# Modeling Real-Time Queries in Sensor Networks

Lisa DiPippo, Victor Fay-Wolfe, David Tucker, Kevin L Bryan, Tiegeng Ren,
William Day, Matthew Murphy, Tim Henry
*Computer Science and Statistics, University of Rhode Island*
*{dipippo, wolfe, tucker, bryank, rentg, wday, murphym, henry}@cs.uri.edu*

## 1. Introduction

Many wireless sensor network applications have real-time requirements where sensed data must be delivered to a base station within a deadline and before the data becomes old. For example, a system that monitors temperature in a nuclear power plant would require that the readings be reported to a base station within enough time for a proper response to be made to a rapid increase in the temperature.

A great deal of research exists for scheduling in traditional networks. Our goal is to leverage some of this work to provide scheduling analysis and techniques for sensor networks. This abstract presents a mapping of a well-known real-time model to a particular type of sensor network application in which queries are made at a base station, and results are collected in the network and routed back to the base station.

Some work has been done in the area of real-time scheduling for sensor networks [1,2,3,4,5]. However, in most of these cases, scheduling is ad hoc, and there is no overall model to provide a level of certainty that the timing constraints will be met.

## 2. Model

**Assumptions.** As a starting point, we assume a static sensor network with known locations of motes and known reliability of communication links, such as might be found in a monitoring a nuclear plant or monitoring a relatively stable environment. We assume that there can be multiple base stations, each of which has an internal model of the sensor network (described below). We assume that the queries for sensed data are periodic, long-lived, and that some have timing constraints on both the query and the validity of the data. Regular monitoring of nuclear reactor temperature data is an example of such a query.

**Motes and Network.** Each mote, $m_i$, has (limited) computing resources, a radio, and possibly a sensor. Between each pair of motes, $m_i$ and $m_j$, we define communication reliability, $R_{ij}$. Each mote, $m_i$, has a set of neighbors $N_i$ that represents the set of all nodes with which $m_i$ can communicate with a reliability above a threshold $R_{min}$.

**Queries.** Each query, $Q$, is periodic ($P_Q$) and may have a deadline ($D_Q$) that is different from the period. The size of a data item requested by query $Q$ is denoted $S_Q$.

## 3. Schedulability Mapping and Analysis

In order to leverage the rich scheduling theory that exists, we map the well-known real-time model consisting of end-to-end tasks and resources [6], to our model of a real-time sensor network.

**Tasks.** The passing of data from a sensor mote to the base station is represented as an *end-to-end task (E2E task)*. That is, the E2E task starts from the point where the data was collected, goes through intermediate motes in the route, and ends at the base station on which the query originated. Each sending of data from one mote to another in the route is represented as a *subtask*. Suppose that a mote $m_i$ requires two hops to get its data to the base station, by going through mote $m_j$. The sending of the data from $m_i$ to the base station becomes an E2E task with two subtasks: $m_i \rightarrow m_j$ and $m_j \rightarrow$ *base station*. There are dependencies among the subtasks such that one subtask may not execute until the subtask that sends to it is complete.

**Execution time.** The sending of a piece of data represents the execution time of the task. The time it takes a mote to transmit a unit of data originating at mote $m_i$ is $E_i$. Thus, the execution time for a $m_i$ to send data on behalf of query $Q$ is $E_i * S_Q$, where $S_Q$ is the size of the requested data.

**Period.** The period of the E2E task is equal to the period of the query for which the E2E task is delivering data, $P_q$.

**Data Validity.** A real-time sensor reading will often have a time frame in which it is considered valid, and after which it is not useful. For example, a temperature reading that was taken an hour ago may not be useful if the temperature of the monitored environment changes often. The validity of data requested by query $Q$ is denoted $V_q$

**Deadline.** The deadline for the E2E task is computed based on the deadline of the query, $d_Q$ and the data validity of the data being requested, $V_q$. This computation synthesizes the constraints into a single deadline using a technique such as *just-in-time data delivery* [7]. Each subtask in the E2E task has an *intermediate deadline*. There are several ways to compute intermediate deadlines [6], and any of them could be used. We will use a simple calculation for illustration. Consider the E2E task sending data originating at mote $m_i$ to the base station. The intermediate deadline of the subtask sending from mote $m_k$ to the next mote in the route, $m_l$, is the E2E deadline $d_q$, minus the product of the execution time $E_i$ and the number of remaining hops from $m_l$ to the base station. This simple intermediate deadline assignment allows enough time to transmit the data at each mote in the route.

**Resources.** We model the radios of the motes as shared *resources*. They are shared under the constraint that if two motes attempt to send to a mote $m_i$ at the same time, the two sends will interfere and fail (recall we make no assumptions about underlying MAC protocols). Thus, concurrency control techniques must be employed to create a correct schedule of transmissions. Another constraint imposed on the shared radio resource is the radio buffer in each mote. Each message to be sent by a particular mote must be prioritized, thus imposing a form of blocking on the tasks to be performed by the mote.

To constructing a transmission schedule, we propose a locking technique similar to a read/write locking used in databases. It uses two kinds of locks: a *send lock* (S), which must be obtained on the sending mote; and a *block send lock* (B), which must be obtained for the receiving mote and for all of the receiving mote's neighbors. That is, in order for subtask $T_{i-j}$ to send data from mote $m_i$ to mote $m_j$, the scheduler must allocate an S lock on $m_i$ and B locks on $m_j$ and all of $m_j$'s neighbors. S locks are exclusive in that any mote locked with an S lock may not have any other lock on it. B locks are compatible with each other. For example, consider three nodes, *X*, *Y* and *Z*, where *X* and *Z* are both neighbors of *Y*. If *X* wants to send data to *Y*, the middleware would require an S lock on *X* and B locks on *Y* and *Z*. Thus, *Z* is blocked from sending to any of its neighbors (and thus interfering with *X's* send) since it has a B lock, but it can receive from a neighbor other than *Y*.

## 4. Some Issues to Consider

In order to apply this model and mapping to real sensor networks, there are several important issues that need to be addressed, and we are currently working on these.

**Schedulability Analysis.** Once the sensor network has been mapped to the RTS model of tasks and resources, we would like to apply a schedulability analysis technique like those found in [6]. If we make several very strong assumptions about the sensor network, like know transmission times and reliable links, then we could use a technique such as time-demand-analysis to analyze and schedule the queries. However, these assumptions are not realistic in most sensor network applications. Instead, we are considering probabilistic scheduling techniques that can take into account the unreliability of links and varying transmission times. We must also consider how to account for the sharing of the link resource not only among different nodes sending in the same neighborhood, but also within the radio buffer of the same node. Existing scheduling techniques may consider this to be a type of blocking. We will examine this further.

**Implementation.** We have begun to develop a middleware solution that involves an up front modeling and analysis tool that we have developed for real-time systems (OpenSTARS) [8]. In this solution, queries will be specified in a real-time extension of TinyDB [9] that is based on RTSQL [10]. The user will input the sensor network model into the OpenSTARS tool. Each query will also be input into the tool, as they arise. OpenSTARS will map the sensor network model to the real-time model, analyze it, and compute a schedule (if one can be found) that meets the specified timing constraints. Then TinyDB will distribute the new schedule to those nodes that are affected by the new query.

**Time synchronization.** In order for the computed schedule to work across the sensor network, the motes must be synchronized so that they all have the same understanding of when a time slot in the schedule occurs. There are various algorithms for time synchronization in sensor networks [11, 12, 13] that we are considering.

**Scalability.** With a large sensor field the number of tasks and resources can explode, making computing a schedule infeasible. We are investigating techniques that abstract portions of the sensor network into a "meta node" that can then be analyzed and scheduled as if it were a single node in the network. By taking advantage of characteristics of some sensor network topologies it may be possible for OpenSTARS to decompose its analysis hierarchically and mitigate some of the state explosion. It may be, however, that our middleware will have limits on the sizes of sensor networks that it can schedule.

**References.**


[1] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, T. He, RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks, *Proceedings of the 2002 IEEE Real-Time and Embedded Technology and Applications Symposium*.

[2] S. Li, S. Son, and J. Stankovic. Event Detection Services Using Data Service Middleware in Distributed Sensor Networks. *Proceedings of the 2nd Internnational Workshop on Information Processing in Sensor Networks*, 2003.

[3] T. He, H. Stankovic, C. Lu, T. Abdelzaher, SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks, *Proceedings of the 2003 International Conference on Distributed Systems*, May 2003

[4] Yilmazer C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, T. He, RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks, *Proceedings of the 2002 Real-Time and Embedded Technology and Applications Symposium*, June 2002.

[5] T. Abdelzaher, S. Prabh, R. Kiran, On Real-time Capacity Limits of Multihop Wireless Sensor Networks, *Proceedings of the 2004 Real-Time Systems Symposium, Dec. 2004.*

[6] J. Liu, *Real-Time Systems*, Prentice-Hall, 2000.

[7] Angela Uvarov, Lisa Cingiser DiPippo, Victor Fay Wolfe, Kevin Bryan, Patrick Gadrow, Timothy Henry, Matthew Murphy, Paul R. Work, Louis P. DiPalma, Static Real-Time Data Distribution, *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium 2004*, 502-509.

[8] K. Bryan, T. Ren, J. Zhang, L. DiPippo, V. Fay-Wolfe, The Design of the OpenSTARS Adaptive Analyzer for Real-Time Distributed Systems. *Proceedings of the 2005 Workshop on Parallel and Distributed Real-Time Systems*, April 2005.

[9] S. Madden, M. Franklin, J. Hellerstein, W. Hong, The design of an acquisitional query processor for sensor networks, *Proceedings of the 2003 ACM International Conference on Management of Data*, 2003.

[10] Paul Fortier, Victor Fay Wolfe, and JJ Prichard. Flexible real-time SQL transactions. Proceedings of the 1994 IEEE Real-Time Systems Symposium, Dec. 1994.

[11] G. Ganeriwal, R. Kumar, M.B. Srivastava. Timing-Sync Protocol for Sensor Networks. Sensys 2003.

[12] M. Maroti, B. Kusy, G. Simon, A. Ledeczi. The Flooding Time Synchronization Protocol. Sensys 2004.

[13] J.E. Elson, L. Girod, D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. OSDI 2002.