PAQ: Adapting Query Fidelity in Dynamic Networks

Vasanth Rajamani, The University of Texas at Austin
Jamie Payton, The University of North Carolina at Charlotte
Christine Julien, The University of Texas at Austin
Gruia-Catalin Roman, Washington University at St. Louis

TR-UTEDGE-2009-003

© Copyright 2009
The University of Texas at Austin
Ubiquitous computing applications have been enabled by a tight embedding of computation to the environment, support for dynamic networks, and the physical distribution of application components. Applications respond to frequent changes, detected through persistent monitoring of the environment in domains such as construction sites, where maintaining safety is essential; in military applications where asset management is important; and in home networks that monitor intrusions and energy consumption.

Programming applications that monitor such open and rapidly changing networks can be challenging. A persistent query can simplify the development of applications that require continuous monitoring. A persistent query allows a programmer to describe the data of interest without requiring him to specify network communication details. At the abstract level, a persistent query may be defined as the continuous reporting of relevant state changes in a dynamic network. However, accurate evaluation of a persistent query that continuously reports all state changes is feasible only in relatively static networks; the cost of continuous monitoring is prohibitive in networks that exhibit rapid change.

To support application development, we have built the Persistent Adaptive Query (PAQ) middleware [4]. PAQ approximates a persistent query using a sequence of one-time queries, i.e., queries evaluated once at a given time over some portion of the network. Although there exists query processing systems that execute long-lived queries for streaming data [1, 2, 3], PAQ more closely simulates continuous monitoring by providing non-relational operators allowing temporal analysis to be specified easily.

Figure 1 shows a data flow representation of our middleware. There are four key strategic components of PAQ: 1) an inquiry strategy, which controls when, how, where, and what type of one-time queries are issued; 2) an integration strategy, which specifies how a history generated by consecutive one-time queries is transformed into an approximation of the persistent query; 3) an introspection strategy, which assesses a persistent query’s execution to determine its suitability; and 4) an adaptation strategy, which maps a current inquiry strategy and introspection metric to a new inquiry strategy, that may in turn use new integration, introspection, and adaptation strategies. We briefly overview each of these strategies; see [4] for complete details.

Inquiry Strategy. In evaluating a persistent query’s component one-time queries, it is important to understand the behavior of an underlying query processing protocol. For example, flooding may be expensive by may achieve strongly consistent results, while randomly sampling a few nodes provides much weaker consistency, but is much less expensive. The manner in which we query the environment, the inquiry strategy, includes not only the one-time query protocol but also the frequency of the one time queries.

Integration Strategy. A persistent query’s result is formed from the component queries using an integration strategy, a function \( f \) evaluated (and reevaluated) over the sequence of one-time query results. Examples of integration strategies include cumulative integration, which returns a union of all of the one-time query results; stable integration, which returns the intersection of all of the one-time query results;
and additive and departure integrations, which monitor the changes in the sequence of one-time queries.

**Introspection Strategy.** As conditions change, applications must determine the suitability of their inquiry strategy. An *introspection strategy* is a function over the tuples in a result that describes the result’s quality. Examples of introspection strategies include semantic discovery, which is simply a binary metric triggered when a particular value is discovered, change rate strategies, which detect the rate at which new values are added or deleted, and environmental introspection, which detects contextual properties.

**Adaptation Strategy.** Based on the value of an introspection metric, an application can specify an *adaptation strategy* to govern changing the inquiry strategy. This strategy maps a current inquiry strategy and a value for an introspection metric to a new inquiry strategy (and potentially new integration, introspection, and adaptation strategies).

**AN APPLICATION IN PAQ**

To demonstrate how the PAQ middleware operates in a dynamic sensor network, consider a scenario for an intrusion detection application in a home. The home is instrumented with a variety of devices for monitoring, recording, and reacting to anomalous conditions that may be indicative of an intrusion. These sensors may be employed for other purposes as well, for example for controlling the occupants’ living environment or monitoring energy usage, but we focus on the intrusion detection application for this demonstration.

For now, consider the intrusion detection application issuing a persistent query over a set of motion sensors placed around the home. For common (e.g., daytime), constant monitoring, the application is conservative in its use of network resources, so the query initially randomly samples the motion detectors in search of any sensors that have detected motion levels above a given threshold. If this query at any time detects a potential intrusion, the threat level is elevated, the homeowner is alerted, but also likely desires additional information to determine the actual situation. This query will introspect on its results to respond to a detection event and will adapt the query to a much more frequent flood of the entire perimeter to corroborate the initial detection:

- **inquiry strategy**: random sampling with a low probability and a low frequency
- **integration strategy**: windowed cumulative integration, to acquire all motion detections over the last 20 seconds
- **introspection strategy**: semantic discovery of any motion above the threshold
- **adaptation strategy**: upon detection of a motion event, flood the network with high frequency

In the next phase, the persistent query is simply looking for corroboration of the initial detection event using a more expensive but robust approach. This new query is:

- **inquiry strategy**: flooding with a high frequency
- **integration strategy**: cumulative integration, to acquire all motion detections since adapting the persistent query
- **introspection strategy**: semantic additive change rate to measure the rate of discovery of corroborating detections
- **adaptation strategy**: if more than 10% of the results are new detections, localize around the area of intrusion

This second adaptation policy allows the network’s resources and the application’s attention to be focused on exactly the desired area. The persistent query continues to monitor this area, adapting the size and the scope of the area as the detected values change. This continues until the danger dissipates, when the query returns the original mode.

Programming such a flexible and expressive application using low-level sensing, communication, and programming technologies is cumbersome and error-prone. Through the persistent query abstraction and the four-component query strategy, PAQ simplifies the interface between the application developer and the network, making rapid development and deployment of highly capable applications feasible. This demonstration shows first how to use the PAQ constructs, how the introspection and adaptation processes enable responsive applications, and how query information can be used in applications’ decision processes.

**CONCLUSIONS**

The PAQ middleware and its persistent query abstractions offer a leap forward in making development of ubiquitous computing applications that entail long-lived monitoring approachable. This demonstration walks through a complete application, demonstrating the implementation, deployment, and execution of expressive persistent queries for a home intrusion detection application. By demonstrating not only how to construct expressive persistent queries, but also their simplicity and the expressiveness of the results, we demonstrate the feasibility of bringing ubiquitous computing application development to fruition.

**REFERENCES**