TinyDB

Presented by Mark Terwilliger
January 27, 2004

Overview

TinyDB is a query processing system for extracting information from a network of TinyOS sensors. Unlike existing solutions for data processing in TinyOS, TinyDB does not require you to write embedded C code for sensors.

My main source

The TinyDB web site: http://telegraph.cs.berkeley.edu/tinydb/index.htm

Web site includes:
- Overview
- Software
- Documentation
- Screenshots
- History
- Further reading
- Contact
Why TinyDB?

Programming Apps is Hard
- Limited power budget
- Lossy, low bandwidth communication
- Require long-lived, zero admin deployments
- Distributed Algorithms
- Limited tools, debugging interfaces

Why TinyDB?

Queries abstract away much of the complexity
- Burden on the database developers
- Users get:
  - Safe, optimizable programs
  - Freedom to think about apps instead of details

How does TinyDB work?

TinyDB provides a simple, SQL-like interface to specify the data you want to extract, along with additional parameters, like the rate at which data should be refreshed -- much as you would pose queries against a traditional database
How does TinyDB work?

- Given a query specifying your data interests, TinyDB collects that data from motes in the environment, filters it, aggregates it together, and routes it out to a PC
- TinyDB does this via power-efficient in-network processing algorithms

How do you use TinyDB?

- You install its TinyOS components onto each mote in your sensor network
- TinyDB provides a simple Java API for writing PC applications that query and extract data from the network
- It also comes with a simple graphical query-builder and result display that uses the API

Some Features of TinyDB

- Metadata management
- High level queries
- Network topology
- Multiple queries
- Incremental deployment
- TASK
Metadata Management

- TinyDB provides a metadata catalog to describe the attributes and commands that are available for querying and invocation in the sensor network.
- Attributes can be sensor readings or internal software/hardware parameters.
- Commands can range from parameter tuning to physical actuations.
- Attributes and commands can be created through the TinySchema components in TinysOS.

High Level Queries

- TinyDB uses a declarative query language that lets you describe the data you want, without requiring you to say how to get it.
- This makes it easier for you to write applications, and helps guarantee that your applications continue to run efficiently as the sensor network changes.

Network Topology

- TinyDB manages the underlying radio network by tracking neighbors, maintaining routing tables, and ensuring that every mote in the network can efficiently and (relatively) reliably deliver its data to the user.
Multiple Queries

- TinyDB allows multiple queries to be run on the same set of motes at the same time.
- Queries can have different sample rates and access different sensor types, and TinyDB efficiently shares work between queries when possible.

Incremental Deployment

- To expand your TinyDB sensor network, you simply download the standard TinyDB code to new motes, and TinyDB does the rest.
- TinyDB motes share queries with each other: when a mote hears a network message for a query that it is not yet running, it automatically asks the sender of that data for a copy of the query, and begins running it.
- No programming or configuration of the new motes is required beyond installing TinyDB.

TASK

- TinyDB is at the heart of the Tiny Application Sensor Kit (TASK), which provides a suite of java based logging, configuration, and visualization tools to assist in sensor network deployment.
- More information about TASK is available at: http://berkeley.intel-research.net/task
Making TinyDB Work (for real)

**Step #1:** Install TinyDB onto each of the motes that you plan to use
- Give each mote a unique ID number, beginning with ID number 0
- Recall to do this, you change into the /apps/TinyDBApp directory and type:
  ```make mica install.0```
- **Note:** replace ‘mica’ with another platform, if necessary, and replace 0 with 1,2,etc. for additional motes

Installing TinyDB on the motes

**Step #1 continued:**
- Connect the mote with ID #0 to the programming/interface board (which should be connected to the PC via serial port/cable)
- Turn on all of the motes

Running the TinyDB GUI

**Step #2:** It may be necessary to compile the TinyDB GUI before you run it. If so, follow the directions in the TinyDB tutorial
- To run the TinyDB GUI, enter the /tools/java directory and type:
  ```java net.tinyos.tinydb.TinyDBMain```
- **Note:** You are executing a Java package so the instruction above is case-sensitive
The TinyDB GUI appears...

Creating a Simple Query
- Clicking on light and nodeid, then pressing the >> button forms the query shown below
- Sample period is time (milliseconds) between samples

Result Window
- Once you click on the Send Query button, a result window will appear:
Observing the Motes During a Query

Once you press the **Send Query** button, the red LEDs of the basestation (node #0) will blink a few times. Soon after that, the red LEDs of the other motes will blink. After a few seconds, the yellow LEDs on all of the motes will blink about once per second.

Observing the results

In a matter of seconds, data will be displayed to the result window:

![Graph showing data](image)

The Where clause

Click on **New Predicate** for more advanced queries:
Aggregate Predicates

Click on AVG (or MIN, MAX, SUM, COUNT) to get an aggregate predicate.

Sensors aggregate their readings with neighbors via efficient in-network processing.

Built-in commands

A pop-up window allows you to send commands to motes, either individually or broadcast.
For example, you can specify a new constant attribute. Once you create the constant, it can be used as part of your queries.

Triggers

A trigger is a query that executes some command (toggle LED or activate sounder) when a result is produced.
Aggregated queries cannot be used by triggers.
TinySQL vs. Standard SQL

TinySQL queries all have the form:

```sql
SELECT select-list
[FROM sensors]  
WHERE where-clause
[GROUP BY gb-list]
[HAVING having-list]
[TRIGGER ACTION command-name[params]]
[EPOCH DURATION integer]
```

- The `FROM` clause must always list exactly one table, entitled `sensors`.
- The `WHERE` and `HAVING` clauses can contain only simple conjunctions over arithmetic comparison operators.

TinySQL vs. Standard SQL

- There is currently no support of sub-SELECT (subqueries)
- There is currently no support for column renaming (SQL's AS construct) in the `gb-list`
- Arithmetic expressions are currently limited to the form `column op column`, where `op` is one of `{+, -, *, /}`

A Simple Java Program to use TinyDB

```java
package net.tinyos.tinydb;
import net.tinyos.tinydb.parser.*;
import java.util.Vector;
import java.io.*;
public class DemoApp implements ResultListener{
  public DemoApp() {
    try {
      TinyDBMain.initMain();  // parse the query next:
      q = SensorQueryer.translateQuery("SELECT light", (byte)1);
      // inject the query, registering ourselves as a listener for result
      System.out.println("Sending query.");
      TinyDBMain.injectQuery( q, this);
    } catch (IOException e) {
      System.out.println("Network error.");
    } catch (ParseException e) {
      System.out.println("Invalid Query.");
    }
  }
  /* ResultListener method called whenever a result arrives */
  public void addResult(QueryResult qr) {
    Vector v = qr.resultVector(); // print the result
    for (int i = 0; i < v.size(); i++) {
      System.out.print("\t" + v.elementAt(i) + \t|"
    }
    System.out.println();
  }
  public static void main(String argv[]) {
    new DemoApp();
  }
  TinyDBQuery q;
}
```
Output from Java DemoApp

From the /tools/java directory, you type

```
java net.tinyos.tinydb.DemoApp
```

The output should look similar to this:

```
Listening for client connections on port 9000
SerialPortIO: initializing
Successfully opened COM1
client connected from localhost.localdomain (127.0.0.1)
Sending query:
  1 |  35 |
  2 |  33 |
  3 |  33 |
  4 |  33 |
  5 |  33 |
  6 |  33 |
  7 |  33 |
...
```

New Features in TinyDB

TinyDB 1.1 was released in September, 2003. New features include:

- **Queries over Flash Memory**: Queries in TinyDB can now log to the on-mote Flash, via the Matchbox filing system. Logged query results can be read fetched via queries as well.

- **Power Management**: When running queries with long sample periods, TinyDB intelligently manages power consumption, allowing the network to last for months at a time.

More New Features of TinyDB

- **Time Synchronization**: Sensors in TinyDB are now time synchronized. This means that aggregate results are guaranteed to have been collected at the same time, and that motes agree on the real-world start and end-time of every epoch.

- **Simulator Support**: TinyDB now runs in the TOSSIM simulator, which can be useful for debugging or demonstration purposes.