Workshop Report

NSF Workshop on

Context Aware Mobile and Sensor Information Management (CAMM)

January 24-25, 2002
Providence, Rhode Island

Jointly funded by

IIS-0099128
Information and Data Management Program/IIS/CISE
Advanced Networking Research Program/ANIR/CISE
Communications Research Program/CCR/CISE
Signal Processing Systems Program /CCR/CISE

Sponsored by

ABC Virtual, West Des Moines, Iowa
University of Missouri-Kansas City, Kansas City, Missouri

Workshop Chairs

Vijay Kumar
University of Missouri, Kansas City
kumarv@umkc.edu

Stan Zdonik
Brown University
sbz@cs.brown.edu

Organizing Committee

Workshop Chairs

Vijay Kumar
SICE, Computer Networking
University of Missouri - Kansas City
5100 Rockhill Road
Kansas City, MO 64110
Voice: (816) 235-2366
Fax: (816) 235-5159
kumarv@umkc.edu

Stan Zdonik
Computer Science
Brown University
P.O. Box 1910
Providence, RI 02912
Voice: (401) 863-7648
Fax: (401) 863-7657
sbz@cs.brown.edu

Committee Members

Name	University	E-mail
Panos Chrysanthis	University of Pittsburgh	panos@cs.pitt.edu
Maggie Dunham	Southern Methodist University	mhd@seas.smu.edu
Anupam Joshi	Univ. of Maryland at Baltimore County	joshi@cs.umbc.edu
Krithi Ramamritham	IIT Bombay, India	krithi@cse.iitb.ac.in
Ouri Wolfson	Univ. of Chicago, Illinois	wolfson@horizon.eecs.uic.edu

Participants

Tulay Adali	U Md, Baltimore	Ravi Jain	Telcordia
B. R. Badrinath	Rutgers	Christian Jensen	Aalborg, Denmark
Sujata Banerjee	HP	Anupam Joshi	U Md, Baltimore
Elisa Bertino	Milan, Italy	P. Krishnamurthy	U Pittsburgh
Bharat Bhargava	Purdue Univ.	Vijay Kumar	U MS, Kansas City
Ugur Cetintemel	Brown Univ.	Wang-Chien Lee	Penn State Univ.
Sharma Chakravarthy	UT Arlington	Sanjay Madria	U Missouri, Rolla
Mitch Cherniack	Brandies Univ.	Dave Maier	OGI
Panos K. Chrysanthis	Univ. Pittsburgh	Ryusuke Masuoka	Fujitsu
Anindya Datta	Chutneytech	Gail Mitchell	BBN Technologies
Maggie Dunham	Southern Methodist	Jignesh Patel	U MI, Ann Arbor
Mike Franklin	UC Berkley	Hans Schek	ETH, Zurich
Johannes Gehrke	Cornell Univ.	Mike Stonebraker	MIT
Daniel J. Hebert	MITRE Corp	Ouri Wolfson	U Illinois, Chicago
Vinayak Hegde	ABC Virtual	Arkady Zaslavasky	Monash, Australia
Guerney D. Hunt	IBM	Stan Zdonik	Brown Univ.

NSF Observers

Julia Abrahams, Jean Scholtz, and Bhavani Thureisingham

Disclaimer

All opinions, findings, conclusions and recommendations in any material resulting from this workshop are those of the workshop participants and do not necessarily reflect the views of the National Science Foundation.

Executive Summary

There is a general agreement among researchers and developers that support for mobility has become one of the key requirements of today's information management infrastructure. Mobility improves productivity by using working time more efficiently. First-generation mobile architectures treat mobile devices like static client-server systems in which clients issue standard transactions [1] to servers. More responsive future applications will require different levels of interaction and service depending on the user's context. Context captures those aspects of the user environment that affect the observable behavior of the system. For example, if a shopper's assistant running on a PDA, which is fully connected via wireless channels, alerts a user of nearby bargains on interesting items, then the context would include the user's position and perhaps a list of things that the user has purchased recently. Context-aware mobile systems in particular represent a research challenge with a new vision of intelligent applications that can automatically provide users with information that is relevant and timely, and that can operate in often unpredictable mobile environments.

Context-aware mobile systems must be based on a coherent underlying infrastructure much as data processing applications have been based on database management systems. Mobility in information management has evolved as an important new discipline that presents many challenging problems. Many aspects of conventional information management must be re-examined, new approaches must be developed, and interactions between information and its consumers must change completely. Naturally, many new concepts such as location-dependent and location-aware query processing, pervasive computing, mobile computing, data streams, mobile sensors, and so on, have flooded the research arena. Furthermore, quite a few conferences, workshops, journals, etc. are promoting these emerging research activities. Unfortunately, the research communities for each of these areas are splintered. Although these are close-knit areas, there is little coordination in the effort to move toward holistic successful approaches. This workshop is an attempt to fill this void.

The workshop participants agreed that the notion of mobile information management is not confined to wireless communication devices such as cell phones, but pervades many other areas such as underwater vehicle navigation, traffic monitoring, and sensor-based systems. Its importance in biological science was also recognized. Thus, to understand the implications of context-aware mobility, the first workshop focused on identifying application areas and on the problems they present for information management. The workshop also tried to probe potential synergies between different sub-areas of computer science (e.g., information management and networking) and sensor network technology [11], data streams, continuous query processing, and related systems architecture and application development [12] were emphasized and discussed. The workshop participants recognized and strongly promoted application development and system architecture for stream data management. In particular the discussions focused on (a) system architectures, (b) profile and context definition and use, and (c) query processing over data streams.

The discussion uncovered a number of different disciplines, which appeared to be connected through the needs of mobility. These areas include networking, databases, pervasive computing, the Web, and personalization systems. The participants agreed that the connecting threads among these were still not very clear and decided that this link discovery and other more detailed aspects should be the topics of subsequent workshops.

1. Introduction

This workshop was intended to draw together scientists from various disciplines who are all interested in technical problems related to mobile information systems. The workshop also adds the challenge of moving towards an infrastructure that can support mobility by reacting to the environment and context of its users. The two days were quite productive, and an interdisciplinary conversation was begun. The rest of this document summarizes some of the findings of the meeting.

1. 1 Background

The idea of CAMM evolved in the 1999 NSF IDM workshop in Chicago. At this workshop, a group of participants discussed at length the importance of information management in the context of mobility. It was recognized that mobility is becoming a more important part of an individual's computing platform, and further research into the nature of the problems that mobility poses was highly desirable. The participants decided that one of the best ways to promote and coordinate further activities (research, development, etc.) was to organize a series of NSF-funded workshops to discuss a possible research agenda.

The vision of the workshops is to move toward an information processing discipline that supports complete mobility in the sense that the data, the processing nodes, and the users are all potentially mobile [13]. Furthermore, computing elements in such an environment would continue to operate at some reasonable level under intermittent disconnection. The organizing committee agreed that the complex nature of such an environment presented difficult new research issues that crossed traditional research boundaries. They saw a need to coordinate existing efforts in the contributing disciplines as well as the work of academics and practitioners. The Workshop series would be a major step in this direction.

This group was composed of Panos K. Chrysanthis, Maggie Dunham, Anupam Joshi, Vijay Kumar, Krithi Ramamritham, Ouri Wolfson, and Stan Zdonik and is often referred to as Mobile Mob or MM. The idea of the workshop series was further discussed at SIGMOD 2000 in Dallas, and a tentative framework of workshops was completed. We presented our plan for the workshop series to Dr. Maria Zemankova, Program Director of IDM, who strongly supported it and who involved other relevant NSF programs. The group identified the Co-PI's for the workshop series and submitted the proposal for the first workshop, which was jointly funded by these programs. The workshop was also sponsored by ABC Virtual, a software development company located at West Des Moines, Iowa, and University of Missouri-Kansas City, Missouri.

It was decided that the first workshop would held at Brown University, Rhode Island, on January 24-25, 2002. The subsequent sections discuss the topics the workshop examined in detail.

1. 2 Sensor Technology: A Driving Application

In Computer Science, most research opportunities result from "sea changes", i.e., changes in fundamental technology that enables a new class of applications or a new way of thinking about current applications. One such area identified by the workshop was "sensor technology" [14, 15]. Although sensor technology is not new, a sea change is occurring in its application domains, architecture, and physical dimensions [16, 17, 18]. Their use for capturing useful data about human organs, capturing random behavior of mobile objects, collecting data from unreachable locations identifies their pervasive capability. In some cases, for example

controlling an underwater vehicle, the wireless sensor network is the only way out. These changes in sensor technology will drive a new class of *monitoring applications*. This section describes the area of mobile sensor-based systems and the research opportunities that result.

The relentless pace of sensor technology will soon enable the following two classes of devices. First, there will be a *dime*, which can be a dime-size or smaller electronic device (e.g., a sensor), and could cost less than a dime.. These two types of devices will relay a programmable collection of information about the state of the object (e.g., temperature, location, identity, heart rate, muscle contraction) to a *beacon* (server) whenever the dime comes within a few feet of the *beacon*. In time, the dime will become a penny or smaller and the maximum distance it can be from the beacon will expand to several meters. In addition, scores of such devices will work in concert to provide information about objects of interest. For example, scientists may wish to study the movement of migratory birds, and medical specialists may wish to monitor human physiological functions.

The other class of devices will be the size of a *pack* of cigarettes. It will be an active dual-purpose device. If out of doors, it will be a Geographical Positioning System (GPS) device capable of reporting its position and other state information over wireless networks. If indoors, where GPS is ineffective, it will report the same state information to a beacon with a range of many meters. The cost of such a device will be perhaps $100 or less in the near future. Berkeley Motes (sold by Crossbow, Inc.) are a good example of this class of device.

Of course, there may be many other classes of devices in between a dime and a pack, as well as other higher-end devices with additional capabilities. In time, a dime or a pack will be attached to everything we care about including cereal boxes, cell phones, cars, and laptops. This will allow such objects to report their state through various wireless networks. We are already witnessing the extensive use of such sensors at home (e.g., smart houses and the interconnection of consumer audio/video systems). The military is interested in using dimes and packs to measure the physical and cognitive state of a soldier in stressful situations (e.g., Land Warrior). The software infrastructure to support the management of data in these environments is still in its infancy. In current applications, resource allocation is typically done in an ad hoc manner making it costly, error-prone and sometimes ineffective. Just as modern database management systems (DBMS's) have rationalized resource allocation for persistent memory systems, similar platforms for mobile sensor-based systems is much needed.

Monitoring Applications

We will use the term *monitoring application* to describe applications whose primary purpose is to maintain the state of objects over time and report *exceptional conditions* that require some sort of intervention. Below we discuss a few such monitoring applications.

Libraries: Many examples can be cited to illustrate the expanding use of monitoring applications. For example, university libraries would like to attach a dime to every book in its collection. There are several uses of this technology. Obviously, this will allow the library to detect when a book that has not been checked out is leaving the building. Besides theft deterrence, the library can essentially end the "lost book" problem. In cases when a book that is not checked out is not in its proper shelf location, the dime attached to the book can report its

position through a beacon network, and a clerk can be dispatched to reshelf it. Also, assuming the presence of a continuous power supply, the dimes will turn an annual inventory to find all books into a simple query of the sensor network.

Parking lots: This technology can also be used to manage a parking lot in which the sensors will report the status of a parking slot, record the issuance of a ticket to an unauthorized parked car, identify and report expired meters, etc. This type of application will be able to minimize the work of a parking attendant. More generally, property management systems, which currently put a plastic property sticker on objects of value, will be upgraded to this technology.

Monitoring patients: Many medical applications can make use of this technology [19, 20]. Sick patients can be sent home with a pack that will report vital signs in real time to a hospital network and ultimately to the attending physician. More generally, the current "medic alert" system whereby an elderly patient can press a button to quickly summon help can be substantially upgraded. A large collection of vital signs can be monitored, and active intervention can be automatically called without the patient having to push a panic button. The military is actively experimenting with technology that will report vital signs for soldiers in the field, so that a monitoring application can instruct him to drink if he becomes dehydrated and can summon a medic if he needs one.

Homeland security: Monitoring applications can be used to detect terrorist activities. A set of sensors can be mounted at vulnerable locations such as airports, air conditioning units of public buildings, municipal water supplies, etc. Software systems can then monitor dangerous situations such as release of poisonous gas and water contamination through toxic chemicals.

Plant operations: This application can also be used to manage large physical plants. For example, devices such as smoke alarms, motion detectors, security cameras, thermostats, etc., can be augmented with this technology. Today such devices operate largely in isolation. In the controlled environment of the future, input from a host of devices can be aggregated and correlated to create a model of the premises. From this model, complex conditions can be detected. For example, the system can turn down the heat in any room that appears to be unoccupied. The system can have a much better idea if there is an actual fire in a campus facility by correlating smoke density and temperature. Lastly, the system can use sensor identity to ensure that repair orders in fact are done to the correct object. Smart homes are typical examples of a preliminary version of such systems.

In summary, sensor technology will enable a class of applications that requires a totally new approach to deal with new types of information. Note that not only the applicability of sensor technology is much wider, it also significantly improves the system efficiency and scalability and we illustrated this with simple examples. Thus, in addition to application domain issues, it has a number of system related issues such as system recovery, data consistency, etc. The existing data repositories will not functional satisfactorily, new approaches to data validation will have to be developed, and so on. This sea change motivates the research agenda discussed in the next section.

Research Challenges

Monitoring applications are inadequately served by current system software architectures for several reasons [2]. First, there is a scaling problem. Applications may need to support 1,000,000 or more sensors all reporting their state in real time to one or more servers. Few software architectures are capable of dealing with this number of independent yet related reporting entities. Second, monitoring applications are fundamentally distributed and must be spread over multiple computing platforms. Static partitioning of the system will not work because of the high variability in system parameters. The number and location of the computing devices change continuously, the load observed at any one node fluctuates, and the connections and their associated bandwidth cannot be counted on to be stable. Load management and balancing across such a dynamic network environment introduces new challenges that do not yield to previously studied techniques. Furthermore, many of the sensors are mobile and may move over significant distances. This may require reassigning a server to the task of recording a given sensor's input. Note that the problem of heterogeneity is much more complex here because of (a) data sources are widely dispersed and (b) there is no standard data format which could be used by all data sources. This raises problems of data acquisition, source to server mapping, and data conversion and validation.

Current middleware and DBMS systems have a "human active/system passive" model whereby the system sits idly by waiting for a human to initiate a transaction. In contrast, most monitoring applications are better served by a "system active/human passive" architecture, whereby the system is responds to a large number of inputs in real time, alerting a human only if there are conditions that require his attention. This reversal of roles necessitates a complete rethinking of most components of system software.

In addition, most system software has been transactional in nature. It is important to provide precise answers and never lose data, even if response time degrades. In contrast, monitoring applications can discard data if they become overloaded, and must provide real time service to certain sensors, such a fire detectors. This requires a rethinking of the traditional consistency and service guarantees provided by current systems. Furthermore, monitoring applications fundamentally deal with time series information, which is badly served by existing database systems. Monitoring applications must deal with imprecise information and must be able to manage location dependent data correctly and consistently. Questions such as "*Where is Mary right now?*" and "*Where am I at present?*" must be answered with interpolation from a last known location and some notion of an intended route [21].

Data processing in mobile environments must be sensitive to the fact that the computing elements will have to cope with varying degrees of connectivity. In a mobile system, we may have (a) continuously connected operation where the client never loses communication with the server, (b) disconnected operation where the client processes data while it is unreachable by the server, and (c) intermittently connected operation where the connection with the server can never be guaranteed to be available but might be available in special circumstances (e.g., near a beacon). Under (a) the definition of data consistency and correctness are defined in a conventional way but in (b) and (c) these properties will have to redefined. While some work has been done in this context, there is little agreement about what the correct approach should be.

Monitoring applications are very poorly served by existing software solutions. In a tracking application, a user may want to know if a friend is nearby. However, if all the parties are moving and the system bandwidth is limited, it might not be possible to exchange enough information to accurately answer the question. There is no scheme for providing global data consistency when the data is used (read and written) by a large number of mobile users and service providers. For this reason most service providers provide only *read-only* facilities. Instead of producing transactionally correct data, the system has to be able to gracefully produce approximations when resources are scarce. Modern data management tools are not up to this task.

Thus, the management of spatial and temporal data and guaranteeing their availability and consistency in mobility domain is an important research area, which must be promoted. The remainder of this report sketches a specific research agenda that encompasses networking along with DBMS and middleware technology.

Tracking [22], i.e. continuously maintaining in a database the transient location of a moving objects, is an enabling technology for several classes of applications. Some important classes of new applications that will be enabled include location-based services, tourist services, mobile electronic commerce, and digital battlefield. Some existing application classes that will benefit from tracking include transportation and air traffic control, weather forecasting, emergency response, mobile resource management, and mobile workforce.

Tracking enables tailoring the information delivered to the mobile user in order to increase relevancy; for example delivering accurate driving directions, instant coupons to customers nearing a store, or nearest resource information like local restaurants, hospitals, ATM machines, or gas stations.

Tracking is also a fundamental component of other technologies such fly-through visualization (visualized terrain changes continuously with the location of the user), context awareness (location of the user determines the content, format, or timing of information delivered), augmented reality (location of both the viewer and the viewed object determines the types of information delivered to viewer), and cellular communication (knowing the location of subscribers enables the network service provider to dynamically allocate bandwidth).

Given that the database cannot be updated continuously, the research issue is how to accurately maintain the current location of a large number of devices while minimizing the number of updates.

2. Research Agenda

The workshop spent the bulk of the discussion time raising research issues that are requirements for the new world of mobile, context-aware systems. These issues were prioritized with the intent of using them as organizing topics for future meetings. We will summarize the key issues in this section.

2.1 New architectures

The computing fabric is changing. The world of desktop computers connected through robust pipes to high-powered backend servers is beginning to yield to a much more mixed set of

computing devices. People are beginning to rely more and more on PDA's, cell phones and embedded devices (e.g., sensors). Connectivity and planetary-scale computing are issues that did not have to be considered in the past. It is clear that old architectures in which the computing devices and the connections are not going to work in the new environment Thus, one set of unanswered questions relates to the discovery of alternative architectures.

Synchronization

The current state-of-the art in commercial, mobile database technology supports small database systems on each mobile device with the capability to synchronize that mobile database with a centralized database on demand. Most of the major database vendors have such products available today. Architecturally, these are essentially client-server systems that can continue to operate without physical wires. Synchronization can happen whenever a connection (wired or wireless) becomes available.

In some applications, this synchronization granularity is sufficient to preserve data consistency. But, there are many situations for which this is not sufficient. A simple example of this involves a mobile device has just acquired critical information and is running out of battery power. Suppose that it can't reach the central server because it is out of range, but that other mobile devices are within range. Peer-to-peer information exchange would be a reasonable alternative; however, routing the messages in an efficient manner is an interesting problem. Moreover, synchronization in a peer-to-peer setting without heavy-weight constraints like serialization will require new paradigms of information consistency.

Dynamic network topologies

Further, consider the problem of gathering information from a difficult to reach area. One proposal (i.e., smart dust) involves a number of tiny sensors that are sprinkled in that area from the air. The sensors must create their own ad hoc information network to gather and dispatch relevant information to the remote server, which could itself be mobile. This requires a completely new type of sensor network topology and information collection scheme to manage this fluid environment. Thus, more dynamic architectures are needed to support a context-aware mobile data environment.

Context-Aware Processing

When thinking about the architecture required to support mobile data management, there are many key attributes that need to be addressed. One key attribute is that data is everywhere and is changing dynamically. A good example of this is in the area of sensor data management. A sensor continuously gathers information about the environment. Some of this information changes infrequently. An example of this would be a chemical sensor that is sampling the level of a chemical in the air. If the concentration isn't changing, the sensor readings will be very similar over time. But, as soon as the sensor reads a concentration level that deviates significantly from the normal value, the information could be highly critical. Sensors are beginning to possess significant computing power and storage capacity. This capability allows part of the processing to be performed at the device. For example, the sensor itself will be able to determine the importance of a reading and in order for the sensor to achieve this, it must have self-calibrating capability. With this capability a sensor will be able to reprogram itself to

capture different type of information and detect the changes from the previous information contents, which is not an easy task.

As the amount of on-board computing power and storage space continues to grow, the sensors themselves will become more involved in the task of data management. Architectures that can exploit this capability will become more important. Given a wired and/or wireless communication infrastructure with widely varying capabilities, the efficient movement of data becomes a serious data management challenge. Solving this problem for evolving "sensor webs" will require careful use of application-level semantics including dependence on context. Individual sensors will need additional data from other sensors located geographically around them as well as data from other systems to provide them with the needed context. Precious system resources such as bandwidth and power can be saved when sensors are able to filter and aggregate data.

Reliability

The problem of reliability becomes more complex in a mobile environment. Physical vulnerability, comparatively shorter mean-time-between-failure, and hard to detect sensor status make it far more difficult to provide the level of reliability that we expect in a wired setting. Critical data must be pushed to other nodes in the mobile network to ensure its availability. When a node captures what it determines to be critical information, either from sensors or from users interactions, that information needs to be reliably delivered to other places in the network in case the node looses connectivity or crashes [3]. How does a node determine which other nodes to send critical information to? What is the necessary metadata that needs to be captured to describe the critical information? Does the semantics of the information need to be transmitted along with the information?

Airports, skyscrapers, banks, etc., are prone to terrorist activity. Sensors can be located in these places to detect a poisonous gas attack, or a radioactive attack (dirty bomb). When a potential attack is identified, information can be relayed by these sensors to a set of servers that will take necessary remedial steps. The sensor network itself might be vulnerable to compromise, thus, to provide necessary reliability, research in the areas of self-organizing wireless network, reliable and secured data communication, location management, and information modeling is necessary.

Information pedigree will be very important in mobile database management. The automatic maintenance of the pedigree of information is necessary to answer questions such as: where did this information come from, and what were the contributing elements for this information? Data pedigree contains information about when, where, and how data was collected to aid in properly controlling its use. Pedigree information is typically associated with data as a data quality annotation. Specifically, this includes such information as the immediate source of the data (system, version, host, user, etc.), timeliness of the data (when will the data cease being valid), time created/modified, or an indication that some of the relevant systems/databases were not available. Annotations can be used to assess the usefulness for some set of consumers. Developing methods for composing annotations as data passes through multiple stages of processing is a key research issue.

Information Services

The computing fabric continues to improve by providing connectivity to the smallest of devices. Cell phones, PDA's, and sensors will always be online. Computing will largely involve linking widely scattered information services in order to accomplish a given task. The linkages between these services will be established long enough to accomplish some task and then they will be torn down. The same task may involve very different information providers at different points in time. The architecture of information services in a mobile, context-aware data environment will be strikingly different from the architectures that are used in the wired case.

In a traditional database management system, discovery of data is typically not considered to be an issue. The data is stored in predetermined locations using a static relational schema and is queried and modified as required. Web-based data sources introduce the need for data discovery because sites can appear and disappear unpredictably. However, the frequency of this kind of behavior is low enough that technologies like web crawlers will often suffice. In a mobile environment, the availability and the form of relevant data can change minute to minute. The very motion of the participants in and out of range can make locating data sources challenging. It is clearly not practical to expect that mobile devices provide the same kinds of stability guarantees that are present in traditional data processing settings. A totally different approach to data distribution and data discovery is needed.

Once the desired data is discovered, then querying and updating the information is the next activity of the architecture needs to be considered. The mechanisms for query optimization, combining multiple query results, concurrent access, and all of the other aspects of querying of information need to be re-examined from the viewpoint of context-aware, mobile data management.

Similarly, the mechanisms to ensure data integrity and security, as well as the need for alternative correctness criteria other than atomicity (all or nothing) of queries and updates, need to be re-examined. For example, being able to reconcile multiple conflicting reports from several sensors about the same area might be more important than supporting the ACID (Atomicity, Consistency, Isolation, and Durability) transaction model.

In an environment like the Web, understanding the semantics of information can often be a problem. Information sources are autonomous and may choose to represent information in many different ways. In a mobile data environment, participants cannot be known a priori. This also leads to a lack of a priori agreement between data sources about data representations (i.e., schema). Thus, data semantics should be conveyed along with the information. For example, an indicator identifying the ontology that captures the semantics of the information could be exchanged along with the data. Then, the receiving node could parse the information based upon the indicated ontology. Details of how an ontology would be used in mobile ad hoc networks is an open problem.

Computational Services

In a context-aware, mobile data environment, dynamic composition of system components is another key requirement. When the data and processing are spread out in a mobile network, these pieces will need to cooperate to accomplish a given task. Often, when there is little or no

way to control the placement of computations on nodes, a bottom-up approach is required. On the other hand, if the distributed components are all under system control, a top-down approach might work better. In this scheme, the computation as a whole is known, and it is decomposed into pieces that are moved throughout the network to balance the load. An understanding of how these two modes of operation work needs to be investigated.

The need for this capability derives from the knowledge that a single node may not have enough processing capability to perform the required task. This could be due to lack of the required software, lack of processing power, or low battery power or the need for cooperative task (e.g., surveillance of an area) such as collecting samples using Autonomous Underwater Vehicles (AUVs). In this setup a number of AUVs explore a large area of the ocean floor either for searching for missing items or for collecting data about the ocean floor. In order to work correctly, these AUVs must coordinate among themselves. The command for coordination may come from a server on the surface. In this setup, how should a mobile application that needs external services, discover the services, enter into a cooperative arrangement, pass information, and get the results back to solve a specific problem? The solution to this problem involves wireless networking, process discovery, and a new scheme for distributed process synchronization. Thus, the architecture needs to provide the underlying mechanisms that will enable this dynamic composition of services.

2.2 Profiles and Context

Data management decisions require knowledge of how data is likely to be used and how to prioritize this use when there is competition for resources. In a standard DBMS, the database administrator (DBA) handles these concerns. The DBA painstakingly interviews the user community and, armed with this usage information and a sense of business priorities, decides (among other things) how data should be partitioned, what should be indexed, and what should be replicated.

In the world of mobile computing, the user community is potentially very large and is constantly changing. There is no way to create a DBA model for the kinds of applications that we are considering here. Instead, the infrastructure must automatically make data management decisions based on some formal description of user needs and priorities. Profiles capture this information.

General Issues

Previously, profiles have been very simple structures. They are often simply a list of keywords. Such a list implies that the user is interested in anything that contains one or more of the keywords. This fairly course grained view of profiles needs to be refined to include notions of data selection by a sophisticated query language. A profile should also be able to describe concepts such as thresholds (e.g., I am interested in no more than 3 temperature readings) and dependencies (e.g., I am only interested in temperature readings when I already know the relative humidity).

For the purposes of this discussion, we will define a *profile* to be any formal specification of user (or application) interest. A profile describes what data is useful or interesting and creates an ordering among these data items that describes relative importance. A *context* describes

something about a user's current state. This might include the user's location, the time, or the user's progress in a complex workflow. Context may be a parameter to a profile. Thus, a user's interest can be dependent on her state. Profiles are related to Quality of Service (QoS) requirements as are common in networking protocols. Typically, QoS is interpreted in very application specific ways, e.g., frames per second. A profile is much more general and should be able to subsume common QoS requirements.

Profiles and context descriptions need to be processed in a coherent manner. The profile processing mechanism is likely to be the engine room of pervasive data management infrastructure – in much the same way as query processing was the heart of modern relational database systems. However, little technology exists for managing and manipulating large numbers of profiles. The following sections will explore some of the issues that were raised at the workshop on the topic of context and profiles.

Language Issues

It was felt that there are important linguistic issues involved in the expression of profiles. In current systems, profiles are essentially a list of keywords. While this provides some mileage for applications that are fundamentally text-based, this is not nearly powerful enough for the kinds of pervasive applications that the group had in mind. In particular, the monitoring applications described above would need much more sophisticated profiling mechanisms.

First, if the data has structure, the profiling language would need to express patterns over that structure. For example, if the data is available in XML, then it is reasonable to expect that the profile language would be able to make distinctions based on the structure of XML documents. This functionality could be adapted from current XML query languages. However, since networked data sources are typically available in a large number of representations, a single pattern-language will not suffice. Incorporating multiple pattern languages in a single profile language is a fertile area for future research.

An effective profile language must be able to specify priorities among objects of interest so that scarce resources can be devoted to important items first. Is it best to express priorities through numerical values or as a partial order? In the first case, there is opportunity to state a degree (i.e., quantitative) of preference; however, in many cases this might be unnatural and a partial ordering might be simpler and more direct. The group felt that semantic issues such as this needed further investigation. This would involve trying to express some real profiles from a real application.

A more complex notion of profile might include the specification of dependent utilities. In this case, an object might have high value if and only if it can be used to obtain some other object. For example, for an investment analyst seeking a balanced view of some potential stock purchases, a company's annual report may only be useful along with major newspaper articles from the last year about the company. Another example might involve a patient's diagnosis history in which the X-ray lab report is only useful along with the pathology lab report.

Context provides a way to create a more dynamic interest specification. If context describes things about the user's current state, then it is possible to have profiles that describe interests that track this state. A general-purpose context description facility might not be possible; however,

an API for context descriptions might not be too far-fetched. For example, perhaps contexts need to support equivalence testing or some notion of progress. Progress would be defined relative to a goal, and certain contexts would be closer to that goal than others. Exploring this for simple contexts like location would be an interesting research topic.

Context might also describe certain dynamically changing aspects of the environment. For example, in a mobile network, a context might describe other users and machines that might currently be in the vicinity. More importantly, it might describe the services that those users or machines might be able to provide or use. In the general sense, then, a context is any specification of the environment that might direct how processing is to proceed. For peer-to-peer computing and for ad hoc networking this is a crucial aspect of the problem. As these forms of networking become more common, the need for general context support services will also increase.

Processing Issues

Once we have determined an appropriate representation for capturing profiles and contexts, there are many open questions regarding how to efficiently process them. For example, for some tasks, the data management infrastructure would not want to continually make decisions such as what to prefetch based on a collection of millions of individual profiles. Instead it would be much more manageable to create one aggregate profile that represented the collective interests and priorities of the entire user community. This then raises the question of how profiles should be combined. We would need algorithms to merge profiles that fuse predicates and combine utilities. For complex profiles, this can be a difficult task.

If profiles are being used to direct the dissemination of information as in publish/subscribe systems, then there needs to be a way to index very large numbers of profiles so that when a new piece of data comes along, the system can rapidly determine which users to send it to. Indexing data is relatively easy; indexing profiles which are similar to predicates is much more difficult. The mechanism would need to reason about profile (predicate) overlap. In the general case, this is not possible. Thus, issues like this could drive us to limit the expressive power that we allow in our profile language.

As a further example, we might use profiles and contexts to select a subset of objects from a much larger set. This subset could be used to load a shared cache or to synchronize a PDA. Profiles can be thought of as a function that assigns a utility value to all possible subsets. If the problem is to maximize the utility of the subset of objects given a profile, then the subset selection becomes a kind of knapsack problem. Of course, if we allow dependencies in our profile specifications, then the knapsack problem would have to be augmented (precedence constrained knapsack problem), and heuristics would need to be invented to make the processing computationally feasible.

2.3 Stream Processing

The database community has had remarkable success in basic research and resulting transfer of technology for the management of *static* data and for efficient support of queries over this data. New classes of applications such as the monitoring applications described earlier have a strong need for query support, but the environment is totally different [4, 5]. In many applications data arrives in high-speed data streams, and is either processed on the fly or within a

time frame or lost forever. This section outlines research challenges in data management and queries for data streams.

Motivation

Traditional Database Management Systems (DBMS) software is built on the assumption of a *static database*, stored reliably on secondary storage or in-memory, queried and updated at the requests of users. For several emerging application domains, however, data arrives in high-speed data streams, and needs to be processed continuously, without the benefit of the data being stored first on persistent storage. Such continuous data streams arise naturally, for example, in the network installations of large Telecom and Internet service providers where detailed usage information (Call-Detail-Records (CDRs), SNMP/RMON packet-flow data, etc.) from different parts of the underlying network is continuously collected and analyzed for interesting trends. Other applications include monitoring sales transactions in retail chains, ATM and credit card operations in banks, financial tickers, data from sensor networks, etc. In some applications, the data stream can actually be accumulated and archived in a DBMS of a (perhaps, off-site) data warehouse for future analysis, but the volume and speed of arrival makes access to the archived data often prohibitively expensive. Furthermore, the capability to process records *online* (i.e., as the data stream arrives) is crucial for many mission-critical tasks (e.g., telecom fraud detection).

Basics

In a data stream, records are not stored on disk or in-memory, but arrive continuously online [6]. Some fundamental assumptions about stored datasets are violated in the data stream model: A data stream has no finite size, the records arrive online without the user having influence on the order of arrival, and once a record is processed, it has to be explicitly stored; otherwise, it cannot be retrieved again.

Streams can have a much richer semantics than just that of an append-only relation. Data streams could consist of general SQL DDL statements, including updates to existing records, deletions of records, or even more regular updates, such as incrementing the value of an attribute by one, or increasing the value of a record by a delta amount. So far data stream models have concentrated on the arrival of individual records (i.e., streams consisting of SQL INSERT statements); streams with richer semantics are an interesting area of future research.

Streams might have *properties* that hold between subsequent records in the stream. An example of such a property is that records arrive in increasing order of their time stamp, but more complicated properties are possible. If we know that a data stream has a certain property, we might be able to utilize this property for efficient query processing, analogous to the way a sort-merge join between two relations is useful if we know that the relations are stored in sort order on the join attribute. Investigation of properties of data streams is a fruitful area of future research.

Data stream processing can have many different goals, such as *filtering*, *aggregation*, *interpolation*, *correlation*, and *data fusion* (analogous to a join for stored relational data). Since data stream processing is still at its infancy, we believe that early applications will be driven by the technology that is available, however, it was felt that research in data stream processing has the potential to enable a broad class of future applications.

Queries over Data Streams

There are several extensions for querying data streams that go beyond the simple query model for static data.

First, users might not be able to express exactly what they are looking for, and might express instead an intent, goal, or interest in classes of records. The collection of such user-specific data is again captured in a *profile*. The profile might also specify how tolerant the user is to delay in receiving the answer as well as a tolerance to inaccuracy or approximation.

In addition, we can imagine a departure from the typical interaction between database and user, where the user is active and the system passive unless computation is triggered through a user request. In this new way of interacting with the system, the user is passive and the system constantly active as it continuously processes arriving records from the data stream [7].

If the semantics of the query is now user-dependent, we might also extend the notion of a relation about which the query is posed. For example, instead of naming relations or data streams explicitly in the query, it might be useful to use the notion of a "universal query", analogous to the wildcard character "*" in the **FROM** part of an SQL query. Due to resource limitations at the database server, it might be useful to express resource specifications at the time the query is posed, or even physical constraints upon the system components involved in the query. For example, when posing queries over a sensor network, the user might want to restrict the query to be processed only by sensors that have confidence beyond a certain threshold in their data readings.

Uncertainty and Approximation

Uncertainty is inherent in certain types of stream data sources. For example, inherent to data that result from a physical measurement is *uncertainty* regarding the true value of the measured quantity. This uncertainty can properly be described by a *continuous probability distribution function* over the possible measurement values. For example, consider a temperature sensor in your office that reports an estimate *T'* of the current temperature *T*; let this estimate be *T'=68* degrees Fahrenheit (F). Given this measurement, do we believe that the temperature in your office is exactly 68 degrees Fahrenheit? Assuming that the error introduced by the sensor has a Gaussian distribution with a known standard deviation, we can compute the probability that the true temperature *T* lies in a range *[T1, T2]*. In the context of a database application, a user should be able to submit a query that retrieves all temperatures whose true values lie in the range *[T1, T2]* with a given probability *p*.

The query in the previous paragraph shows just one example of uncertainty. Other facets include uncertainty about the actual statement of the query, the result of the query, the context in which the query is posed, or the profile of the user.

Uncertainty is also not the only issue to consider; inaccuracy of data (for example, introduced due to discretization of an attribute value to the last two digits), dilution and pollution of data, repetition and redundancy in the data, as well as incompleteness are different sources of uncertainty that have to be taken into consideration.

We would like to point out that the issue of uncertainty is not necessarily inherent only to data streams; the same problems arise with static data, and it might be a good first step to try to address these problems for static data. Note that the types of uncertainty we are talking about are different from the traditional uncertainty addressed in the literature on probabilistic data models.

Models of computation for data streams usually have resource limitations, for example limited amount of main memory, or limited amount of processing time for each record in the stream. If the amount of memory is limited, approximate query answers will be an inherent part of data stream queries. We need ways to specify acceptable ways of approximating a query answer [8] that are acceptable to the user. Note that the degree of approximation of a query answer does not need to be static, but could vary over time depending on system load and available resources.

Approximation can happen at three different levels:

- *At the level of the data source*. A query might involve a data stream that could be provided from several data service providers at different resolution and quality of service, for example, a low-quality versus high-quality video stream. It should be left to the system to make the right choice of data source selection depending on the user's preferences and resources available.
- *At the level of the query*. Rewriting of resource-intensive queries into simpler queries that are "similar" to the query posed by the user but use much less resources is a fruitful area of future research.
- *At the level of individual records*. Possible techniques include load shedding by dropping records, or changing attribute values through discretization.

Another issue that might be of relevance is *query refinement*. A user might start with an initial query that uses less resources, and then "drill down" using more resource-intensive queries. Models of query refinement for data stream queries are an interesting direction for future research.

Long-running activities

Processing long-running data streams is a challenge [9]. Query plans must adapt to changing selectivities, arrival rates of records, and arrival and departure of other queries in the system. Multi-query optimization of several long-running queries is a major challenge, and data movement between shared sub-queries has to be planned judiciously. An analogy that the workgroup found useful was the notion of a "data basin", that permits records to flow to individual operators as necessary. Eddies [10] might be a first interesting step in this direction.

Second, isolation models for long-running queries over data streams need to be developed. In a system with continuous queries, the usual notion of a transaction is no longer valid, and we need alternative definitions of consistency for the system.

Possible directions include expressing consistency in terms of the state of users. After a system crash, users might have query answers cached that represent the output of the query over a significant time period before the crash. In this case, the system needs to restore the user's data to a consistent state, so that the query can be "continued". Thus recovery techniques might have

to buffer a recent window of the streams and have the capability of selectively repeating parts of the streams. An alternative is for clients to ask the server to send records that "fill holes" and make the client's state consistent.

3. Recommendations

The workshop accomplished two main goals. It first assembled a group of scientists that represented different aspects of the discipline of wireless and pervasive computing (e.g., data management, data mining, sensors, mobile computing, networking). The interaction of these players is an important first step toward making progress in this inherently interdisciplinary field. Second, it began a discussion of a research agenda that could be supported by this disparate group of people. This report summarizes the main areas of this discussion.

The group felt that the two days were well spent, but that the details of the research agenda would need additional activities. Thus, the group felt that a series of follow-on workshops, perhaps focused on some of the topics that were described above would make a lot of sense to help focus and coordinate the efforts of researchers from both academia and industry. It was felt that the interdisciplinary nature of this workshop set it apart from other meetings in this area, and the group recommends that any subsequent meetings retain this characteristic.

4. References

1. Gray, J. and A. Reuter, *Transaction Processing: Concepts and Techniques*. 1992: Morgan Kaufmann.
2. D. Carney, e.a. *Monitoring Streams - A New Class of Data Management Applications*. in *International Conference on Very Large Databases (VLDB)*. 2002. Hong Kong.
3. Agrawal, D. and A.E. Abbadi, *An Efficient and Fault-Tolerant Solution for Distributed Mutual Exclusion.* ACM Transactions on Computing Systems, 1991. 9(1): p. 1-20.
4. J. Chen, D.D., F. Tian, and Y. Wang. *NiagaraCQ: A Scalable Continuous Query System for Internet Databases*. in *ACM Conference on the Management of Data (SIGMOD)*. 2000.
5. S. Madden, M.S., J.M. Hellerstein, and V. Ramen. *Continuously Adaptive Continuous Queries Over Streams*. in *ACM Conference on the Management of Data (SIGMOD)*. 2002. Madison, Wisconsin.
6. Miller, F.W., P.J. Keleher, and S.K. Tripathi. *General Data Streaming*. in *The 19th IEEE Real-Time Systems Symposium*. December 1998.
7. Madden, S. and M.J. Franklin. *Fjording the Stream: An Architecture for Queries over Streaming Sensor Data*. in *Proceedings of the 18th International Conference on Data Engineering (ICDE)*. 2002. San Jose, California, USA.
8. Shasha, Y.Z.a.D. *StatStream: Statistical monitoring of Thousands of Data Streams in Real Time*. in *Int.l Conference on Very Large Databases (VLDB)*. 2002. Hong Kong.
9. Carney, D., et al. *Monitoring Streams: A New Class of Data Management Applications*. in *proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*. 2002. Hong Kong, China.
10. Hellerstein, R.A.a.J. *Eddies: Continuously Adaptive Query Processing*. in *ACM Conference on the Management of Data (SIGMOD)*. 2000.
11. Richard Brooks and S. S. Iyengar, *Multi-Sensor Fusion*. Prentice-Hall PTR, 1998.

12. Ayse Seyedim, Maggie Dunham, and Vijay Kumar, *An Architecture for Location Dependent Query Processing", 4th Int. Workshop on Mobility in Databases and Distributed Systems*, Technical University of Munich, Germany, September 3-7, 2001.
13. Forman, H. George and Zahorjan, J. *The Challenges of Mobile Computing, IEEE Computers*, Vol. 27, No. 4, April 1994.
14. R. R. Brooks and S. S. Iyengar. *Distributed Dynamic Sensor Fusion, Proc. of the Workshop on Foundation of Inf./Decision Fusion*, Oak Ridge National Lab., 1996.
15. D. N. Jayasimha, S. S. Iyengar, and R. L. Kashyap, *Information Integration and Synchronization in Distributed Sensor Networks", IEEE Trans. On Systems, Man, and Cybernetics*. 21, 5(Sep.-Oct.), 1991.
16. www.signal.uu.se/Research/LearningSystems/LS_main.html, *Computerized Learning Systems for Sensor System Data Processing*.
17. http://asapdata.arc.nasa.gov/Sensors.pdf, *Sensor Systems of the NASA Airborne Sc.*
18. Martin David Adams, *Sensor Modeling, Design and Data Processing for Autonomous Navigation. World Scientific Series in Robotics and Intelligent Systems - Vol. 13*.
19. E. Jovanov, D. Raskovic, J. Price, A. Moore, J. Chapman, A. Krishnamurthy, *Patient Monitoring Using Personal Area Networks of Wireless Intelligent Sensors, Biomedical Sciences Instrumentation Vol. 37, Proc. 38th Annual Rocky Mountain Bioengineering Symposium,* April 2001, Copper Mountain, pp. 373-378, 2001.
20. E. Jovanov, D. Raskovic, J. Price, A. Moore, J. Chapman, A. Krishnamurthy, *Patient Monitoring Using Personal Area Networks of Wireless Intelligent Sensors, 38th Annual Rocky Mountain Bioengineering Symposium*, April 2001, Copper Mountain, Colorado.
21. Yasemin Seydim, Margaret Dunham, and Vijay Kumar, *Location Dependent Query Processing", 2nd ACM Int. Workshop on Data Engineering for Wireless and Mobile Access (MOBIDE01)*, Santa Barbara, May 20, 2001.
22. O. Wolfson, *Moving Objects Information Management: The Database Challenge, Proc. of the 5th Workshop on Next Generation Information Technologies and Systems* (NGITS'2002), Caesarea, Israel, June 25-26, 2002. Springer Lecture Notes in Computer Science 2382, pp. 75-89.