PerLa: a Language and Middleware Architecture for Data Management and Integration in Pervasive Information Systems

ABSTRACT

A declarative SQL-like language and a middleware infrastructure are presented for collecting data from different nodes of a pervasive system. Data management is performed by hiding the complexity due to the large underlying heterogeneity of devices, which can span from passive RFID(s) to ad-hoc sensor boards to portable computers. An important feature of the presented middleware is to make the integration of new device types in the system easy, through the use of device self-description. Two case studies are described for PerLa usage, and a survey is made for comparing our approach with other projects in the area.

Existing System

Disadvantages:

However, the existence of an infrastructure allowing a systematic data collection process is fundamental both to implement working prototypes of autonomic components (and other high level abstractions) and for engineering the development of pervasive applications. The main goal of present system is to introduce the requirements and the issues related to the design and the implementation of a solid substratum, able to provide sampling
and first data processing functionalities, abstracting from devices' specificities. We will show how this goal can be achieved, both through the definition of a query language and the implementation of a middleware infrastructure.

Proposed System

A goal of the PerLa project is to extend this approach in order to support a whole pervasive system. However, the database abstraction in the ”world of sensors” should handle some additional issues that do not exist in traditional databases. The dynamics of the nodes must certainly be considered since a device can frequently join or leave the pervasive system, for example when the status of the battery or of the wireless connection changes. Especially for monitoring purposes, the use of a declarative language is the approach that better allows to abstract the pervasive system, hiding the technical details related to device access. The advantage of defining a *SQL-like* language is the reduction of the effort needed by a user already experienced with standard SQL to learn it.

IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Modules:

1. Query Parser and Query Analyzer:

The Query Parser and the Query Analyzer compose the user interface of the entire PerLa System. The PerLa Language defines two classes of queries:

• Low Level Queries (LLQ): define the behavior of single devices

• High Level Queries (HLQ): manipulate data streams generated by LLQs.

The Query Parser is PerLa’s front-end to final users. This component receives and parses textual queries and sends them to the Query Analyzer. The Query Analyzer is responsible for the creation of both the High Level Query Executor (HLQE) and the Low Level Query Executor (LLQE).

2. High Level Query Executor and Low Level Query Executor:

Low Level Queries (LLQ) allow one to precisely define the behaviour of a single device. The main role of a low level statement is to define the sampling operations, but also to allow the application of SQL operators (such as: grouping, aggregation, filtering) on sampled data. When a LLQ is injected in the system, the set of Logical Objects that meet all the requirements to execute it is computed. Then, an instance of the query is deployed on each selected logical object; data produced by all these instances are collected in a stream that can be further manipulated by *High Level Queries* or sent as output to the final user.

High Level Queries (HLQ) allow one to define data manipulation operations on streams generated by Low Level Queries or other High Level Queries. These kinds of queries do not directly deal with the logical object abstraction since they operate only on streams independently from their sources. As a consequence no ad hoc clauses are needed: the syntax and the semantics of these statements are similar to those of streaming DSMS.

3. FPC factory:

The main software module of the low level support layer is the *Functionality Proxy Component* (FPC). As suggested by its name, the FPC’s primary task is to act as a proxy among sensing nodes and the rest of the PerLa System. FPCs are created by means of the *FPC Factory*, a software module responsible for PerLa’s Plug & Play system. FPC Registry is basically a Main Memory Database (MMDB) in which a reference to any existing FPC is maintained. Its main goal is to provide support for the evaluation of the EXECUTE IF clause of Low Level Queries. After a new FPC is generated, the factory registers it; the device metadata (supported attributes and events, maximum sampling frequencies, etc.), the attribute values and the remote reference to the FPC object are stored in the FPC Registry.

4. Channel Manager:

The PerLaMiddleware deals with this problem using a specific component, the Channel Manager. This software module is responsible for the creation of Virtual Channels. Virtual channels use the existing communication facilities of the pervasive system to provide a bidirectional, point-to-point logical communication layer. By means of this software abstraction, the FPC and the corresponding sensing device can communicate among them as if they were connected through a dedicated link. The Channel Manager handles every routing and multiplexing operation required to conceal the physical communication difficulties (e.g., a single physical link shared among many sensing nodes).

System Configuration:-

H/W System Configuration:-

Processor	-	Pentium –III
Speed	-	1.1 Ghz
RAM	-	256 MB(min)
Hard Disk	-	20 GB
Floppy Drive	-	1.44 MB
Key Board	-	Standard Windows Keyboard
Mouse	-	Two or Three Button Mouse
Monitor	-	SVGA

S/W System Configuration:-

❖ Operating System	:Windows95/98/2000/XP
❖ Front End	: java, jdk1.6
❖ Database	: My sqlserver 2005
❖ Database Connectivity	: JDBC.