CHAPTER 4

Rule-Based Data Aggregation in a Distributed Diagnostic System

Qing Cao, Raghul Gunasekaran, Hairong Qi

Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996

CONTENTS

1. INTRODUCTION

Modern medical diagnostic systems promise the transition from "central" and "hospital-based" diagnosis and healthcare delivery to "distributed" diagnosis and "home" healthcare (D2H2) [1, 2]. One critical challenge in D2H2 is timely detection of potential problems with patients, so that immediate responses can be taken. We envision that wireless medical telemetry devices can be deployed in a home environment where each device is able to record data and transmit it (or a compressed version of the raw data) to a base unit which in turn forwards the data to a PC, mobile phone, or PDA of a family member, staff at a living facility, or an attending physician. In this paper, we consider practical D2H2 systems for assisted living purposes based on wireless sensor networks. There have

ISBN: x-xxxxx-xxx-x

Distributed Diagnosis and Home Healthcare
Edited by E. Y. K. Ng, U. Rajendra Acharya, and Toshiyo Tamura
Volume 3: Pages 1–14

been several successful studies conducted that build wireless telemetry devices based on mote-like platforms [3–5]. Sensor nodes have the advantages of being able to perform non-intrusive data collection and feature low deployment costs. By installing networked sensor nodes for monitoring purposes, we can obtain critical medical data on patients, ranging from their behavioral patterns to critical life conditions including heartbeat rate and blood pressure.

However, while being flexible to deploy, wireless sensor networks face significant communication constraints because they rely on unreliable wireless channels. For example, current sensor networks usually follow the IEEE 802.15.4 standard [7] which typically has a lower bandwidth compared to wireless LANs. It is not only inefficient, but also time-consuming for each sensor node to deliver raw data to the base station for data processing and remote diagnosis. Furthermore, because sensor nodes are typically powered by battery, transmitting excessive data will also significantly decrease system lifetime. Reducing the amount of traffic will both increase the system lifetime and make it more responsive to those events of interest because of reduced end-to-end delivery latency.

To address the problems associated with the communication constraints, in the previous work, data aggregation [8, 9] has commonly been discussed in the context of communication layer protocols. Specifically, data aggregation aims to combine data coming from different sources in the network to eliminate redundancy, minimize the number of transmissions, and save energy. This paradigm shifts the focus of conventional address-centric approaches for networking, such as the Internet, to data-centric approaches typically characterized by constructing routing structures from multiple sources to a single destination, which allows in-network consolidation of redundant data. These approaches have proven to be effective in reducing communication overhead and end-to-end delays.

Different from the existing work, in this paper, we propose a *rule-based data aggregation* scheme that is integrated with routing protocols for real-time monitoring of valuable events. Our rule-based system works together with already deployed applications. When data are generated, this system helps administrators to determine whether the data are *valuable* or not, based on a novel declarative language system to model the temporal and spatial correlations among collected data. In particular, we follow an event-driven approach. That is, in D2H2 systems, the events of interest may be those that are relevant to the monitored patients, such as heartbeat rate, behavioral patterns, and blood pressure. We observe that data collected in D2H2 systems usually correspond to measurement events directly. For example, a deployed system that measures the heartbeat rate may increase a software counter whenever heartbeat events occur. Periodically this counter is sent as collected data to the base station. In addition, we also observe that these event types could be associated with spatial (such as behavioral movement patterns) or temporal (such as heart rate) properties. For example, from the medical perspective, a dynamic range for the heart rate may be considered healthy, where rates above or below this range will be considered indications of heart diseases. In practice, additional considerations may affect the rules for making judgments. For example, these heartbeat readings can be combined with other sensors, such as motion sensors, to distinguish between normal and abnormal cardiovascular activity, e.g., the aforementioned healthy range should be adjusted during deep relaxation or hard exercise.

Specifically, our proposed system consists of two parts to collect valuable information. First, it deploys runtime filtering mechanisms: whenever data are collected, our proposed mechanisms allow filtering collected data in real time through instrumentation of the software that performs data collection. For example, suppose that we want to deploy a patient blood pressure monitoring system. As long as the readings are considered normal, the system does not need to generate any data, and only a periodic signal is needed to inform the user that the current readings fall into the normal blood pressure range. In this way, the generated network traffic is reduced.

In the second part, the proposed system develops a volume-based cross-layer routing protocol to transmit data. In this protocol, different transmission power levels based on the volume of transmitted data are selected. The resulting protocol, XLRP, exploits the application layer information and the capabilities of the physical layer. Our experimental

results show that this protocol can reduce the energy consumption and latency under different settings.

The rest of this paper is organized as follows. Section 2 presents the design and implementation of the rule-based data aggregation framework. Section 3 presents the volume-based cross-layer routing protocol. Section 4 presents the experimental results. Section 5 describes the related work, and conclusions are given in Section 6.

1.1. Rule-Based Data Aggregation Framework

In this section, we describe the rule-based data aggregation framework. This framework primarily applies to *events* collected by a D2H2 system. For example, one common category of events we are concerned with is measurement events, i.e., sensor readings on various parameters ranging from heartbeats to blood pressure. These events are typically associated with timestamps. Our central approach exploits the fact that in a medical context, the sensor readings for such events can either be considered valuable, or considered redundant. For example, suppose that only a certain range of blood pressure readings are of interest to doctors. It will be redundant to transfer the full spectrum of blood pressure readings. Instead, only those valuable readings need to be transferred over the network.

Based on this insight, we chose *constraint graphs* as the theoretical foundation for modeling data. Constraint graphs are directed graphs whose vertices represent variables of interest and edges represent constraints between these variables. They have been used to express graph related analysis such as instruction flows and memory coherence [20, 21]. The novelty of our method lies in that, to the best of our knowledge, we are the first to apply constraint graphs to classify events in D2H2 contexts. In our approach, we assign vertices with events (that generate data) in D2H2 systems, and we focus on a special type of constraint called *orderings*. Specifically, we assign edges with temporal, spatial, and causal orderings between these events.

The major contribution of our approach is that we observe that constraint graphs allow us to express a wide range of data semantics. For example, unusually high heartbeat rates may be considered violations of the temporal orderings of consecutive heartbeat events. By checking if violations have occurred, we can find out whether the data collected are valuable or not. In this sense, constraint graphs allow us to perform filtering to remove redundancy. To achieve this goal, we formalize these constraint graphs rigorously with a declarative language called DataSQL, which is then compiled into distributed runtime components for automatic classification and filtering of collected data. At the base station, original data can be inferred based on the semantics of the rules. We next describe the different components of our approach in greater detail.

1.1.1. Theoretical Foundation of Constraint Graphs

In this section, we describe the theoretical foundation of constraint graphs and how we formulate them. The goal of applying constraint graphs is to allow us to reason about the semantics of collected data readings. Specifically, we focus on data with ordering relations. To model data with constraint graphs, we use vertices to represent events that generate data, and edges representing the orderings between them. Specifically, we classify edges into three categories:

1. **Temporal constraint edges**: These edges reflect the ordering constraints placed in the temporal dimension for measurement events on the same node or on multiple nodes. For example, consecutive heart beat events can be connected by temporal edges whose intervals are associated with timing constraints.
2. **Spatial constraint edges**: These edges reflect the ordering constraints placed in the spatial dimension for measurement events. The specific ordering constraints can be flexible. For example, we could define a metric as the distance between the measurement event to a reference point (referred to as the origin), and use the comparison result to determine edge directions. Obviously, spatial constraints apply only to events occurring on different nodes.

3. **Causal constraint edges**: These edges reflect the dependence of events on the same node or on multiple nodes. Note that the causal ordering will always imply temporal ordering, but not vice versa, so the former is a stronger constraint. If both types of edges appear between two vertices, the temporal constraint edge could be removed.

Figure 1 illustrates these three types of orderings. In the first example, we illustrate temporal constraints through sensors measuring heartbeat activities. If the intervals between consecutive readings satisfy certain requirements, the data are considered valuable and should catch the attention of doctors. Such a requirement, therefore, can be used for data filtering purposes. In the second example, we show spatial constraints through an example with sensors deployed to monitor the movements of appliances, such as doors and fans. As there are distance relations for the placement of appliances in a home environment, we can derive spatial constraints between events. Finally, in the third example, we show causal ordering examples, where two types of sensors are deployed, one for measuring exercise activities, and the second for heartbeat rates. Observe that the first type of event will lead to changes in measurement results in the second type of events. Therefore, they establish casual ordering relations.

1.1.2. Specification of Constraint Graphs

In order to enforce constraints, the system needs the ability to express constraint graph semantics without ambiguity, so that the data can be classified depending on whether they are considered valuable or not. To achieve this objective, we design a novel language to express constraint graphs in networked embedded systems with the following characteristics:

- Ability to represent different types of events, such as sensing, control, communication, and actuation.
- Ability to represent different types of constraints, including temporal, spatial, causal, and their combinations (e.g., the AND and OR relations between edges).
- Ability to correlate events on the same node as well as across multiple nodes.

This set of requirements creates novel challenges for language design. In the following, we describe how we develop DataSQL, a data-oriented language that is based on our earlier work called TraceSQL [18] for debugging purposes, yet has novel features compared to TraceSQL. We first describe briefly the common elements between DataSQL and TraceSQL, followed by the design details of DataSQL and its novel features.

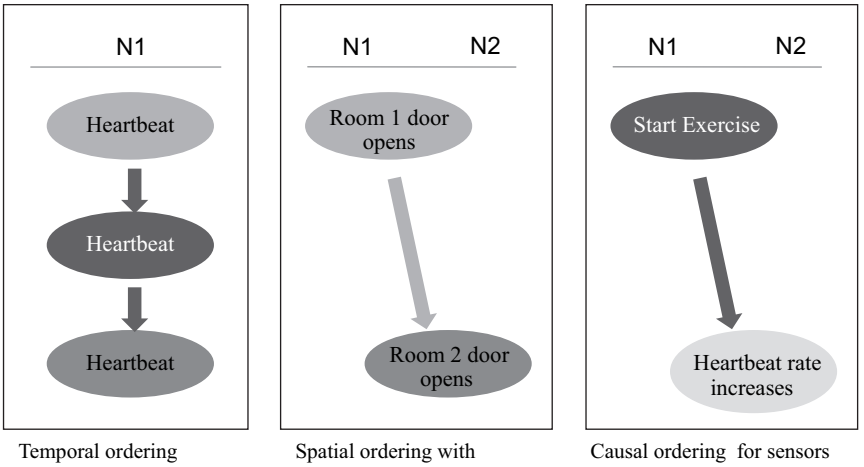

Figure 1. Constraint edge examples.

As the name suggests, DataSQL adopts a declarative language syntax because declarative languages are well known for their ability to express complicated operations with short programs. DataSQL works directly in applications that collect data: it instruments the applications directly to check if the data collected satisfy certain rules. The locations of instrumentation are called *tracepoints*. Specifically, each DataSQL program consists of three parts: tracepoint addresses, tracepoint actions, and optional condition predicates. Formally, a tracepoint declaration has the following grammar:

TRACE {...} **FROM** {...} **EXECUTE** {...} **WHERE** {...}.

To illustrate what DataSQL programs look like, we consider the following application that records the number of certain activities in an application. The specific semantics of the activity may be versatile—it could either be the number of times that a light bulb in a room turns on, or how many times a patient starts to use the air conditioner. We assume that there are already sensor nodes deployed for such monitoring purposes. In this following DataSQL program, we focus on collecting such application independent statistics. For this purpose, we define a variable, *activitycount*, which is shared by two tracepoint statements. The overall goal of this piece of code is to collect how many times a particular activity is triggered. Another virtual tracepoint is triggered once every 100 seconds and collects information for reporting purposes.

```
INTEGER @activitycount = 0;
TRACE activity() FROM user.c EXECUTE {
  @activitycount++;
}
TRACE PERIOD 100s FOR REPEAT EXECUTE {
REPORT @activitycount;
  @activitycount = 0;
}
```

As illustrated in this example, we use the keywords **TRACE** and **FROM** to specify the triggering conditions of tracepoints, which can be further classified into function tracepoints, statement tracepoints, and virtual tracepoints. As their names suggest, function tracepoints insert probes into functions, addressed by the name of the file and the function; statement tracepoints insert probes into source code statements, addressed by the name of the file and the line of code; and virtual tracepoints are triggered by timers, once or periodically, to perform specific actions.

To express constraint graph vertices, we observe that **TRACE** and **FROM** provide very suitable mechanisms. Specifically, for most events or operations, we can either localize them in the original program through the function, or through the line of code and file name. To specify constraints between vertices, the **EXECUTE** keyword is suitable since it allows specifying the types of checking actions the current code will perform, which can be further qualified by the **WHERE** keyword, which starts a condition predicate. Collectively, they allow us to check if constraints are satisfied.

We now describe the differences between DataSQL and TraceSQL. First, different from TraceSQL, DataSQL supports correlation rules between multiple nodes. To achieve this goal, DataSQL extends tracepoint declarations with a fourth type: group tracepoint declarations. Intuitively, group tracepoints allow defining event triggers on a group of nodes rather than any individual node. These tracepoints can be activated when (i) all nodes in one group have satisfied the condition, (ii) at least one node has satisfied the condition, or (iii) some nodes have satisfied the condition.

Second, different from TraceSQL, in DataSQL, a constraint could be checked only after a group of correlated events (at least two) have occurred. To this end, DataSQL adds support for collaborative actions, which have two features. First, to enforce ordering constraints, collaborative actions allow reading/writing variables and invoking functions across nodes. This is necessary, for example, for reading the timestamps and locations of neighbors. For run-time checking, we implement RPC (Remote Procedure Call)-like syntax for this purpose, where one node can remotely access another node to perform these actions. Second, to simplify programming and to provide more powerful semantics,

collaborative actions support group actions on multiple nodes so that multiple actions can be combined together with a shorter piece of code.

As a summarizing example, we illustrate an example of DataSQL for the heartbeat checking example as follows.

```
@TIMESTAMP timestamp[MAX];
INTEGER counter = 0;
TRACE heartbeat() FROM main.c EXECUTE {
  CASE heartbeat():
  timestamp[counter++] = getCurrentTime();
  ORDER TemporalRules(timestamp);
  break;
}
```

This example shows that we use an array to keep the timestamps of consecutive heartbeats. For each heartbeat, we check if the temporal rules have been violated, and if they are, the collected data will be considered valuable and will be reported.

1.1.3. Compilation of Constraint Graphs

In this section, we briefly describe how we compile DataSQL programs into binary applications. Specifically, Figure 2 shows the system architecture of DataSQL.

At the highest level of abstraction, the DataSQL language system is implemented on the LiteOS operating system, which is a Unix-like, multi-threaded operating system developed for controlling wireless sensor networks [19]. The runtime system for DataSQL consists of three modules: the configuration controller for setting up the environment, the tracepoint controller for instrumenting tracepoints, and the action handler for handling triggered actions. The first module, the configuration controller, sets up the operating environment. In this part, it opens files for trace output in the LiteOS file system (LiteFS). LiteFS is a hierarchical file system that provides Unix-like operations on the external flash available on the motes. For MicaZ [6], the size of the flash is 512 Kbytes. To reduce writing operations, we also keep an internal buffer to temporarily store traces.

The second module, the tracepoint instrumentation, is based on dynamic instrumentation, a technique that modifies application binary code at runtime. On MicaZ, we rely on the binary rewriting capability of the Atmega128 processor to modify application binaries. It inserts branch instructions into specified locations of original user application binaries as tracepoint portals. The displaced user application instructions are mirrored in the action handlers. Note that, the displaced instructions cannot contain destinations of other branch instructions in the original application. Otherwise, the correct execution of the original application can no longer be enforced (another branch instruction could

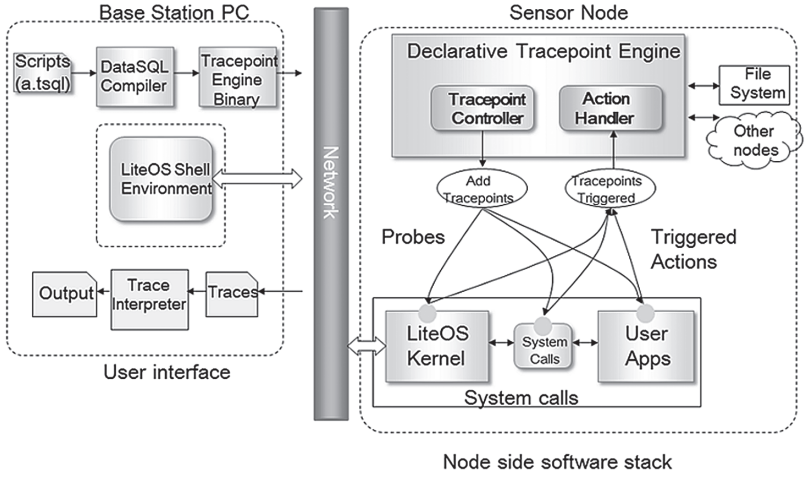

Figure 2. System implementation of DataSQL.

jump to the middle of the tracepoint portal, causing unexpected errors). This limitation is typically not a problem, because the probability that one instruction being the destination of another branch instruction is measured to be very low in benchmark applications.

Finally, for action handlers, we implement comprehensive support for performing actions. The implementation of most actions is straightforward by carrying out operations on memory. For the more complicated ones, we have developed specialized techniques for those purposes. Our compilation results of sample programs demonstrate that we achieved our goals successfully.

In summary, we present a runtime system for DataSQL that relies on the dynamic rewriting of the binary code for the purpose of data filtering. While our implementation is based on the MicaZ platform, we observe that most popular microcontrollers, such as the MSP series, also provide comparable dynamic rewriting support. Therefore, our implementation is inherently portable. Another concern is on the safety of rewriting binary code at deployment time. This design choice is motivated by the observation that we want to apply our proposed approach not only to those projects whose source code are accessible, but also to those whose source code are not available, either because of privacy or security concerns. For such systems, the dynamic rewriting approach has the advantage in that it leads to predictable changes without the need for re-compilation and re-deployment. Based on these reasons we believe that our approach is not only feasible but also cost-effective.

2. VOLUME-BASED CROSS-LAYER ROUTING

Based on DataSQL, valuable events can be retrieved via runtime filtering, and the total amount of data transmitted between the sensor node and the base station can be greatly reduced. When the traffic load is low, existing routing protocols may be effective. However, when certain wireless telemetry devices, e.g., handheld digital camera with specific spectral range, are adopted, instead of one-dimensional data, two-dimensional images are collected from digital cameras. Although computer-aided diagnosis tools have been intensively studied, they normally serve as secondary opinions when doctors diagnose diseases. Under such scenarios, transmitting compressed images is still necessary. Typical applications include, for example, early detection of ulcers using handheld devices for handicapped or elderly patients who lie in bed most of the time [22]. In this section, we discuss a routing algorithm that is versatile enough to cope with transmitting both types of data, i.e., small-volume data filtered by DataSQL, and large-volume two-dimensional images from the sensor nodes to the destination node.

Protocol designs in wireless sensor networks have conventionally resorted to low transmission power levels to save energy and to reduce interference. In our proposed routing algorithm, we select different transmission power levels based on the "volume" of transmitted data. This results in a cross layer routing protocol, referred to as XLRP, that leverages both the application layer information and the capabilities of the physical layer.

As one of the most energy consuming modules in sensor nodes, the radio has been predominantly controlled by the MAC layer, which determines the radios states (active/passive/idle). Our proposed protocol operates independently of the MAC layer. That is, in addition to the functionalities provided by the MAC layer, XLRP further enhances radio management by controlling the radio modules based on application level information. The proposed model differs from conventional cross layer approach in four aspects. First, although XLRP relies on the information from the application and physical layer, it operates independently of the specific layer functionalities. Second, the proposed algorithm supports data transmission at varying power levels. XLRP works on switching transmission power levels based on the volume of the data being transmitted to achieve energy efficiency. Third, the cross layer design saves on energy resources by switching OFF unintended receivers based on the power of the received radio signal. Finally, the design also allows piggy-backing information and unintended receivers extracting information from received messages, avoiding a beacon-based approach and reducing the control overhead of the protocol.

2.1. Overview

Generally speaking, radio modules at sensor nodes are normally programmed to operate at low transmitting power levels to consume less energy and cause less interference, so that extended network lifetime can be achieved. However, resorting to low transmitter power level is not always the most energy efficient strategy, as this will lead to more hops for packet transmission, which is especially true when there is a large amount of data to be transmitted. On the other hand, transmitting at high power levels results in increased transmission radius, and an increase in the number of unintended receivers, resulting in additional energy costs. In the proposed XLRP, we switch transmission power levels based on the volume of the data being transmitted to save energy. This proposed algorithm also saves energy by switching OFF unintended receivers based on the power of the received radio signal.

2.2. The XLRP Protocol Design

The design of the XLRP protocol can be summarized into three phases: the Neighbor Discovery Phase, the Interest Dissemination Phase, and the Data Delivery Phase.

The Neighbor Discovery phase is the initial phase of sensor network setup where nodes broadcast their identity. The first time nodes are activated, they broadcast their identity messages at different transmitting power levels. During this phase nodes learn about their neighbor nodes and the power levels by which the respective nodes can be reached. Nodes retransmit identity messages only when a new node joins the network and at the power level required to reach the corresponding node. The identity message is also broadcasted when there is a considerable change in the energy level of the node compared to the average neighbor energy level. Also, nodes observe a set time delay between transmissions of identity messages, limiting the number of exchanges.

During the Interest Dissemination phase, the interest message is broadcasted at the lowest transmitter power level. Such interest messages are generally short and are flooded throughout the network. An interest message includes an interest ID and the last two hops along the interest dissemination path. Interest IDs help reinforce paths with minimum latency and avoid duplication of interest messages. The last two-hop node information is used for switching transmitter power levels in the data delivery process.

The Data Delivery phase occurs when an event is sensed, and information is routed from the source node to the sink node. Data messages are unicast messages, directed to the neighbor node with matching interest. The next hop node for forwarding data is based on measures discussed in the following section.

2.3. Criteria for Routing Decisions

This section provides the details on the factors in determining the next hop node based on the proposed routing algorithm, XLRP.

2.3.1. Switching Transmission Power Based on Data Volume

Information disseminated through the network could be an interest message, sensor data, a control message, or an acknowledgement/reinforcement message. The messages can be classified by their size in bytes. This application layer information is made available to the network layer, which helps determine the next hop neighbor, and accordingly, how to set transmitter power level. In XLRP, nodes switch to high transmission power levels for large a volume of data and remain in lower power levels for small packet transmission. On one hand, transmitting larger packets at low transmitter power levels results in an increased number of hops and delayed reception. While multiple nodes drain minimal energy by themselves, they will collectively consume more energy. On the other hand, transmitting larger packets implies the sensor node stays at a higher power state for a longer period of time. Switching to higher transmitter levels will reduce the number of hops, leading to a lower number of involved nodes and shorter overall delay. However, the individual node would have drained more energy at a faster rate compared to a

node transmitting at a lower power level. Nevertheless, the collective energy drain would be less, as will be demonstrated through simulations, due to the reduced number of transmitters under the assumption of a reduced number of unintended receivers, which are discussed in detail later.

2.3.2. Signal Strength Based Decision Making

With nodes capable of operating at different power levels and with the knowledge of the received signal strength, our proposed model conceptualizes a back-off mechanism for unintended receiver nodes to switch OFF their radio modules to save energy. As the nodes switch to higher transmitter power levels, the transmission radius increases, therefore, the number of receivers increases. In our design, we select which unintended receivers to be turned OFF based on the received signal strength. For example, if a node wants to talk to a distant node, it would switch to a higher transmitter power level sufficient for the desired node to receive packets (the transmitter node has prior knowledge of the required transmission power level). Although all nodes within the transmission range of the sender would receive the signal, the signal strength received at the node adjacent to the transmitter node would be higher than that received by a farther node. Truly the sender would not have transmitted at a very high power level to reach its adjacent neighbor node, should it be able to reach the node at a lower transmitter power level. Therefore, by setting up a signal strength threshold value, receiver modules that sense signals above the set threshold consider themselves unintentional receivers and correspondingly turn OFF their radio modules. This is an iterative learning process where nodes learn from messages received (during the neighbor discovery phase) and set their threshold values.

To switch OFF the radio module, the node needs to determine the received signal strength (RSSI) and decide on the radio state. If the packet being transmitted is only a few bytes of data, the time taken to make a decision and enforce it would be longer, for the packet would have been received over the period. The minimum time the node would wait before switching OFF the radio, if necessary, would be the time taken to receive the identity message.

2.3.3. Communication Costs

The communication cost parameter is calculated in terms of the energy consumed by the node for a specific radio communication. The cost calculation is based on the energy consumed for the transmission, the energy consumed in switching nodes to the required state, and the resulting energy drain. It has been shown that in low power wireless sensor modules built on 802.15.4/Zigbee compliant transceivers, the energy consumed in switching between radio states is significant. Also, energy consumed in switching to various power levels increases as we step-up in transmitter levels. However, for larger packet traffic, such as hundreds of kilobytes, the radio is operated for a longer period of time, therefore, the energy consumed for switching would be insignificant compared to that for transmission. If a source node needs to transmit a series of packets, it would be a significant energy drain for the node if all the packets are transmitted at the peak transmitter power level. This is limited, in XLRP, by defining a percentage drop in energy and restricting the node to transmit at a lower power level, so that any possible large variations in energy among nodes can be avoided. This implies that node can change transmitter radio state along the network path after a sequence of transmissions.

2.3.4. Energy as a Relative Measure

Energy availability in the node is particularly important in determining what types of operations the node could perform and the overall network lifetime. The node functionality is based on a relative measure of the energy with respect to its peer neighbor nodes. Nodes with less energy will not take part in energy demanding operations, such as switching to higher transmission power levels and acting as relays for multihop communication. From the definition of network lifetime, i.e., the time taken for the first node

in the network to fail, an ideal protocol design would drain energy uniformly among all the nodes, eventually leading to the death of all nodes at the same time or in close intervals.

Our proposed strategy of switching to higher transmission power based on data volume, and switching OFF the radio to save energy, would not guarantee uniform energy drain among the nodes. To achieve uniform energy drain, we consider energy as a relative quantity and define a circle of neighborhood. Specifically, when a node transmits at a higher power level, the nodes immediately surrounding the transmitter would turn OFF their radio if they sense very high radio signals, resulting in different energy consumption rates among neighbors. We can calculate the difference in energy levels as the ratio of the node's energy to the average energy of its neighbors. If the ratio is greater than one, it implies that the node being considered is more resourceful compared to its neighbors, while a less-than-one ratio would push the node to operate in the energy conserving state, i.e., transmitting at lower transmitting power levels and withdrawing itself from network route relay paths. The ratio would also vary based on the neighbor nodes being considered. If the ratio is with respect to the first tier neighbors, meaning nodes that can be reached with the least transmitting power, the energy ratio would vary even for small difference in work load and relatively uniform energy drain. However, considering larger subset of neighbors would average out the high difference, XLRP algorithm restricts its neighborhood region to nodes within its least transmission range.

2.3.5. Node Connectivity

The locations of nodes and their connectivity play a critical role in determining how nodes participate in efficient routing. In scenarios where node density is high, the nodes exhibit more connectivity, and the network load can be effectively shared among the nodes. On the other hand, a network with very limited topology will be partitioned by the death of just a few nodes. Therefore, the metric of connectivity will considerably affect node functionality: if a node finds it has poor connectivity, it will be less likely to switch to high transmitting power, or serve as a relay node, in order to save energy. In addition, the node connectivity parameter can be used to determine the next hop neighbor, and data could be forwarded via nodes which have a higher number of neighbors; in cases of two promising next hop neighbors with equal energy levels and distance to the final destination, the node with higher connectivity would be an ideal route. This also prevents edge nodes (sensor nodes along the boundary of the sensing environment) from running out of battery.

3. EXPERIMENTS AND RESULTS

To demonstrate the effectiveness of the proposed protocol, we use one other protocol, GEAR (Geographic and Energy Aware Routing) [10], for comparison. We focus on demonstrating the effectiveness of transmitting at various power levels, compared to fixed transmitter power level. Because GEAR has the advantage of knowing geographic information, it is helpful to demonstrate the capabilities of XLRP. The performance of the protocol is evaluated in terms of energy consumption and latency.

3.1. Energy Efficiency

To test the energy efficiency of the protocol we create a network of nodes. We vary the data size and calculate the energy consumed for end-to-end data delivery between nodes communicating over a multi-hop. We follow identical network topology for comparing XLRP and GEAR. The simulation results are shown in Figure 3. For small packet sizes of 32 or 48 bytes, the energy consumed is identical for both protocols. However, for increased packet sizes, XLRP has shown to consume less power compared to that of geographic aware routing. The energy savings are attributed to the reduction in the number of transmissions and by switching OFF receivers. We also observe high standard deviation in energy for packet sizes around 256 bytes. This is because of the higher influence

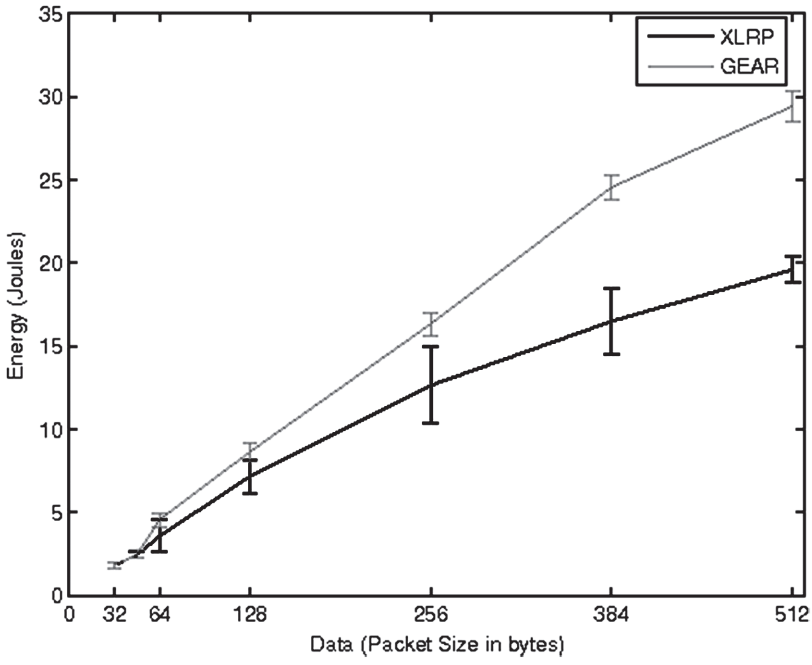

Figure 3. Energy consumption with varying data.

of the communication cost factor in the decision making process of switching to higher transmitter power levels. XLRP also has the benefit of switching between neighbor nodes at different transmitter powers along the interest dissemination path, thereby supporting uniform energy drain along the path. That is, though a single path is reinforced with data, not all the nodes along the path are active as nodes alternate radio state by switching transmitter power levels.

3.2. Reduced Latency

In D2H2 systems, the timely delivery of data is as important as the data itself. The plot in Figure 4 compares the delay in data being delivered across the network from a source node to a destination node. It is evident from the plot that XLRP has fifty percent less delay compared to that of GEAR. This is because of the reduced number of hops along the transmission path. However, for smaller packet sizes the delay in both protocols is the same as it is preferred to remain in lower power transmission mode for lower energy consumption.

4. RELATED WORK

Several previous projects in the literature explored similar themes compared to our work. Among them, we describe related work in three themes: use of sensor nodes for medical purposes, such as wearable sensors for motion analysis, data aggregation algorithms to reduce the amount of traffic to be transferred over the network, and cross-layer routing algorithm design for energy saving purpose. We next describe these three themes separately.

4.1. Sensor Platforms for Medical Purposes

Sensor networks have been proposed for medical purposes in a wide range of research efforts. Of particular interest are those that are developed in wearable platforms for movement measurements [4, 5, 11–13]. Among the representative examples are the Mercury project [12] and the AlarmNet project [5]. In the Mercury project, a specialized type of hardware, SHIMMER, is developed. SHIMMER is powered by Li-poly battery,

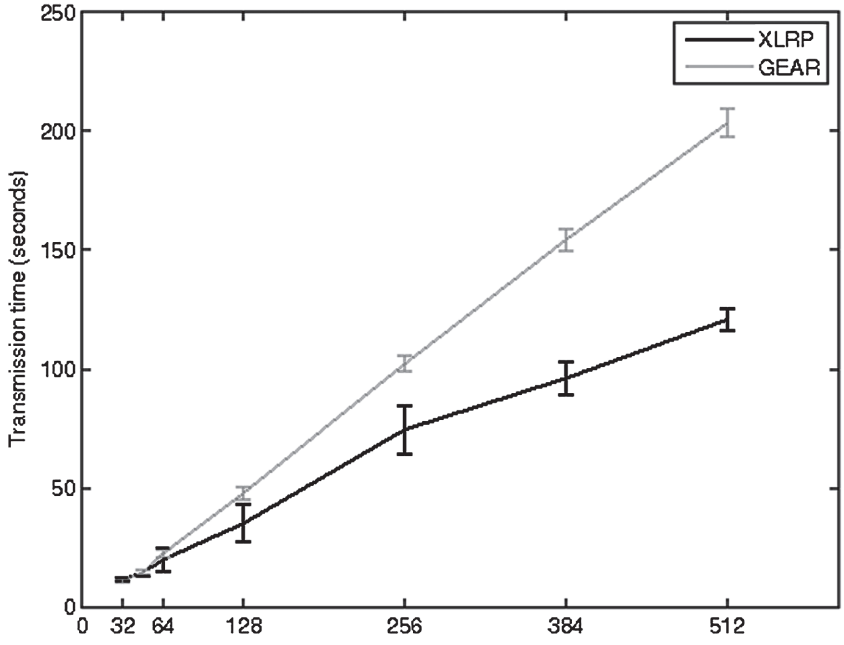

Figure 4. Latency with varying data size.

enabling them to have light weights. Patients being treated for neuromotor disorders, such as Parkinson's Disease, epilepsy, and stroke, wear up to eight nodes in the experimental studies. Instead of transmitting raw data, individual nodes compute descriptive features from the raw signals, and transmit only these processed features to the base station. Compared to Mercury, our proposed work differs in the way data redundancy is reduced: our proposed DataSQL runtime system can apply to legacy systems where source code is no longer available, while in contrast, Mercury requires modifying the deployed software directly. Furthermore, we also present a routing protocol to provide more energy-efficient data delivery, and there is no such support present in the Mercury project.

Another important piece of work is the AlarmNet project. Just like our proposed work, AlarmNet uses MicaZ sensor nodes for the purposes of assisted-living and residential monitoring. It also develops a query protocol for real-time data collection and processing by user interfaces and back-end analysis programs. For example, the circadian activity rhythms of residents can be collected for medical uses. AlarmNet also combines the data measurement with end-to-end communication support. But different from our work that focuses on energy consumption, AlarmNet puts special emphasis on security to protect sensitive medical and operational information in the design of the communication protocols. Finally, AlarmNet differs from our work in the basic software support: our work is based on the LiteOS operating system, while AlarmNet is based on the TinyOS operating system.

4.2. Data Aggregation Techniques

There has been abundant research on energy-efficient, in-network computation of aggregate queries in sensor networks [14–17]. One way to classify the previous work is based on the topology. Some protocols construct routing trees, while others form more free-style paths. Usually, each node is assumed to be generating a stream of sensor readings. Once queries are sent from the base station, each node produces results locally. Along the aggregation topology, local results are combined with redundant information removed or compressed. Optionally, nodes can monitor the link quality or throughput to adjust the level of aggregation and to control the amount of transmitted information.

The DataSQL based approach as presented in this paper is fundamentally complementary with our approach as it performs data filtering at the source nodes, making it

possible for the results to be further aggregated along the tree or other free-form structures. Therefore, we envision the existing data aggregation techniques can be combined with our proposed technique in this paper.

4.3. Cross-Layer Routing Protocol Design

The concept of layered protocol architecture helps lay down a functional framework, where each layer has a well defined functionality and service definition, working independently of the other protocols in the stack. The layered model eases complex network design and supports heterogenously networked systems. However, the layered architecture is best suited for general wired or wireless networks, where the Quality of Service (QoS) and interoperability are of prime importance. However, in wireless sensor networks, the emphasis is towards energy efficiency and optimal usage of network resources. Moreover, sensor networks, being application specific and built to serve a specific purpose, have a protocol design that can be customized towards an application scenario. In line with this fact, cross layer protocols were proposed for wireless sensor networks [23–26], where the layered functionalities are grouped to form a single entity.

Driven by the need for a cross layer design for efficient routing, in Refs. [25] and [26] the MAC layer and the network layer were unified. In Ref. [23], a resource efficient unified protocol was proposed combining the functionalities of all the layers into a single protocol, a complete alternative to traditional layered protocol architecture; the design complexity of the protocol increased as the number of factors influencing a decision making process became large; nevertheless the protocol proved to be efficient, in terms of energy and throughput. In Ref. [25], the lifetime of the sensor network was extended by uniting the functionalities of the network layer onto the MAC layer capabilities. However, a small time sync disturbance in the MAC scheme or the death of an active node would require the re-synchronization of the entire network, and new routes need to be devised; the penalty of over dependence on the MAC layer. In Ref. [26], efficient cross layer routing was achieved by combining the MAC functionality based on the routing decisions made at the network layer, reducing the overhead involved for medium access. The route path is set up by a stateless routing mechanism at the receiver node rather than the sender node, resulting in receiver contention rather than the sender contending to send information. The mechanism is based on geographic routing and is an end-to-end decision making process, making the protocol computationally complex as most protocols designed for sensor networks are based on localized information at the node. CoLaNet [24], a cross layer design of energy efficient wireless sensor networks, worked on collaborating the characteristics and requirements of the application layer with the network layer in forming a route tree and the MAC layer scheduling algorithm. The network tree is based on data gathering from many to one sensors and the MAC used a TDMA-based channel assignment algorithm.

The radio being one of the most energy consuming modules in the sensor node is predominantly controlled by the MAC layer, deciding on the radios ON/OFF cycle. When a sensor network is deployed, the MAC is always active, irrespective of data being sensed, and the upper layers are reactive and act only on sensing events. The active state of MAC is unavoidable in the case of multi-hop sensor networks for effective data communication across the network. Increasing the functionality of MAC by clubbing on network layer functionalities would tend to drain more energy when no event is being sensed. The cross layer routing protocol proposed in this paper, XLRP, operates independent of the MAC layer but enhances power management by controlling the radio modules based on application level information. Moreover, XLRP though relying on the information from the application and physical layer operates independent of the layer specifications and functionalities.

5. CONCLUSIONS

In this chapter, we presented a systematic framework for expressing data in declarative language syntax and delivering data over multi-hop networks with routing protocols for

D2H2 systems. We considered systems that are based on wireless sensor networks. We observed that data are usually closely related to measurement events. Because of the limited bandwidth in communication, it is not only inefficient, but also time-consuming for each sensor node to deliver raw data to the base station for data processing and diagnosis. Therefore, we proposed an efficient rule-based data aggregation scheme, called DataSQL, for distributed event filtering and collection in D2H2 systems. In this system, we represented data semantics and their temporal, spatial, or casual correlations. We then demonstrated how to filter and reconstruct data through this system.

The proposed data representation and filtering system is closely integrated with the proposed volume-based cross-layer routing protocol (XLRP) design. XLRP takes advantage of both the application layer and physical layer information such that the protocol can adapt to both small-size scalar data transmission and large-volume image data transmission. XLRP has several contributions such as tuning transmission power based on the amount of transmitted data, turning off unintended receivers to save energy, and choosing paths by considering network connectivity. Evaluation results showed the effectiveness of XLRP in both energy consumption and delay.

REFERENCES

1. U. R. Acharya, T. Tamura, E. Y. K. Ng, L. C. Min, and J. S. Suri, Eds., "Distributed Diagnosis and Home Healthcare (D2H2)." American Scientific Publishers, Los Angeles, CA, 2010.
2. The First Trans-disciplinary Conference on Distributed Diagnosis and Home Healthcare, 2006.
3. Crossbow, http://www.xbow.com/Home/wHomePage.aspx.
4. CodeBlue project: Wireless Sensors for Medical Care, Harvard University. http://fiji.eecs.harvard.edu/CodeBlue.
5. G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, R. Stoleru, S. Lin, and J. A. Stankovic, *Proc. Transdisc. Conf. Distr. Diag. Home Healthcare (D2H2)* (2006).
6. MicaZ, http://www.xbow.com/Products/productdetails.aspx?sid=164.
7. 802.15.4 standard, http://www.ieee802.org/15/pub/TG4.html.
8. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, *Proc. 5th Symp. Oper. Sys. Des. Impl., (OSDI)* (2002).
9. S. Nath, P. B. Gibbons, S. Seshan, and Z. Andersonlifornia, *ACM Trans. Sensor Networks (TOSN)* 4 (2008).
10. Y. Yu, R. Govindan, and D. Estrin, Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks. UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, 2001.
11. T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, P. Klasnja, K. Koscher, A. LaMarca, J. Lester, J. Landay, L. Legrand, A. Rahimi, A. Rea, and D. Wyatt, *IEEE Pervasive Computing* 7, 32, IEEE Computer Society Press, 2008.
12. K. Lorincz, B. Chen, G. Werner Challen, A. Chowdhury, S. Patel, P. Bonato, and M. Welsh, *Proc. 7th ACM Conf. Embed. Netw. Sensor Sys. (SenSys'09)*.
13. The Assistive Cognition Project, University of Washington. http://www.cs.washington.edu/assistcog.
14. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, *IEEE/ACM Trans. Networking* 11, 1 (2003).
15. J. Considine, F. Li, G. Kollios, and J. Byers, *Proc. 20th Intl. Conf. Data Engineering (ICDE)* (2004).
16. A. Manjhi, S. Nath, and P. B. Gibbons, *Proc. ACM SIGMOD* (2005).
17. N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians, *Proc. 2nd Int. Conf. Embed. Networked Sensor Sys.* (2004).
18. Q. Cao, T. Abdelzaher, J. Stankovic, K. Whitehouse, and L. Luo, *Proc. 6th ACM Conf. Embed. Networked Sensor Sys. (ACM Sensys)* (2008).
19. Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, *Proc. 7th Int. Conf. Infor. Proc. Sensor Networks (ACM/IEEE IPSN)* (2008).
20. H. W. Cain, M. H. Lipasti, and N. Ravi, *Proc. 12th Int. Conf. Parallel Architectures and Compilation Techniques* (2003).
21. K. Chen, S. Malik, and P. Patra, *Proc. High Perf. Computer Arch. (HPCA)* (2008).
22. H. Qi, L. Kong, C. Wang, L. Miao, *J. Medical Systems* (2010).
23. M. C. Vuran, I. F. Akyildiz, and O. B. Akan, *Conf. Infor. Sci. Sys. (CISS '06)* (2006).
24. K. T. Chuang and C. F. Chou, *Proc. Systems Comm.* (2005).
25. J. Wu, P. Havinga, L. van Hoesel, and T. Nieberg, *IEEE Wireless Comm. Mag.* 11 (2004).
26. A. Bahai, P. Skraba, and Hamid K. Aghajan, *Third Int. Conf. Ad-Hoc, Mobile, and Wireless Networks, ADHOC-NOW* (2004).