message. The route includes only the list device identifiers of intermediate devices. Each intermediate device is responsible for connecting to the next hop in the route using the physical network connections.

Table 1. Comparison of PerSON, JXTA, and Konark

Feature	PerSON	JXTA	Konark
Support for resource constrained devices	Yes. Uses binary messages and is light-weight	Partial support. Uses XML messages for JXTA and binary messages for JXME	No support. Uses XML messages
Support for heterogeneous network	Supports TCP/IP and Bluetooth networks	Current implementation supports only TCP/IP networks. Supports different message transport binding	No support. Depends on IP multicasting
Support for multiple network interfaces	Yes.	Yes	No
Support for dynamic networks	Yes	Yes	Yes
Service discovery	Highly decentralized Only reactive.	Uses distributed service indices. Reactive and proactive	Highly decentralized Reactive and proactive
Service description	Simple Text	XML	XML
Support for service composition	No	No	No
Platform Independence	Yes	Yes	Yes
Scalability	Locally scalable	Scalable over Internet	Depends on the scalability of IP multicasting

6. Conclusion

PerSON is a framework for constructing service overlay network to facilitate interoperability in pervasive environments. It provides a light-weight abstraction of the complexities of the underlying heterogeneous network to the applications and services. A reference implementation of the framework is developed and used to provide the overlay network for the pervasive computing. The prototype is implemented using the services for an enhanced emergency response system. Future work includes support for service composition and security. Authentication and authorization will be incorporated in future versions of PerSON. The framework will also be evaluated for scalability and throughput.

Acknowledgements: The material presented in this paper is based on the work supported by the National Science Foundation Research Grants STI -0129682 and IIS-0326505.

References

1. Weiser, M.: The Computer for the 21st Century. Scientific American, 265(3). (1991) 94-104.
2. Satyanarayanan, M.: Pervasive Computing: Vision and Challenges. IEEE Personal Communications, 8(4). (2001) 10-17.
3. Perkins, C.E., & Koodli, R: Service discovery in on-demand ad hoc networks. IETF Internet Draft (work in progress). (2002)
4. Engelstad, P.E. & Zheng, Y.: Evaluation of Service Discovery Architectures for Mobile Ad Hoc Networks. Wireless On-demand Network Systems and Services. Proceedings of the Second Annual Conference on. (2005) 2-15.
5. Perkins, C.E., Royer, E.M., Das, S.R., & Marina, M.K.: Performance comparison of two on-demand routing protocols for ad hoc networks, IEEE Personal Communications, 8(1). (2001) 16-28.
6. Kalasapur, S., Senthivel, K. & Kumar, M.: Service Oriented Pervasive Computing for Emergency Response Systems. Pervasive Computing and Communications Workshops 2006. Ubicare'06. Proceedings of the Fourth Annual IEEE International Conference on. (2006) 517 – 521.
7. Kumar, M., Shirazi, B., Das, S.K., Singhal, M., Sung, B.Y., and Levine, D.: PICO: A Middleware framework for Pervasive Computing. IEEE Pervasive Computing, 2(3), (2003) 72-79.
8. Duan, Z., Zhang, Z.L. & Hou, Y.T.: Service overlay networks: SLAs, QoS and bandwidth provisioning. Network Protocols. Proceedings of the 10th IEEE International Conference on. (2002) 334-343.
9. Anderson, D., Balakrishnan, H., Kaashoek, M., & Morris, R.: Resilient Overlay Network. Proceedings of ACM Symp on Operating Systems Principles. (2001)
10. Ratnasamy, S., Francis, P., Handley, M., Karp, R. & Shenker, S.: A scalable content-addressable network. Proceedings of ACM SIGCOMM. (2001) 161-172.
11. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F. & Balakrishnan, H.: Chord: a scalable peer-to-peer lookup protocol for Internet applications. Networking. IEEE/ACM Transactions on, 11(1). (2003) 17 – 32.
12. Rowstron, A. & Druschel, P. Pastry: Scalable, Distributed, Object Location and Routing for Large-Scale Peer-to-Peer Systems. IFIP/ACM International Conference. Distributed System Platforms (Middleware). (2001) 329–350.
13. Choonhwa, L., Helal, A., Desai, N., Verma, V. & Arslan, B.: Konark: A system and protocols for device independent, peer-to-peer discovery and delivery of mobile services. Systems, Man and Cybernetics. IEEE Transactions on, 33(6). (2003) 682-696.
14. Li Gong: JXTA: a network programming environment. IEEE Internet Computing, 5(3). (2001) 88-95.
15. Traversat, B., Arora, A., Abdelaziz, M., Duigou, M., Haywood, C., Hugly, J., Pouyoul, E. & Yeager, B.: Project JXTA 2.0 Super-Peer Virtual Network. http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf. (2003)