# QoS Middleware Support for Pervasive Computing Applications

Behrooz Shirazi, Mohan Kumar, Byung Y. Sung
*University of Texas at Arlington*
*Department of Computer Science and Engineering*
*Box 19015 Arlington, TX 76019-0015*
*{shirazi|kumar|sung}@cse.uta.edu*

## Abstract

*Today, pervasive computing technologies are being developed to provide automated, real-time, continual, and unobtrusive user services in dynamic heterogeneous environments such as telemedicine, manufacturing, space endeavors, crisis management, and military. However, the full potential of pervasive computing cannot be realized without enabling middleware technologies – henceforth referred to as "middleware services"; i.e., they provide services for high level applications. In this paper we introduce middleware services that are required to address the challenges related to adapting to dynamically changing situations, meeting communication QoS requirements, and achieving scalability in large-scale pervasive computing applications over heterogeneous network infrastructures. Our approach in developing middleware support for pervasive computing is based on the community computing concept, which provides us with a unified method to dynamically integrate the middleware services with each other and with high level applications on a just-in-time basis. The proposed middleware services for pervasive computing are highly modular, lightweight, and easily deployable to meet the needs of pervasive computing applications in networked environments. In this paper we will describe our proposed QoS algorithms and techniques for synergistic integration of middleware services into pervasive computing applications. In addition, we propose new performance metrics and a benchmark suite approach to evaluate middleware services in pervasive computing. Finally, we present some early prototype results from our proof of concept implementation.*

## 1. Introduction and Motivation

Over the past few years pervasive, or ubiquitous, computing and communication has evolved into a mature computer science and information technology field. Mark Weiser, who is widely known as the original architect of ubiquitous computing, defines the discipline as the creation of environments saturated with computing and communication capability, yet gracefully integrated with human users [1]. He envisioned an environment in which computing is put in the background and the users are unaware of the existence of computers. Today, there are several major pervasive computing projects at universities [2, 3, 4, 5, 6, 7, 8, 9] and industries [10, 11, 12, 13]. Interested readers are referred to [7, 14, 15] and the IEEE Pervasive Computing journal for further details.

Today, pervasive computing technologies and the associated software are being developed to facilitate such applications as telemedicine, education, space endeavors, marketing, crisis management, transportation, manufacturing, and military for *all the time* and *everywhere* use. These applications demand automated, continual, and unobtrusive services and proactive real-time collaborations among devices, software agents, and geographically distributed personnel in dynamic heterogeneous environments. It is an extremely challenging task to provide *scalable*, *efficient*, and *seamless* pervasive computing services in dynamically changing environments [15]. In this paper we describe middleware services that will facilitate implementation of pervasive computing applications in dynamic and complex environments such as those encountered in telemedicine, space endeavors, or the military. This includes Quality of Service (QoS) and Resource Management (RM) middleware services to guarantee end-to-end soft real-time communication on heterogeneous intranet and the Internet infrastructures. We describe scalable, efficient (*just-in-time*) techniques for the synergistic integration of middleware services into pervasive computing applications. We also propose new performance metrics and techniques for a benchmark suite to measure the effectiveness of the proposed middleware services. Finally, we present the results from our early proof of concept implementation of the proposed middleware services.

Through a project[1], called PICO (Pervasive Information Community Organization), we are designing and developing new architectural models for Internet-based pervasive computing applications [5, 6]. With assistance and consultation of researchers at the University of Texas Southwestern Medical Center, we are also developing a PICO test-bed prototype environment for telemedicine applications. A PICO application consists of a set of embedded hardware devices and sensors, called camileuns (context-aware, mobile, intelligent, learned, ubiquitous nodes), and their associated software agents, called delegents (intelligent delegates). Delegents perform goal-oriented tasks on behalf of their associated camileuns. For example, a delegent associated with a heart monitor camileun will continuously analyze, record, and (when needed) transmit heart health information. A major contribution of the PICO project is the introduction of a novel concept, called *community computing*, and using it as a framework for collaboration among delegents. In community computing, delegents working on behalf of camileuns, collaborate with each other to carry out application-specific services. Pervasive computing challenges being addressed in the PICO project include design and development of: i) PICO architecture and its building block elements, ii) computing community operations, iii) collaboration and/or communication protocols among delegents and communities, and iv) modeling, analysis, simulation, and prototyping of the PICO environment using a telemedicine application.

The PICO project relies on existing and primitive middleware services to support its operations. While such an approach is sufficient for experimentation and for proving the PICO concept, it is obviously insufficient in meeting real-world challenges. For example, in future PICO-enabled (pervasive computing) environments, millions (perhaps even billions) of hardware sensors/ devices and their associated software agents need to collaborate (communicate) with each other, over the Internet or future networking infrastructures, to provide dynamic services to millions of users. In such an environment the QoS requirements of varied pervasive computing applications (e.g., entertainment vs. emergency telemedicine) must be met efficiently and seamlessly. *The goal of this paper is to present novel middleware services that are required to address the challenges related to adapting to changing situations, meeting efficiency requirements, and achieving scalability in large-scale pervasive computing applications. More specifically, we address QoS and RM services that address scalability, efficiency, and end-to-end soft real-time communication on heterogeneous intranet and the Internet infrastructures*. These services are integrated into the PICO pervasive computing environment and tested using a telemedicine application.

[1] Supported in part by NSF Award STI-0129682.

## 2. Middleware in Pervasive Computing

The PICO pervasive computing project is an ideal framework for studying problems in dynamic complex environments due to its automated methods for creation of mission-oriented communities of collaborative software agents to provide services to users. At the low, physical layer, PICO camileuns are deployed to interact with the environment, collect sensory data, and communicate with other camileuns. Any physical device that possesses a CPU, memory, and communication ability can serve as a camileun. Therefore, smart dust [16], a UC Berkeley network sensor platform [17], a PDA, a cell phone, a laptop computer, and a high-end multiprocessor computer all may serve as camileuns in a PICO environment. Camileuns interact using any available networking infrastructure ranging from ad hoc networks to Personal Area Networks (PANs) [18] to structured intranets to the Internet in general. At a higher level in PICO, there is a software agent, or delegent, associated with each camileun (or group of camileuns). A delegent may be created by the programmer, user, application, or another delegent. Delegents perform mission-oriented services on behalf of their camileuns. The delegents for camileuns with limited hardware resources reside on the network (other camileuns) and perform their tasks on behalf of their camileuns. PICO introduces community computing as a framework for delegents to collaborate with each other and to carry out application-specific services. We will demonstrate this concept through an example scenario.

Consider a heart patient with a heart monitor and a cell phone (camileuns). As depicted in Figure 1, each camileun also has an associated delegent (represented by the winged icons). If the patient begins to have a heart attack, this condition is detected by the heart monitor delegent, which will then form a community with the cell phone delegent, using patient's PAN, as depicted in Figure 2. The goal of this community is to call for an ambulance. Therfore, the cell phone delegent, using the cell phone as a data communication device, requests assistance of an

**Figure 1: Heart patient**

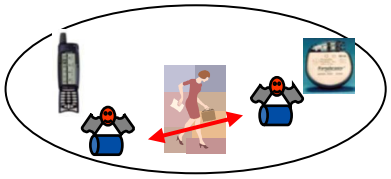

**Figure 2: Heart patient community.**

ambulance. As shown in Figure 3, even before the ambulance has arrived, a larger community of delegents consisting of the heart monitor, cell phone, and ambulance is formed to provide valuable heart information to the participating paramedics. The communication within the ambulance-patient community (Figure 3) takes place over the Internet using a mixture of wireless and land-based networks. In the PICO project we have developed a formal framework to model and analyze delegents' behavior and a rule-based, goal-driven inference engine methodology to implement the delegents and guide

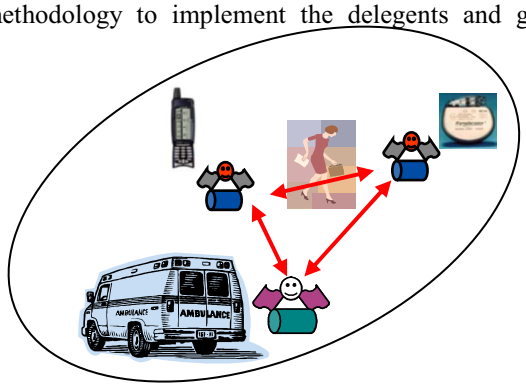


**Figure 3: Ambulance-patient community**

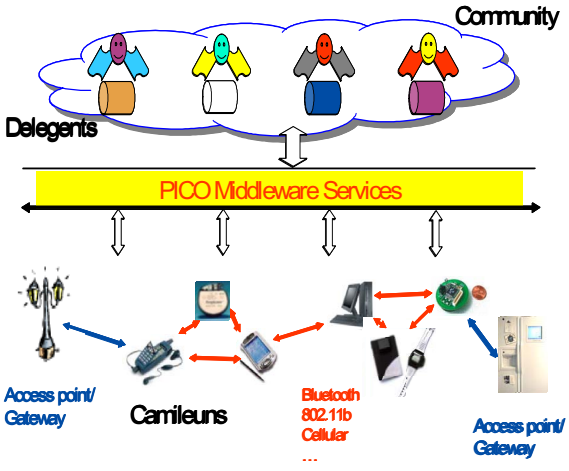


**Fig. 4 PICO Architecture**

community formation and dismantling [19]. From this simple scenario, one can observe that:

- Delegents are always active, performing services on behalf of users.
- Services provided by the delegents are automatic and in the background, often without the user's knowledge.
- In response to dynamic events, delegents automatically form communities to perform services on an as-needed-basis. This provides a great deal of adaptability to dynamically changing environments.
- The relationship among the delegents, or the composition of the communities, is not predefined. *Therefore, the delegents heavily rely on middleware services to carry out their tasks*.

Figure 4 depicts the overall PICO architecture. At the low level, camileuns collect environmental information and perform user services. They communicate using a variety of technologies such as 802.11, bluetooth, cellular, and wired communication. At the high level, the delegents carry out monitoring, event detection, and service provision activities. In response to external events or user's needs, the delegents form communities to provide higher level user services.

The delegents are designed to respond to unforeseen events and may form (or join) communities with different delegents, depending on different situations and environmental conditions. Therefore, delegents and communities heavily rely on middleware services, such as service discovery, location awareness, migration and mobility, transcoding, and communication, to dynamically identify and collaborate with other delegents. Additionally, QoS and RM middleware services are needed to address challenges arising from scalability (large number of communicating devices and delegents), efficiency (better bandwidth utilization), and meeting soft real-time deadlines (particularly for high priority, time-sensitive applications such as telemedicine).

## 3. Background and Challenges

In this paper we focus on issues related to middleware services for communication in pervasive computing. Our work on middleware services for service discovery, location awareness, migration and mobility, and transcoding will be reported separately.

### 3.1 QoS and RM middleware services

In this section, QoS and RM middleware services for bandwidth and end-to-end soft real-time communication guarantees in pervasive computing will be discussed. Performance-related middleware services such as QoS and RM for meeting bandwidth and end-

to-end communication delays are critical to pervasive computing applications. Timely collaboration among software agents and hardware devices in dynamic situations is crucial for time-critical applications. For example, provisioning QoS for on-the-fly video communication between an ambulance and the hospital is not possible with the existing best-effort-based Internet services [20]. Also, architectures proposed for providing integrated or differentiated services do not address the QoS aspects required to meet the demands of time-critical applications [21]. Guaranteeing sustained QoS in dynamic situations requires accurate system awareness through monitoring and proactive QoS and resource management.

### 3.2 Scalable and efficient techniques

QoS and middleware services for both Internet applications and large distributed systems have been extensively investigated by many researchers, including the authors [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]. However, none of the existing solutions are designed to address the challenges encountered in pervasive computing when dealing with an extremely large number of users/applications or dynamic, unforeseen situations. In the near future, many millions of sensors (camileuns) and software agents (delegents) will be deployed to carry out pervasive computing services using the networking infrastructures. These services not only have different priorities and QoS requirements, but the priorities and requirements dynamically change depending on the changes in the environment. For example, consider a patient under a home care program. A community of delegents representing the patient, doctor, and nurse continuously monitors the patient and provides health services as prescribed by the medical team. Under normal conditions, the priority for communications among the delegents of this continuous community may be high. However, if there is a crisis (such as a terrorist attack or an earth quake), not only the system load scales up dramatically, but the priorities associated to this and many other communities (such as movie times or webcasts) are lowered. The existing middleware solutions, whether centralized or distributed, are not equipped to handle such dynamic situations. *Any solution for QoS and middleware services for pervasive computing must scale as the number of users and applications increase or shrink and must be able to seamlessly adapt to dynamically changing conditions and priorities*.

### 3.3 Performance metrics:

To evaluate different QoS and RM middleware services in pervasive computing applications, there is a need for a set of standard performance metrics. Obviously, there are several well-defined, well-understood metrics, such as end-to-end delays, bandwidth utilization, and efficiency, which are also applicable to middleware services in pervasive computing. However, better metrics are needed not only to measure the overall performance, but also gauge the complexity and overhead, robustness (error margins in estimations and predictions), degree of scalability, quality (performance vs. resources used), degree of openness (availability for modification), and degree of testability of middleware services in different applications. Additionally, to the best of our knowledge, currently there is no benchmark (suite), as a measuring stick, to evaluate and compare different middleware services in pervasive computing applications.

## 4. Proposed Solutions

We present three solutions for the above mentioned challenges. First, we propose efficient middleware services that are useful in pervasive computing applications. These services will be implemented as delegents (software agents) in PICO environments. Second, we use PICO's community computing as a means to dynamically, and on a *just-in-time* basis, integrate the middleware delegents with the application delegents to form middleware communities. This approach will provide us with the basis to address scalability, adaptability, and efficiency in middleware services. Finally, we introduce new performance metrics and define the framework for a benchmark suite to evaluate middleware services for pervasive computing applications.

***Building block QoS and RM middleware services for communication support in pervasive computing***:

In the next subsection we will present a novel scheme for QoS (along the bandwidth and end-to-end delay dimensions) and resource management in dynamic, heterogeneous networks, using the community computing concept. The proposed QoS management and negotiation method, however, relies on a set of building block middleware services that are described in this section. We believe that a successful QoS and resource manager in a dynamically changing environment must: i) be aware of the current state of resources (e.g., network load), ii) know the application requirements, iii) be able to predict QoS violations, iv) be able to diagnose the causes of violations, v) succeed in managing and allocating the resources, and vi) have the tools to carry out QoS and resource management decision. *This paper focuses on the overall architecture of the proposed services and their integration. The working details of our middleware*

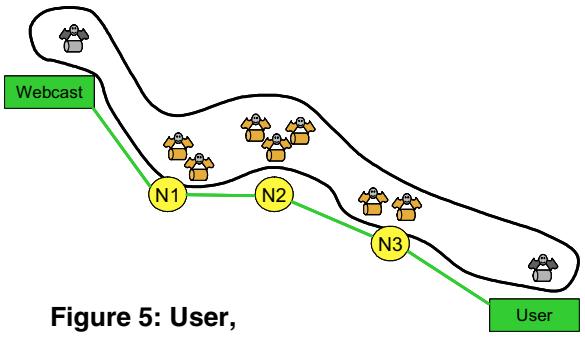


**Figure 5: User, webcast application**

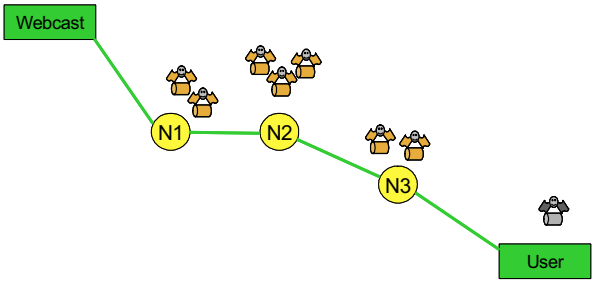


**Figure 6 Middleware services community**

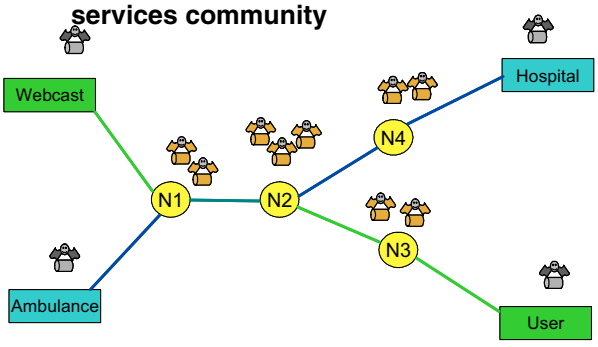


**Figure 7 Ambulance, hospital application**

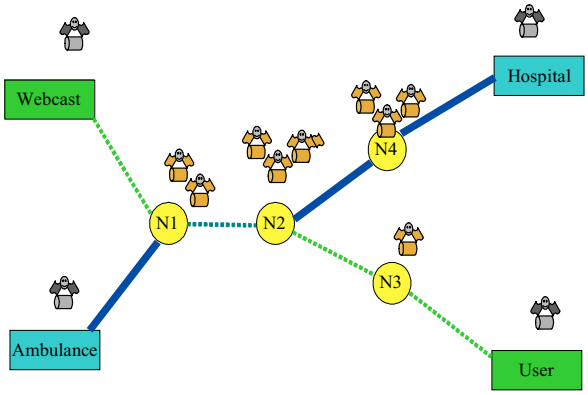


**Figure 8 A larger middleware community**

*services will be available on our web site (PIC) and in future publications.*

The building block QoS and RM middleware services used in our scheme include: system monitor, application profiler, resource manager, and action delegents.

The **monitor** delegents continually and autonomously collect and accumulate network performance data. $K$ ($K \geq 1$) neighboring network resources (network nodes and links) are monitored by a dedicated monitor delegent. $K$ is a system specified parameter and its value depends on the size of the network, communication latencies, and computational speed of the network processors.

The application **profiler** delegents are designed to profile communication patterns (e.g., type and priority) and their performance characteristics (e.g., bandwidth and end-to-end delays) for applications. The profiled information is later used in resource management decisions and for prediction of QoS violations.

**Resource manager** delegents are used to allocate communication requests to network resources in a dynamic and efficient manner. Currently we use heuristic algorithms to carry out resource management [28]. However, we are investigating the application of Game theory to solve this problem. We plan to formulate resource management operations as a bidding game among communication delegents that compete for the limited network bandwidth and queuing storage. This approach is flexible in adjusting the trade-off between the quality of solution and the complexity of the algorithm.

***Putting it all together:*** Once a QoS or resource management decision is made, it must be enacted by a QoS **action** delegent. Scalable, efficient (just-in-time), synergistic integration of middleware services into pervasive computing applications is critical to PICO. As previously discussed, any solution for QoS and middleware services for pervasive computing must scale as the number of users fluctuates and must be able to seamlessly adapt to dynamically changing environments and priorities. We assume that in the near future active networks [33, 34, 35] will be deployed for both intranet networks and the Internet. In active networks, the network nodes are no longer simple routers; rather, they have processing (even multiprocessing) capabilities and are capable of analyzing and dynamically routing communication packages. In such an infrastructure, middleware services can be implemented as delegents that exist on network nodes, providing QoS services as previously described. We now propose to use the community computing principle to form *just-in-time* middleware service communities to address pervasive computing

applications' dynamic needs. The following example explains our idea.

Consider Figure 5, in which a user would like to hear the webcast of a game on the Internet, going through network nodes N1, N2, and N3. As depicted in Figure 6, the delegents of the user, webcast, and middleware services (running on network nodes, providing QoS and RM services) will form a community to ensure the proper bandwidth is allocated to this communication so the user would receive the requested broadcast quality. Now assume a high priority, high bandwidth communication request between an ambulance and hospital is issued, which utilizes nodes N1 and N2 (Figure 7.) At this point, the middleware delegents running on the network nodes N1 through N4 form a larger community (Figure 8) to ensure the requirements of the ambulance-hospital communication is met while providing a best-effort communication to the user-webcast connection. The following points can be observed from this example:

1. Middleware service communities are formed just-in-time and only when needed.
2. Middleware services are only limited to the affected communications, thus limiting overheads and impact on wider area networks.

Both scalability and adaptability are achieved because of points 1 and 2.

However, the integration of QoS and RM services into pervasive computing applications is an entirely different matter. In this section we describe a QoS manager that will use the building block middleware services, such as monitoring, profiling, resource management, and QoS action, to negotiate QoS and allocate resources as needed. We solve this problem by introducing "*QoS manager delegents*," which are dynamically formed as middleware community managers. The function of the middleware community manager (QoS manager delegent) is to optimize the QoS requests within its jurisdiction (community) using the building block middleware delegents. *This simple and elegant solution allows us to dynamically form QoS and RM services as they are required. Additionally, the system is scalable in the sense that at any given time multiple middleware communities can locally (within their community) optimize their QoS requirements*. Two questions remain: how will the QoS manager delegents function? and how will the QoS manager delegents be dynamically formed? These questions will be answered next.

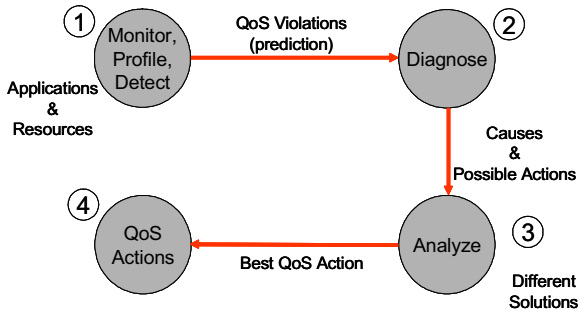


**Figure 9: QoS manager**

**QoS manger functionality**:

Figure 9 depicts our general solution approach for the QoS manager delegent functionality [36, 37]. In step 1, the QoS manager collects monitoring and application profiling information from the monitor and profiler delegents in the community. It then uses communication QoS requirements and the systems state information to detect QoS violations – *we plan to investigate prediction techniques to predict violations in near future.* Once a violation is detected, the causes of the violation are diagnosed and possible solutions are proposed in step 2. We currently use a simple, rule-based inference method [38] to implement an efficient diagnosis algorithm. In step 3, different RM solutions obtained from the resource management delegents are analyzed to determine the best QoS action or RM allocation to implement. Our analysis is based on "*robustness"* and performance metrics defined in the next section. Finally, in step 4, the selected QoS action (message duplication, re-routing, or re-scheduling) is carried out by the action delegents.

**Dynamic formation of QoS manger delegents**:

Middleware communities are formed whenever there is a significant change in the state of the system, requiring QoS management services (e.g., Figures 5-8). Therefore, a dedicated QoS manager delegent needs to be dynamically created whenever a middleware community is formed. Initially we distribute *m* dormant QoS manager delegents among *n* network nodes (one per network domain). If a community is contained in a network domain, then the QoS manager for that domain will serve as the QoS manager for the community. Whenever a middleware community crosses more than one domain, then the nearest, least loaded QoS manager delegent is selected to join the community and serve as its QoS manager. In the future we plan to investigate a more dynamic approach in which we can: *i) embed the QoS manager functionality in every QoS delegent described and ii) whenever the community is formed, we can deploy a voting/polling algorithm to determine which QoS delegent in the community will also serve as the QoS manager delegent*. Naturally, whenever a middleware community is dismantled, its QoS manager delegent is released (returned to dormant state).

***Performance metrics***:

As previously discussed, there is a need for a set of comprehensive and standard performance metrics to evaluate different QoS and RM middleware services in pervasive computing applications. Of course well-defined, well-understood metrics, such as end-to-end delays, bandwidth utilization, and efficiency are useful. Additionally, we propose several other performance metrics that are useful in the context of middleware services for pervasive computing applications, including:

*Performance*: metrics for QoS/RM middleware services that include: latencies of different components, such as resource allocation routines, whether or not the latencies of QoS/RM delegents are deterministic, and complexity/overhead of the QoS/RM delegents.

*Robustness*: Decisions about how to allocate resources are often based on estimated or predicted values of communication delays and system parameters. However, the actual QoS that is delivered may be less than predicted due to changes in circumstances such as sudden link/router failures, higher than expected system workloads, or inaccuracies in the estimation of communication delays and system parameters. An important question then arises: given a system design, what extent of departure from the assumed or predicted circumstances will cause the QoS to be unacceptably degraded? That is, how *robust* is the system? A useful robustness metric defined in [39, 40]can be easily adopted in pervasive computing.

*Quality*: This is probably one of the most important, as well as one of the most difficult to measure, assessment metrics for evaluation of QoS/RM middleware services. A quality metric may include:

QoS violation rate: This metric can be computed post-mortem with the notion that a better quality QoS/RM middleware service manages the resources during the execution such that there are fewer QoS violations.

Ratio of QoS to resources consumed: If the degree of QoS provided by QoS/RM services can be measured as a function of QoS parameters (meeting end-to-end deadlines or bandwidth requirement), then this ratio will give a higher mark to a QoS/RM service that achieves a higher degree of QoS with fewer resources.

*Sensitivity*: Sensitivity of a QoS/RM action to changes in number of users (scalability) and environmental conditions (adaptability).

Openness: Does the QoS/RM middleware services have an open architecture, allowing different/new components/delegents to be introduced into the systems?

Testability/Verifiability: Are the QoS/RM middleware services easily testable and verifiable?

***System prototyping and benchmark suite for validation and evaluation***:

We are in the process of designing and developing a middleware benchmark suite to evaluate and compare different middleware services in pervasive computing applications. The high level application will be the *emulated* prototype telemedicine application being developed under the PICO project. We are well aware of the ethical, privacy, and social issues we would encounter if telemedicine pervasive computing applications were to be deployed in the real-world today. Nevertheless, we choose to proceed with telemedicine as our target prototype application because: i) we, including many doctors we have consulted at the UT Southwestern Medical Center, believe it to be the "*killer application*" for pervasive computing, ii) it best represents the dynamic characteristics found in pervasive applications, and iii) we believe that as pervasive computing becomes more prevalent in society, pervasive medical applications will become more common, at least on a voluntary basis.

The PICO emulated prototype is designed to behave like a typical dynamic telemedicine application described in Section 3. Laptops and PDAs are used to emulate patients, heart-monitors, cell phones, ambulances, hospitals, etc. These emulated entities are monitored and controlled by their associated delegents and actually communicate over a real heterogeneous network (wired, wireless, intranet, and Internet). We are also developing a methodology to inject dynamic changes in the system behavior on a *deterministic* and *replicable* manner. To be able to evaluate the middleware delegents suite fairly and validly, we must be able to exactly repeat the same *dynamic* environment and conditions from one experiment to another. For example, if the communication load is increased at a given time during an experimental run, then exactly the same communication load must be increased at the exact same time during the next run of the experiment. To achieve deterministic and replicable experiments under dynamically changing conditions, we are developing "scenario files" and "experiment generators."

A scenario file is used to define the dynamic events that may take place during an experimental run of the benchmark suite (telemedicine application plus the middleware services). Thus, a scenario file will specify a sequence of events to take place at given *relative time units*, starting with time 0. For example, we may specify that at time 0 the communication load is 0 and no application is executing. At time 5, a user on camileun *x* requests to watch a webcast on camileun *y*. At time 10, camileun *z* detects a heart attack, and so on.

An experiment generator is a meta level application that reads the scenario file and activates the specified events at the given times in the test-bed prototype environment. In other words, an experiment generator is the event driver in our event-driven simulation test-bed.

## 5. Performance Evaluation

In this section we present some early results from our implementation of a small, proof of concept system for PICO and the proposed QoS middleware. The system set up is based on the prototype environment explained in the previous subsection, using the webcast – ambulance scenario described in Section 5 (Figures 5 – 8). Four Pentium-based computers, running Linux, are used to represent the user, webcast, ambulance, and hospital camileuns. The interconnection among the camileuns is emulated using 12 Linux-based PCs, emulating network switches and routers.

Figure 10 depicts the results of our first experiment. The X axis shows the experiment duration in seconds, starting from time zero. The Y axis shows the webcast communication latency (in seconds) between the webcast site and the user. The desired latency (or soft deadline) for this communication is 0.002 sec. The inter-arrival rate for this continuous transmission is exponential with a mean of 0.01 sec and the traffic size is 1K bytes per frame. As shown in Figure 10, the webcast latency is below the deadline up to around 100 seconds into the experiment. At this point additional traffic is injected into the network, causing the latency to surpass the deadline around time 112 seconds. This violation is then detected by the QoS monitor, causing activation of the QoS manager. The QoS manager re-routes the traffic through other nodes with more bandwidth availability, causing the latency to drop below the deadline around 125 seconds into the experimentation.

Figure 11 depicts the results of a similar experiment involving two communications. The graph denoted by diamond-shaped data points shows the behavior of the webcast communication with the same traffic characteristics as before. The graph denoted by the triangle-shaped data points shows the communication behavior for the ambulance – hospital transmission. The desired latency for this communication is 0.2 sec. The inter-arrival rate is exponential with a mean of 0.01 sec and the traffic size is 8K bytes per frame. Unlike the continuous webcast traffic, the ambulance – hospital transmission is sporadic with an exponential duration with a mean of 600 seconds. As depicted in Figure 11, up to around time 70, everything is fine. At this time, the ambulance – hospital transmission begins, sharing two network nodes with the webcast communication. As more and more network bandwidth is allocated to the ambulance communication, the latency of the webcast increases and eventually exceeds its deadline around time 95. Also, the ambulance-hospital communication latency eventually exceeds its deadline because

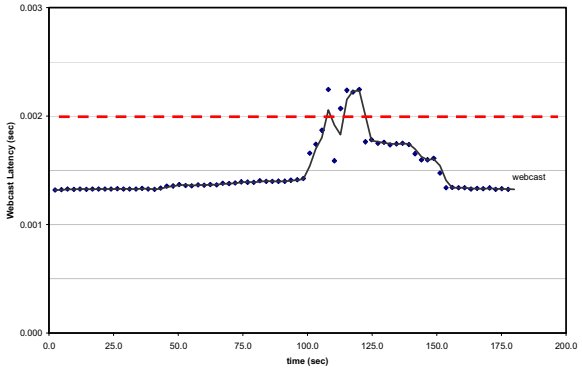


**Figure 10 Experiment 1: Webcast communication behavior in PICO.**

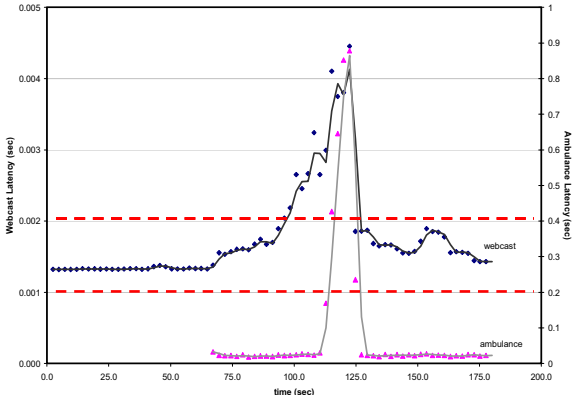


**Figure 11 Experiment 2: Webcast and ambulance-hospital communication behavior in PICO.**

sufficient bandwidth cannot be allocated to its transmission. At this point, the QoS manager re-allocates the webcast transmission to other network nodes, allowing the two communications to proceed while meeting their QoS requirements.

## 6. Conclusions

In this paper we introduced middleware services that are required to address the challenges related to adapting to dynamically changing situations in pervasive computing applications. In particular, these services are designed to meet communication QoS requirements and achieve scalability in large-scale pervasive computing applications over heterogeneous network infrastructures. The community computing is used as a unifying method to dynamically integrate the middleware services with each other and with high level applications on a just-in-time basis. In addition, we proposed new performance metrics and a benchmark suite approach to evaluate middleware services in pervasive computing. Early performance evaluation results from a proof of concept implementation validate the proposed approach.

***Acknowledgement:*** The material presented in this paper is based on work supported by the National Science Foundation under Grant No. 0129682.

## 7. References

[1] M. Weiser, "The computer for the 21st Century," Sci. American, Sept.1991.

[2] Ambient Computing: http://www.ambientcomputing.com/company.html.

[3] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, Project Aura: Towards Distraction-Free Pervasive Computing", IEEE Pervasive Computing, special issue on "Integrated Pervasive Computing Environments", Volume 1, Number 2, April-June 2002, pages 22-31.

[4] M.L. Dertouzos, "The Future of Computing," Scientific American, pp.52-55, 1999.

[5] M Kumar, S K Das, B Shirazi, D Levine, J Marquis, and L Welch , Pervasive Information Community Organization (PICO) of Camileuns and Delegents for Future Global Networking LSN Workshop, Vienna, USA, March 12-14, 2001, http://www.ana.lcs.mit.edu/~sollins/LSN-Workshop/papers/.

[6] B. Sung, B. Shirazi, M Kumar, "Pervasive Community Organization," Eurasia 2002, Tehran, November 2002.

[7] M. Esler, "Next century challenges: data-centric networking for invisible computing," The Portolano Project at the University of Washington, Proceedings of Fifth Annual ACM/IEEE MOBICOM 97, p. 256-62, Seattle WA, USA, September 1997.

[8] Endeavour project: http://endeavour.cs.berkeley.edu/.

[9] Equator project: http://www.iam.ecs.soton.ac.uk/projects/equator/.

[10] G. Banavar, J. Beck and E. Gluzberg, "Challenges: An application model for pervasive computing," in Proceedings of 6th MOBICOM 2000, pp.266-274, 2000, Boston MA, USA.

[11] Hopper, A., The Royal Society Clifford Paterson Lecture, 1999, Sentient Computing. AT&T Lab Cambridge Technical Report 1999.12, 1999: p. 1-10.

[12] PIMA project: http://www.research.ibm.com/PIMA/.

[13] Proactive computing: http://www.intel.com/research/proactivepdf.pdf/.

[14] Bill Schilit, "Mega-Utilities Drive Invisible Technologies," IEEE Computer, Feb. 2003, pp. 97-99.

[15] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," IEEE Personal Computing, August 2001, pp. 10-17.

[16] SMART DUST, Autonomous sensing and communication in a cubic millimeter, http://robotics.eecs.berkeley.edu/~pister/SmartDust/.

[17] Berkeley Sensor and Actuator Center, http://www-bsac.eecs.berkeley.edu/.

[18] Personal Area Networks: http://grouper.ieee.org/groups/802/15/.

[19] PICO tech report and website: http://www.cse.uta.edu/pico@cse/.

[20] R.H. Katz, "The Endeavour Expedition: Chating the Fluid Information Utility," Web/Internet Site**:** http://endeavour.cs.berkeley.edu/proposal/.

[21] The Internet's Coming of Age, National Academy Press, Computer Science and Telecommunication Board, NRC, ISBN: 0-309-06992-0.

[22] Seoung-Bum Lee and Andrew T. Campbell "INSIGNIA: In-bandsignaling support for QOS in mobile ad hoc networks", Proc of 5thInternational Workshop on Mobile Multimedia Communications (MoMuC,98) , Berlin,Germany, October 1998.

[23] Globus project: http://www.globus.org/.

[24] Grid computing: http://www.gridcomputing.com/.

[25] C.D. Cavanaugh, L.R. Welch and B.A. Shirazi, E-N Huh and S. Anwar, "Quality of Service Negotiation for Distributed Dynamic Real-time Systems," Lecture Notes in

Computer Science, #1800, Springer Verlag, pp.757-765, April 2000.

[26] E.H. Huh, Lonnie R. Welch, Behrooz A. Shirazi, Charles Cavanaugh, Shafqat Anwar, "Heterogeneous Resource Management for Distributed Real-time Systems," Heterogeneous Computing Workshop (HCW'00), April 2000.

[27] L.R. Welch, B. Ravindran, B.A. Shirazi, C. Bruggeman, "Specification and Analysis of Dynamic, Distributed Real-time Systems," Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS'98), pp. 72-81,1998.

[28] L.R. Welch, P.V. Werme, L.A. Fontenot, M.W. Masters, B.A. Shirazi, B. Ravindran, and D.W. Mills, "Adaptive QoS and Resource Management Using A Posteriori Workload Characterizations," Fifth IEEE Real-Time Technology and Applications Symposium (RTAS'99), May 1999.

[29] W. Lau, M. Kumar. And S. Venkatesh, A cache-based mobility-aware scheme for real-time continuous media delivery in wireless networks, International conference on Multimedia and Expo, Tokyo, August 2001.

[30] W. Lau, M Kumar, and S. Venkatesh, A Cooperative Cache Architecture in Support of Caching Multimedia Objects in MANETs, 5th ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'00), Atlanta, September 2002.

[31] S. Ali, J-K. Kim, H. J. Siegel, A. A. Maciejewski, Y. Yu, S. B. Gundala, S. Gertphol and V. Prasanna, "Greedy Heuristics for Resource Allocation in Dynamic Distributed Real-Time Heterogeneous Computing Systems," 2002 International Conference on Parallel and Distributed Processing Technologies and Applications (PDPTA 2002), cosponsors: CSREA et al., Vol. II, pp. 519-530, Las Vegas, NV, June 2002.

[32] T. D. Braun, H. J. Siegel, and A.A. Maciejewski, "Static Mapping Heuristics for Tasks with Dependencies, Priorities, Deadlines, and Multiple Versions in Heterogeneous Environments," 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), sponsor: IEEE Computer Society, in the CD-ROM proceedings, Fort Lauderdale, FL, Apr. 2002.

[33] D. L. Tenenhouse and D. J. Wetherall. *Towards an active network architecture*. Computer Communication Review, 26(2), 1996.

[34] K. W. Chin, M. Kumar, AMTree : An Active Approach to Multicasting in Mobile Networks, ACM/Baltzer, Mobile Networks and Applications, Vol. 6, No. 4. 2001, pp.361-376.

[35] K.W. Chin, M. Kumar, and C. Farrell, A Model for Enhancing Connection Rerouting in Mobile Networks, ACM/Baltzer Wireless Networks, Vol. 7, No. 3, 2001, pp. 249-267.

[36] B.A. Shirazi, L.R. Welch, B. Ravindran, C. Cavanaugh, and E. Huh, "DynBench: A Benchmark Suite for Dynamic Real-Time Systems," Journal of Parallel and Distributed Computing Practices, pp. 89-108, March 2000.

[37] L.R. Welch and B.A. Shirazi, "A Dynamic Real-time Benchmark for Assessment of QoS and Resource Management Technology," Fifth IEEE Real-Time Technology and Applications Symposium (RTAS'99), May 1999.

[38] B. Li , K. Nahrstedt, QualProbes: middleware QoS profiling services for configuring adaptive applications, IFIP/ACM Intl./ Conference on Distributed systems platforms, New York, pp. 256-272, April 2000.

[39] S. Ali, A. A. Maciejewski, H. J. Siegel, and J-K. Kim, "Definition of a Robustness Metric for Resource Allocation," 17th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2003), to appear, Apr. 2003.

[40] S. Ali, A. A. Maciejewski, H. J> Siegel, and J.-K. Kim, "On the Robusteness of Resource Allocation for Parallel and Distributed Computing and Communications," The 2003 International Multiconference in Computer Science and Computer Engineering, June 2003. Invited Speech by H .J. Siegel.