

DALi: A Communication-Centric Data Abstraction Layer for Energy-Constrained Devices in Mobile Sensor Networks

Christopher M. Sadler and Margaret Martonosi

Department of Electrical Engineering
Princeton University
{csadler, mrm}@princeton.edu

ABSTRACT

Communications in mobile and frequently-disconnected sensor networks are characterized by low-bandwidth radios, unreliable links, and disproportionately high energy costs compared to other system operations. Therefore, we must use as efficiently as possible any periods of connectivity that we have. For this reason, nodes in these networks need mechanisms that organize data to streamline search operations, local computation, and communications.

This work proposes a Data Abstraction Layer (DALi), which is inserted between the application layer and the file system. DALi organizes data with networking in mind to facilitate the development of services for Data Search, Naming, and Reduction that combine to make communications more efficient. From the resulting two-tiered data hierarchy, we develop a multi-layer drill-down search structure that can locate data multiple orders of magnitude faster (and with much lower energy) than simpler data storage structures. Additionally, DALi conserves energy and bandwidth through a mechanism that acknowledges and removes specific data segments from a mobile sensor network. Finally, it seamlessly integrates in a lossless compression algorithm specifically designed for sensor networks to save additional energy.

Categories and Subject Descriptors: C.2.1 [Network Architecture and Design]: Wireless communication; C.3 [Special-Purpose and Application-Based Systems]: Real-time and embedded systems; D.4.3 [File Systems Management]: File Organization; E.5 [Files]: Sorting/Searching

General Terms: Algorithms, Management, Performance

Keywords: Data Search and Storage, Energy Efficient Communications, Mobile Ad Hoc Sensor Networks

1. INTRODUCTION

Mobile and frequently disconnected sensor networks form an interesting subset of the sensor network design space. The target applications vary drastically, from zebra track-

ing [39] to polar monitoring [4]. They have an underlying set of common traits, however, largely based on their harsh operating environments which can make physical node access difficult and which place logistical limits on the size of the deployments.

These systems are characterized by both the severe resource constraints of sensor nodes and by short periods of unreliable, low quality communications over low bandwidth radios. Beyond their sensors, they collect and store data using an ultra-low power microcontroller and energy-efficient, non-volatile memory in an effort to operate for months at a time on a limited energy budget.

Over time, sensor node microcontrollers have become more capable, the amount of storage space has increased, and the energy costs of CPU and storage have decreased. These trends are likely to continue. However, radio transmissions have remained expensive and unreliable and this is unlikely to improve significantly over time. Significant challenges exist regarding the physical energy costs of wireless signal propagation, the difficulties of designing appropriate antennas, and environmental factors which are exacerbated by a constantly changing network topology. Additionally, in a mobile network, nodes may transmit multiple replicated copies of the data to balance latency and energy constraints [32][34]; unnecessarily transmitting the data either to nodes that already have it or to anyone after the sink has received a copy wastes valuable bandwidth and energy.

As a result, a good mobile sensor system must be designed with the data storage and communication infrastructure in mind. Current Flash file systems designed for stationary sensor networks offer clear advantages over raw application management of data, but on their own these systems do not meet our goals. For example, files can grow to the size of the Flash, there is no efficient way to identify particular data items in files, and there is minimal support for compression. However, for the tasks for which they were intended, such as using the Flash efficiently and ensuring data integrity, these file systems perform well. For this reason, we have developed DALi, a Data Abstraction Layer for mobile sensor networks that lies between the application and the file system and provides nodes with Data Search, Naming, and Reduction services.

Data Search is the ability to quickly locate specific data on the node, by name or by value, and summarize it when appropriate. We emphasize search speed because minimizing query response times improves bandwidth efficiency.

Data Naming is the ability to identify specific sections of data in a granularity that can be easily transmitted through

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiSys'07, June 11-14, 2007, San Juan, Puerto Rico, USA.
Copyright 2007 ACM 978-1-59593-614-1/07/0006 ...\$5.00.

the network. Using this, we can build “delete lists” which aim to conserve energy and bandwidth by stopping data (once delivered to the sink) from being unnecessarily transmitted further. We can likewise prevent nodes from transmitting data to other nodes that already have it.

Data Reduction is the ability to shrink the data through in-network computation, data aggregation, or compression. This mechanism conserves energy and bandwidth by reducing the volume of data in the network.

These three functions are interdependent and a truly effective system for mobile sensor networks needs to provide services for all three.

The contributions of this work include:

- We design and develop a prototype of a Data Abstraction Layer (DALi) that restructures data in a way that simplifies communications and uniquely incorporates each of the processes of Data Search, Naming, and Reduction.
- We introduce an efficient way to incorporate “delete lists” into the system, which can reduce energy consumption by multiple orders of magnitude by reducing unnecessary transmissions.
- We demonstrate that our hierarchical data organization serves as the basis for a drill-down search structure that allows for simple, fast sensor data searches on both spatial and temporal data. DALi can effectively search large real-world datasets in the amount of time it takes to send a handful of packets.

The remainder of this paper is structured as follows. Section 2 discusses related work. Section 3 then presents an overview of DALi. Section 4 introduces DALi’s Data Search service and Section 5 introduces its Data Naming and Data Reduction services. Then, Section 6 offers a discussion and evaluation of our work, and Section 7 concludes the paper.

2. RELATED WORK

Sensor network data management systems are typically tailored to stationary, well-connected sensor networks and, therefore, do not attempt to leverage Flash data storage to make opportunistic mobile communications more efficient. Additionally, no existing data organization offers a combination of Data Search, Naming, and Reduction services similar to those offered by DALi.

Proactive relational query processors [11, 13, 19, 38] use queries to activate specific sensors on a node, collect readings for a given period of time, and return the results. They may also compute data summaries and aggregate data from multiple nodes. However, mobile delay-tolerant networks must be reactive because of the long latency required to deliver queries to nodes. Additionally, the often-changing network topology prevents nodes from employing aggregation techniques that rely on data correlation in the network or distributed schemes that rely on specific sensors to execute specialized data reduction algorithms; nodes should primarily rely on data reduction algorithms which can be executed locally. For stationary sensor networks, both more traditional [30] and distributed [24] storage abstractions exist as well, but frequent disconnections and the sparse distribution of nodes prohibit us from using these methods.

Generic Flash file systems [1, 35] and Flash file systems for sensor networks [6, 7, 10] store data in arbitrarily large files, like PC-based file systems. These files can be far larger than mobile nodes can transmit in one communications pe-

riod. The file systems have no mechanisms for identifying smaller segments of the file, which is critical to preventing unnecessary communications related to duplicate copies of data in the network. Our work provides the additional services necessary for proper data location and identification to assist communications.

Our mechanism for subdividing data into smaller chunks is similar to the BitTorrent peer-to-peer file distribution system [5]. However, existing variants for MANETs [23] and sensors [33] are inappropriate for our networks because each node uses a tracker to find out which peers have the file it wants—information that is not likely to be available—and assume good connectivity and reliable multi-hop routes through the network that can move large volumes of data at once. DALi, on the other hand, gradually acquires data over the independent, opportunistic peer-to-peer links characteristic of mobile sensors.

Our data division mechanism also resembles the SPIN routing protocol [12, 15], which, in simulation, breaks data into 500B segments and uniquely names them in an effort to suppress redundant transmissions. However, SPIN requires that each application provide its own naming scheme. DALi, on the other hand, provides a standard two-level naming structure which is applicable across applications and allows names to be merged so that one name can represent much more data. It also provides search and data reduction services that SPIN does not consider.

Other Flash-based sensor storage systems offer useful data structures (e.g., Capsule [20]) and search capabilities (e.g., MicroHash [16]) and are a strong influence for our work. However, they do not attempt to tackle the issue of simplifying communications and they were never intended to transmit more than short data summaries, directly related to their intended use on stationary, connected networks. Additionally, the search algorithms for systems such as MicroHash are only designed to handle data collected from a single node and will not work if the data is not stored in time order. Unordered data is common in systems that store and process data from multiple nodes like DALi does.

Ganesan et. al. use wavelet summarization both on a single node and over groups of nodes to offer multiple granularities of data for transmission and search [9]. Their concept of drill-down queries is similar to ours, but we generate meta-data rather than wavelets since they are not appropriate for answering queries on all types of data. Additionally, we cannot expect to have enough nodes, the proper node topology, or the data correlation necessary for wavelets to be effective across groups of nodes.

Delay-Tolerant Networks [8] may use data mules to gather data files from stationary sensor nodes [26]. However, communications between the sensing nodes and the data mule may be unpredictable, unreliable, and intermittent, especially if the mule’s movements are random. DALi can assist data delivery in these scenarios by dividing files into more communicable segments.

3. DALI ARCHITECTURE

Nodes in sparse, mobile sensor networks will often adapt between either sending all of their data, which minimizes latency, or responding to specific queries, which minimizes communications. If the application wants all nodes to send all data, nodes need intelligent ways to prevent costly unnecessary communications and to improve the efficiency of

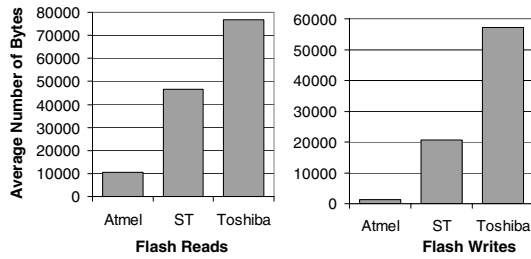


Figure 1: Average number of bytes that a node can read from and write to Flash (assuming page sized operations) for the same amount of energy as transmitting one byte over the XTend radio at 500mW.

necessary ones. If the application just wants answers to specific questions, nodes need to quickly query and summarize a subset of the data that they have collected.

DALi provides these mechanisms on resource constrained sensor nodes and, in the process, provides a semblance of communications reliability in an unreliable realm. These mechanisms should be applicable to both data collected locally and to data received from other nodes, so that a node can preemptively answer data requests for its peers when possible.

An important point in this design discussion is that we are not deeply concerned about storage energy in mobile and frequently disconnected networks. As first pointed out in [21], the energy profile of the Flash has improved significantly over time. Figure 1 compares the energy cost of reading a byte from Flash and writing a byte to Flash with the energy cost of transmitting one byte over the XTend radio [22] at 500mW, as used in the ZebraNet project [39]¹. The comparison covers three Flash modules: a 4Mbit Atmel module [3] used in numerous prior sensor deployments (including ZebraNet), a newer 8Mbit ST module [27] which has many technological similarities to the Atmel Flash and is being incorporated into some newer sensor deployments, and a 1Gbit Toshiba module [31] which has been used in some recent sensor research [16, 20] and is likely to be incorporated into sensor deployments in the near future. For the Toshiba Flash, this translates to writing close to 3.7 million bytes for the energy required to transmit one 64B packet.

These trends suggest that we should use the Flash to our advantage as a way to improve the efficiency of our communications. DALi does this by reorganizing the data as it is written by the application; however, this should be done in a way that keeps the application interface simple.

3.1 DALi Structural Overview

DALi’s architecture is shown in Figure 2 and an abstract view of the data organization is shown in Figure 3. The application creates a file, which we call a virtual file, in which it stores collected sensor data. This structure provides the application with a familiar file interface. In turn, DALi creates multiple physical files which combine to store headers and metadata used to identify and search the data, as well as the data itself (which is stored in a single physical file).

¹All numbers assume page sized operations, although the actual size of the page varies between the modules. We measured the XTend and Atmel energy numbers in prior work [25]. The ST energy numbers are from the datasheet [27] and assume a 1Mbps connection with the microcontroller. The Toshiba energy numbers are from [20].

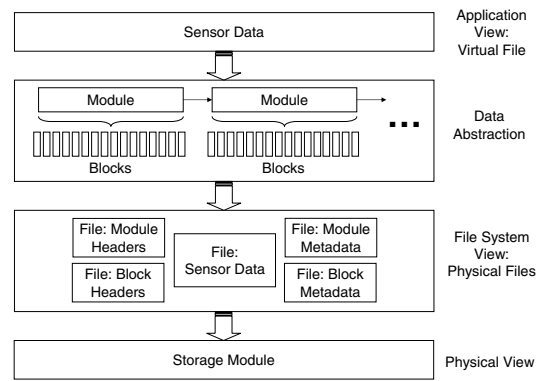


Figure 2: DALi’s Architecture.

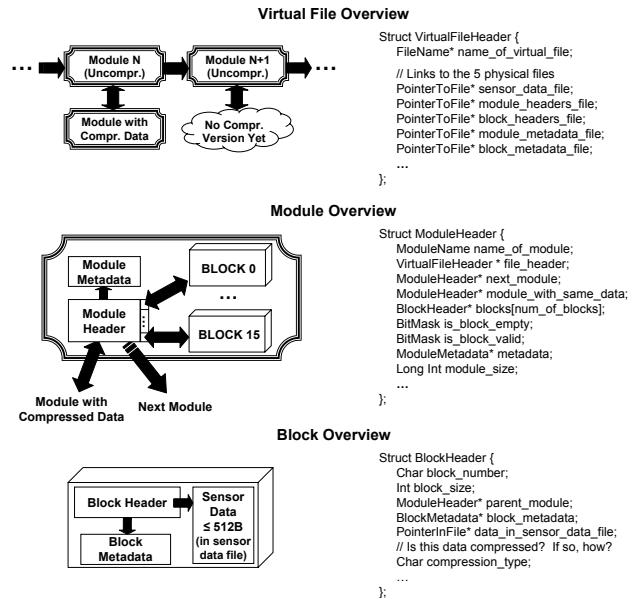


Figure 3: Left: Abstract view of how data is stored in Virtual Files, Modules, and Blocks. Right: Pseudo-code structure definitions of Virtual Files, Modules, and Blocks. Only selected fields are listed for each structure. Although this code depicts a number of memory consuming pointers, the actual implementation is designed in a way that minimizes RAM usage.

DALi breaks the data collected by the application into *Modules*, which are further subdivided into *Blocks*, a hierarchical design which was influenced by the structure of software updates in the Impala Middleware System [18]. Modules are designed to be easy to identify and Blocks are designed to pack data in small, trackable chunks. This structure simplifies the processes of delivering data to the sink and acknowledging that it arrived (see Section 5.1).

Modules consist of groups of 16 Blocks, enabling us to index any Block or groups of Blocks in a Module with a 16 bit mask. DALi uses a Block size of 512B so that the data fits in well with our sensor compression algorithm described in Section 5.2; it fills the Block with sensor readings until it is as close to 512B as possible without exceeding that amount. We define a fixed Block size rather than allowing it to vary based on characteristics of the data such as, for example, grouping all of the data gathered in a given week in a Block, in order to encourage efficient communications. Most data

reduction mechanisms will prove ineffective on tens of bytes of data and it is very difficult to transmit thousands of bytes of data in extremely unreliable networks.

Modules and Blocks are each given headers, which are central to DALi. The organization of these headers, as well as a view of how they are connected to the virtual and physical files, is shown in the pseudo-code on the right hand side of Figure 3. Module headers contain pointer information on where to find the Block headers as well as the overall Module size, the name of the corresponding virtual file, and information on which Blocks are empty or valid. These headers are stored in their own physical file so that they can be read and scanned independently of the data.

Block headers contain pointers to locate the sensor data in the physical sensor data file and the size of the Block, among other things. These headers are also stored in their own physical file; they are stored separately from the Module headers to create an easily scannable data hierarchy. The Module and Block headers hold the pointers to all of the other data on the node. When we refer to “opening” a Module or a Block in this work, we mean that the node is reading the header from Flash into RAM rather than reading the actual sensor data.

Additionally, at both the Module level and the Block level we store metadata summaries in order to assist with searches. Both Module metadata and Block metadata are stored in their own physical files. In Section 4, we will further discuss these summaries, as well as the versatile drill-down search structure that this data organization provides.

3.2 DALi: Module Naming Convention

Each Module name must be unique so that all data can be quickly identified in the network. We use a combination of the 2B (16-bit) node ID of the node generating the data², a 4B time stamp which counts seconds, and a 2B file counter that is simply incremented as virtual files are created and kept constant as Modules are added to the file. This setup is important when we attempt to move acknowledgements through the network, which we describe in Section 5.1.

A time stamp is not truly necessary. It could be replaced with a simple counter and DALi will still work properly. However, as we show in Section 4, time stamps allow for faster, more refined searches so we recommend that they be included in the implementation.

Finally, our decision of how to divide Modules into Blocks fits in well with this naming convention, since for communications purposes, a node can identify any data in the network at a Block granularity with just an 8B Module name and a 2B bit mask.

3.3 Module Name Location and the Time Ordered Structure

DALi requires data structures that can locate data quickly. One of our primary considerations is that nodes must process and store all incoming data, but that they will likely only encounter a small subset of the overall possibilities. Additionally, our resource constraints suggest that we use simple structures to minimize code size and RAM usage.

²For this work, we use a 2B node ID since the typical networks on which DALi will be deployed do not have more than 2^{16} nodes. However, this is easily changed if needed.

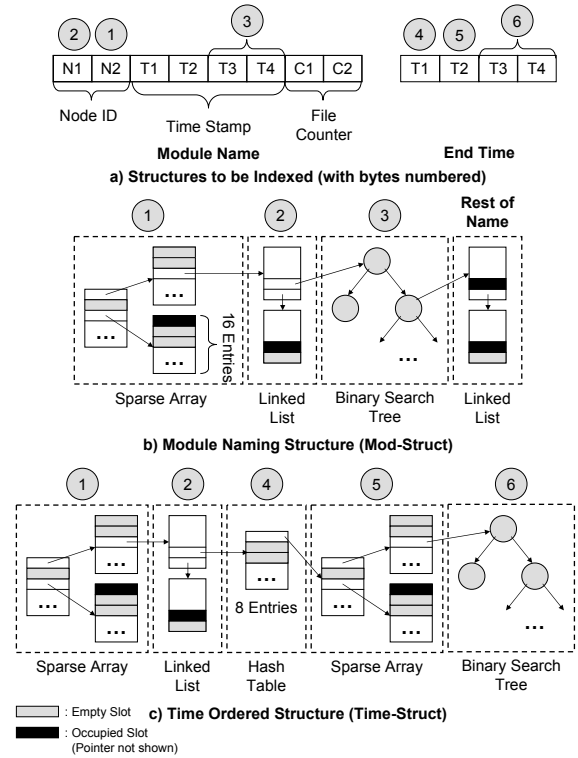


Figure 4: a) Parts of the Module name and the end time are used to index the simple data structures that link together to form b) the Module Naming Structure (Mod-Struct), used for general Module location, and c) the Time Ordered Structure (Time-Struct), used for temporal search. The linked lists in the structures are used to resolve collisions in the structures that precede them.

Although we want to abstract DALi from the physical storage medium, as we design these data structures it is also unwise to ignore the fundamental limitations of the Flash memory modules often found on sensor nodes. For example, Flash memory cannot overwrite data in a page unless the entire page is erased first so it is not possible to simply change pointers on the fly. Implementations using different physical storage media may benefit from different data structures than those discussed here, but the basic goals and principles of those implementations are ultimately the same.

3.3.1 Module Naming Structure

DALi includes naming structures to support both general name location as well as temporal search, which are depicted in Figure 4. The first component, the Module Naming Structure (Mod-Struct), is designed to keep search speed fast while minimizing the number of basic data structures. The top of the structure is a sparse array indexed by the LSB of the node’s ID.

In DALi, a sparse array indexes a single byte of data. It breaks the byte into two 4-bit halves each used to index separate 16-entry tables as shown in Figure 5. The primary table indexes the 4 least significant bits and is created when the sparse array is created. It holds pointers to other tables which are indexed by the rest of the byte. When a slot in the primary table is used for the first time, the node creates the second 16-entry table. This structure enables us to index all

