# The BikeNet Mobile Sensing System for Cyclist Experience Mapping

S. B. Eisenman,[†] N. D. Lane,[*] E. Miluzzo,[*] R. A. Peterson,[*] G-S. Ahn,[†] A. T. Campbell[*]

[*]Computer Science, Dartmouth College, {campbell,niclane,miluzzo,rapjr}@cs.dartmouth.edu
[†]Electrical Engineering, Columbia University, {ahngang,shane}@ee.columbia.edu

## Abstract

**In this paper, we describe our experiences deploying BikeNet, an extensible mobile sensing system for cyclist experience mapping leveraging opportunistic sensor networking principles and techniques. BikeNet represents a multifaceted sensing system and explores personal, bicycle, and environmental sensing using dynamically role-assigned bike area networking based on customized Moteiv Tmote Invent motes and sensor-enabled Nokia N80 mobile phones. Among bicycles that rendezvouz en route we explore inter-bicycle networking in the following forms: resource sharing; in situ data sharing both directly, and indirectly via static location-specific storage/aggregation entities; bike-to-bike networking via data muling; and real-time and delay-tolerant uploading of data via a number of sensor access points (SAPs) to a networked repository. The repository provides a cyclist with data archiving, retrieval, and visualization services. BikeNet promotes the social networking of the cycling community through the provision of a web portal that facilitates back end sharing of real-time and archived cycling-related data from the repository. Furthermore, we design and build the sensing system to collect the data to feed into this portal. In this paper, we present: a description of the system architecture and implementation; an evaluation of sensing and inferencing that quantifies cyclist performance and the cyclist environment; a study of the potential for data and resource sharing between cyclists en route; a report on networking performance in an environment characterized by bicycle mobility and human unpredictability; as well as describing BikeNet system-user interfaces. Visit [5] to see how the BikeNet system visualizes a user's rides.**

**Categories and Subject Descriptors:** C.2.1 [Network Architecture and Design]: Wireless Communications; J.3 [Life and Medical Sciences]: Health.

**General Terms:** Design, Experimentation, Performance, Reliability.

**Keywords:** Applications, Bicycling, Recreation, System design.

## 1 Introduction

There is substantial interest in the cycling community in collecting data quantifying various aspects of the cycling experience. For example, current and aspiring professional bicyclists are interested in measurements quantifying bicycle, body posture and clothing aerodynamics and are willing to pay $1000 per wind tunnel session to collect and analyze this data [35]. Professionals and serious amateurs are interested in determining their maximum sustainable performance capacity via lactate threshold testing ($150 per test [10]), and inferring aerobic potential via respiration efficiency ($VO_2$) testing ($150 per test [10]). These sophisticated measurements can accurately characterize cyclist fitness, and are integral parts in the successful training regimen of the competitive cyclist. However, these laboratory tests are expensive and disruptive, and target a narrow segment of the bicycling population. In this paper, we focus on developing a system to quantify cycling performance aspects and environmental conditions that the mainstream recreational cyclist can appreciate and afford, akin to the Nike + iPod system [9] for recreational runners.

Existing commercial systems targeting this demographic are designed to measure and report (e.g., via LCD display) simple data such as wheel speed, and provide simple inferencing such as distance traveled and calories burned; these have become increasingly more sophisticated and miniaturized. More recently, several companies (e.g., [18] [11]) have begun to offer products that integrate data from multiple sensors on a single user display, including biometric, advanced cyclo-performance and GPS location data. These range from rather limited $40 devices to very capable $500 devices [11]. Products with a slightly different focus offer integrated hardware and software solutions (e.g., [17]) to help cyclists with pre-ride route planning, and in-ride navigation cues via pre-downloaded maps combined with real-time GPS data. Others (e.g., [20]) offer offline planning software packaged with an online GPS tracking service available via a select set of cellular providers.

While some products allow uploading of logged trip data via a wired connection at the completion of a ride, a limitation of the currently available products is the inability to share data with others (e.g., nearby riders, interested family members) in real-time. Further, real-time performance analysis of locally collected data is often limited to local display of simple statistics like min, max and mean over the entire trip. When the road terrain is highly non-uniform and uphills can last a long time, comparing current speed against a trip-wide average loses significance. Also, even the most sophisticated existing systems focus exclusively on sensing or inferring cyclist performance (power efficiency, cadence, etc.) while ignoring other environmental factors that might be more important to the recreational cyclist.

Even among recreational cyclists there is a spread in the level of interest about various characteristics of a ride. Some are competitive with their friends for the sake of bragging rights, and may want to initiate challenges to set up virtual competitions among geographically separated cyclists a la Nike + iPod [9]; some focus on health-related aspects such as personal fitness; many view bicycling as a time to relax while getting some moderate exercise and are most interested in finding routes that are safe and quiet; others want to simply archive statistics about their rides for later analysis [8]. A system is needed that targets the interests of mainstream recreational cyclists, including characteristics measured or inferred by current commercial systems (e.g., current speed, average speed, total calories burned), but that also measures important environmental attributes of the ride that often dictate whether or to what extent a given route is enjoyable (e.g., amount of automobile traffic, amount of pollution). In the following we outline the system requirements necessary to realize such a system.

- **Cyclist Performance Measurement.** The system should be able to collect and store data about the following baseline cycling performance characteristics: current speed, average speed, distance traveled, calories burned. These are the metrics commonly available in many existing commercial products. In addition, the system should be able to collect and store the following advanced metrics, some of which are available from higher end commercial products: path incline, heart rate, galvanic skin response (a simple indicator of emotional excitement or stress level). The data collection sampling rate and duty cycle should be sufficient to capture a faithful representation of the cyclist's ride. Sensed data should be stamped with time and location metadata.

- **Environment/Experience Mapping.** The system should be able to provide quantitative guidance to cyclists about the that affect the enjoyment factor of a given route. This means collecting and storing data about pollution levels, allergen levels, noise levels, and roughness of the terrain. These measurements, together with data from cyclist performance measurements, can be correlated to create a holistic picture of the cycling experience. Sensed data should be stamped with time and location metadata.

- **Real-time Cooperative/Comparitive Sharing.** Cyclists should be able to set preferences for selective sharing of specified data with other cyclists or other individuals encountered en route. Shared data could be either related to cyclist performance or environment, e.g., as a benchmark of comparison to other riders trying to achieve the best performance on a given hill, or as an indicator to a cyclist seeking to avoid rough steep routes, or roads with higher volumes of automobile traffic. Furthermore, a cyclist may wish to share certain data during a ride with certain remote individuals, e.g., sharing periodic location information with friends or family members tracking the cyclist's progress to get a rough estimate of arrival time.

- **Long Term Performance Trend Analysis.** Collected data should persist beyond the ride on which it is collected. The system should enable the upload of data traces into a personal repository that can be selectively shared with other individuals, or into a public database. The data should be archived in such a way as to facilitate spatio-temporal trend analysis.

- **Data Collection and Local Presentation.** Cyclists should be able to specify what data the system collects, when it is collected (e.g., a timed experiment), where it is collected (e.g., a distance interval experiment), and under what correlated conditions sensor data capture occurs (e.g., increase the sampling rate of the heart rate when the path incline is above a threshold). Further, data should be made available for real time local display to the cyclist (e.g., via an LCD mounted on the handle bar, via a mobile phone display) and the selection of what data appears should be customizable by the system user.

- **Data Query and Remote Presentation.** The system should provide a web-based portal on the back end as a means for querying against and displaying collected data. The portal can also be used as a means to inject queries into the system to request particular bicycling-related data of interest to the back end system user. Finally, the portal can be used as a place to publish/share data with friends/competitors in the same interest group, similar to the Nike + iPod group challenge feature [9].

- **Disconnected Operation.** Because of the mobility inherent in all cycling activities, in situ data sharing requires a wireless networking solution for communications. Also, since some sensors might be embedded in the bicycle, it may be inconvenient to remove possibly many devices for direct wired upload to a centralized data repository. Therefore, it is appropriate to rely on wireless communication for data upload as well. It is expected that cyclists will eventually complete their ride and can use wireless upload protocols to deliver data to a back end repository in a delay-tolerant manner. What about transporting data to the back end while en route? The cyclist may or may not have access to a means of real-time data upload while on the trail (e.g., via cellular phone). Muling [6] between bicycles can increase the probability that sensed data in a sparse and mobile network environment can be delivered in a more timely but still delay tolerant manner.

- **General Purpose/Modular Design.** The system should be extensible beyond the sensors included in the initial implementation, and should provide a mechanism for general purpose tasking via the back end user interface. New sensors should be "plug-and-play" to the system architecture. If a cyclist specifies a given sensor as required by his or her preference profile then the sensor will be included in the data collection. The system should have application beyond cyclist experience mapping, and should provide a general purpose sensing system environmental mapping.

We design and implement a customized system that meets this set of requirements, addressing the limitations of existing systems. Our BikeNet implementation uses sensors embedded or interfaced with the Moteiv Tmote Invent [15] platform as well as camera and microphone sensors available on the Nokia N80 [36] platform. All Tmote Invent and N80 platforms are networked over a common IEEE 802.15.4 short range radio channel. The N80 additionally possesses IEEE 802.11g, BlueTooth and GSM cellular radio interfaces. With these additional radios the N80 can act as a mobile sensor gateway (i.e., a mobile SAP [12]) to support both real-time data uploading to the back end and real time queries from the back end, over the wide area cellular network while the cyclist is en route. This paper also contributes to the development of a general purpose people-centric system for mobile sensing.

With BikeNet, we provide a web portal that allows cyclists to publish quantitative data about themselves and the paths they traverse for real-time or delayed display; also we provide the sensing system to collect this data. In so doing, we hope to provide a useful tool to network members of the cycling community through data of mutual interest. We investigate two data sharing approaches: direct bicycle-to-bicycle sharing via short range radios, or indirect sharing through neutral third-party entities called *rocks*. Rocks are untethered storage and aggregation devices that are placed along roads and trails frequented by cyclists; they store location-specific performance data on per-cyclist and aggregate bases. We have designed and tested a communication protocol by which cyclists interface with rocks.

BikeNet goes beyond characterizing the cyclist performance, and provides a quantitative discription of the environment local to the cyclist. This is done not only to provide additional context to cyclist performance data, but also because environmental data is important in its own right, having a direct impact on the perceived enjoyability of a bicycling experience.

BikeNet allows the user to customize, via a profile of preferences, what data is collected by the system and with whom to share selected data. The profile also indicates how data is to be presented, both locally on the bicycle when *en route* and through BikeNet data access and presentation methods once the data has been delivered to the back end repository.

In addition to sensor data sharing between cyclists, as part of our opportunistic sensing system we investigate resource sharing between two or more bikes traveling within a common local radio range. That is, given that each bicycle is equipped with a subset of all possible sensors, all members of a group can possibly benefit from the most capable and most expensive sensors attached to any member of the peloton through resource sharing. For example, if cyclist *A*'s bicycle has a particulate sensor, but cyclist *B*'s bicycle does not, cyclist *B*'s bicycle can *borrow* the data from the particulate sensor of cyclist *A*'s bicycle, with only a small loss in fidelity. We perform a preliminary experimental feasibility study of resource sharing using a number of sensing resources present on BikeNet's Invent- and N80-based sensing and communication platforms.

BikeNet utilizes an opportunistic networking paradigm, whereby data is shared, muled or uploaded according to the opportunities that arise as a result of the uncontrolled mobility of the cyclists themselves. BikeNet adopts a MetroSense-style [12] architecture to address the challeges of the mobile environment, while optimizing the design for the application domain. This choice allows us to experiment with a more general purpose mobile people-centric sensor network wherein the data collected is not only of use/interest to the cyclists but can be queried against by other system users via a back end web portal. In this context, we consider how data sharing extends to a query- or pull-based paradigm, and how other applications might ride transparently on the back of the system deployed for cyclist experience mapping.

In the following sections, we describe our experiences deploying a sensing system for cyclist experience mapping, leveraging opportunistic sensor networking principles and techniques [12]. We discuss the system architecture and design in Section 2. In Section 3, we detail the sensing and communication equipment in use in the BikeNet prototype implementation, and the BikeNet software system implementation. Section 4 describes an extensive evaluation of our cyclist experience mapping application, including sensing accuracy and inferencing techniques, communication protocol performance, and feasibility results. Related work is discussed is Section 5 before we conclude with a summary and a discussion of future work in Section 6.

## 2 System Architecture and Design

BikeNet is a network characterized by mobile sensing and sparse radio network connectivity. Given these characteristics, and the application requirements for the system, we design the BikeNet system as an instantiation of the MetroSense architecture [12]. The MetroSense architecture is a people-centric paradigm for large-scale sensing at the edge of the Internet based on the use of an opportunistic sensor networking approach. Generally, opportunistic sensor networking leverages mobility-enabled interactions and provides coordination between people-centric mobile sensors, static sensors and edge wireless access nodes (i.e., SAPs) in support of sensing, tasking, and data collection. The BikeNet architecture represents the first realistic application of the MetroSense principles and design, including optimizations for the bicycling application domain. Figure 1 shows a pictorial overview of the BikeNet system, Figure 3 shows a custom-built sensor-enabled bicycle, and Figure 2 shows a logical representation of the bike area network (BAN).

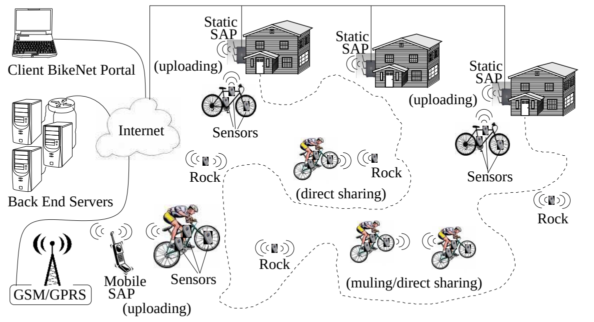


Figure 1: BikeNet System Overview. Sensors collect cyclist and environmental data along the route. Application tasking and sensed data uploading can occur when the sensors come within radio range of a static sensor access point (SAP) or via a mobile SAP along the route or at the route endpoints (see Section 2.1 for the BikeNet hardware architecture description). Sensed data muling and direct sharing can occur when cyclists come within mutual radio range. Indirect sharing can occur when when cyclists and rocks are within mutual radio range. We collect data about the cyclist (heart rate, galvanic skin response), about the cyclist's performance (wheel speed, pedaling cadence, frame tilt, frame lateral tilt, magnetic heading), and about the cyclist's surroundings (sound level, carbon dioxide level, ferromagnetic objects, e.g., cars).

## 2.1 Hardware Architecture

The BikeNet system hardware is organized into three tiers, the back end server tier, the sensor access point (SAP) tier and the sensor tier (including *rocks*). The sensor tier incorporates a number of bicycle-mounted and human-mounted sensor platforms (e.g., Tmote Invent), with possibly multiple sensor types installed per platform, that gather data concerning cycling performance, cyclist health and fitness, and the environment surrounding the cyclists' routes. The sensor platforms mounted to a particular bicycle, along with the sensor platforms mounted to the human riding the particular bicycle, constitute a *bicycle area network* (BAN). Sensor platforms within the BAN have opportunity to communicate via short range radio (e.g., IEEE 802.15.4). The BAN architecture is designed in a modular way such that sensing components can be added or substracted simply according to user preferences (dynamically) set in software. Further, the BikeNet system allows for resource sharing between BANs such that it is not strictly necessary that a given BAN "own" all the sensing components (e.g., expensive or specialized sensors) that it uses. Figure 2 illustrates the design of the BAN in our BikeNet system.

Rocks are deployed along roads and trails and share the same short range radio as BAN sensor platforms. Rocks interact with passing BAN members to facilitate indirect data sharing between cyclists who either can not interact directly or desire to maintain a higher level of anonymity, while still participating in a location/activity-oriented group. Rocks provide storage capacity for small amounts, per cyclist, of location-specific data (e.g., $\leq 1k$B per cyclist) on-location. Rocks can also offer a temporary (due to their limited memory and the priority given to their primary function of location-specific storage) mailbox to improve the probability of successful data routing in a disconnected network.

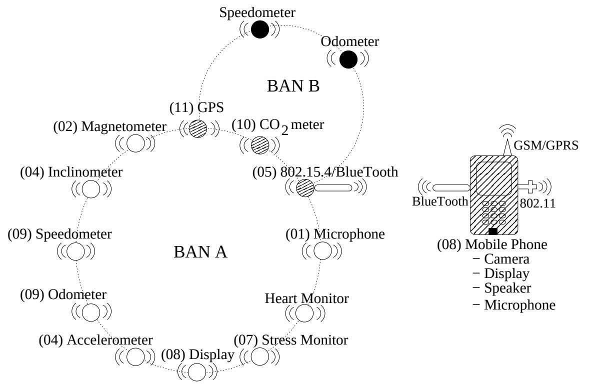


Figure 2: Logical representation of bicycle area networking. Sensors share a common IEEE 802.15.4 channel. BAN B can share the resources of BAN A in order to fulfill the sensing preferences of the user without the resource residing physically on the the BAN B's bicycle. A mobile phone plays a dual architectural role depending on whether its cellular radio is active/connected. If connected to the cellular backend the mobile phone acts as a mobile sensor access point (SAP) facilitating real-time sensing; else it acts as a local member of a BAN engaged in delay tolerant sensing.

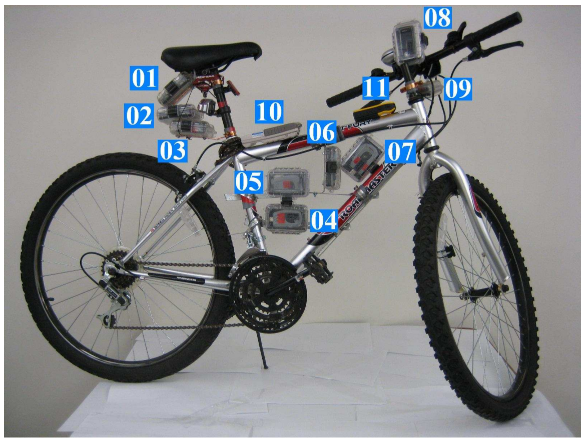


Figure 3: Physical implementation of the BikeNet system. Numbered sensors installed on the bicycle map to the sensor types labeled in the logical representation in Figure 2.

The SAP tier offers high performance, high reliability, and secure gateway access from the sensor tier to the back end servers. This access allows sensed data to flow to the system repositories, and provides a point of command for the architecture to task available sensors with user application requests/queries. When possible, these gateways are symbiotically implemented [12] on the back of existing network infrastructure by plugging a short range radio module into the exising network element (e.g., IEEE 802.11 access point), allowing it to communicate with the sensor tier. SAPs can be static and wired directly to the Internet, or can be mobile and use a data service over a wide area radio access network to provide connectivity to the back end (e.g., mobile phone with GSM/GPRS). For example, the Nokia N80 can use both IEEE 802.11g (as in the CarTel architecture [3]) or GPRS to connect to the back end, and IEEE 802.15.4 to communicate with the sensor tier. We study both tasking and uploading via both static and mobile SAPs in our implementation. SAPs also can be equipped with sensors to provide context and validation for uploaded data [12].

Members of the back end are Ethernet-connected servers equipped with practically unbounded storage and computational power. These provide a number of services to the architecture, some of which are described in Section 2.2.6. In particular, it is to the back end servers that system users connect to submit application requests/queries for execution in the sensor tier, and to retrieve and visualize collected sensed data.

## 2.2 Software Architecture

Figures 4(a), 4(b) and 4(c) shows how the BikeNet software system maps to the three tier hardware architecture, respectively defining the bike sensor platform, SAP and back end software sub-architectures. The default TinyOS software architecture [21] at and below the local communications networking stack is used for communication between the sensor and SAP sub-architectures; the default TCP/IP architecture is used for communication between the back end communications networking stack in the SAP and back end sub-architectures. In the following, we focus on those software components we add to meet the particular BikeNet system requirements outlined in Section 1.

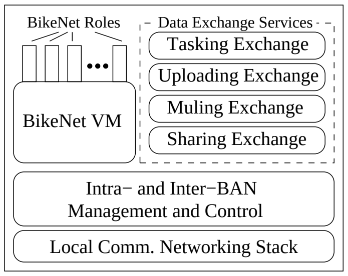
### *2.2.1 BikeNet VM/Role Assignment*

For purposes of modularity the functional requirements within a BAN are divided into *roles*. To streamline role assignment to BAN members all motes are programmed with a BikeNet virtual machine (an application specific virtual machine (ASVM) approach [19]) by default. This virtual machine supports role-specific functions such as query handling and sensing parameterization, and supports the assignment of the BikeNet role(s) themselves to each sensor platform. Assignment is carried out via a resource discovery protocol based on the capabilities of a mote to provide the required sensor readings. Though a physical sensor platform can, depending on sensing capabilities, take on more than one of BikeNet roles, for our current work we allow only one role per sensor platform (e.g., Tmote Invent or other more capable platform) to increase the design, deployment and evaluation parallelism in the BikeNet, and to stress the bike area network (BAN) radio communication protocols. The design will evolve to integrate as many roles as possible on each platform to minimize the number of sensing platforms in a BAN and the associated costs (i.e., monetary, radio congestion).

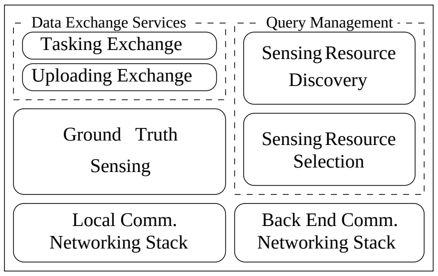
A number of BikeNet roles are defined. The *PedalSensor* and *WheelSensor* roles measure the angular velocity of the pedal and front wheel, respectively. From these the current and average speed, distance traveled, pedaling cadence and gear ratio is measured or inferred. The *TiltSensor* role measures the angle of incline of the bicycle frame with respect to the gravitational force vector, allowing for real time slope calculation and a mapping of the terrain along a cyclist's route. The *LateralTiltSensor* role measures the lateral angle of incline of the bicycle frame. The *CompassSensor* role measures the instantaneous direction of motion of the bicycle with respect to the Earth's magnetic field, allowing for a form of dead reckoning when used in combination with speed and distance information obtained from the WheelSensor role. The *MetalDetector* role measures magnetic deviation from the Earth's magnetic field caused by nearby ferromagnetic metals, allowing inference of the amount of passing automobile traffic. The *SyncSprinkler* role provides a common absolute notion of time and location to all members of the bike area network via periodic short range broadcasts. The *LocalDisplay* role provides a means to display data sensed locally in the bike area network, and data shared from other cyclists via inter-BAN communication (either direct or indirect) to the cyclist. The $CO_2$*Sensor* role measures the carbon dioxide content in the atmosphere surrounding the bicycle, allowing the system to infer whether the cyclist is passing through an urban area (more $CO_2$ from auto exhaust) or a rural area (less $CO_2$ due to plant respiration). The *SoundSensor* role measures the volume of noise in the environment surrounding the cyclist, and is used for voice triggered sensing and audio annotation of a cyclist's ride. The *CameraSensor* role provides triggered capture of an image, or a video clip of specified duration. These captures provide ground truth context to the measurements or inferences provided by other sensor roles. The *GSRSensor* role measure the galvanic skin response of the cyclist, allowing the system to infer the stress level of the cyclist. The *PersonalNode* role does not directly involve sensing, but rather it provides control via short range radio over the other sensing roles, including executing user preferences within the BAN (e.g., required sensors, sensing parameterization), and signaling the start and stop of a cycling trip. Each cyclist necessarily possesses a PersonalNode, but all other roles are optional, depending on the sensing preferences of the cyclist (reference the role assignment implementation in Section 3.2.2). In the rest of the paper, unless the distinction is necessary, we treat the role and the platform to which the role is assigned as synonymous.

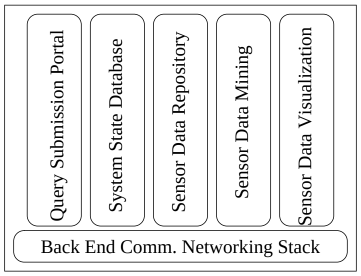
### *2.2.2 Intra-BAN and Inter-BAN Management*

**Localization and Synchronization.** The SyncSprinkler exists to provide, via a broadcast within the BAN, periodic samples of the instantaneous absolute time and location obtained from a GPS unit. Besides GPS, the SyncSprinkler has other potential implementations that may have lower cost and good accuracy. In Section 4.3, we evaluate the comparative accuracy of using the combination of a magnetometer reading from a mote serving the CompassSensor role, and distance traveled as



(a) Bike sensor platform architecture.


Data Exchange Services
Query Management
Tasking Exchange
Uploading Exchange
Sensing Resource Discovery
Ground Truth Sensing
Sensing Resource Selection
Local Comm. Networking Stack
Back End Comm. Networking Stack
(b) Sensor access point (SAP) architecture.



(c) Back end architecture.

Figure 4

inferred from a mote serving the WheelSensor role, assuming a known starting location. Time synchronization is most important among motes in the same BAN. To achieve this, a simple beacon-based approach is used to maintain a consistent view of relative time. When a BAN uses a non-GPS implementation of the SyncSprinkler, both the estimated location and estimated time are corrected any time a BAN member overhears a GPS-based SyncSprinkler broadcast message from another BAN.

**Event Triggered Sensing vs. Continuous Sensing.** Sensing is set up to occur either continuously or only when triggered by other events. In the continuous case, the user preferences executed by the PersonalNode parameterize the sensing capture (e.g., sampling rate, duration, local processing functions) that takes effect immediately upon receiving the *start* message (see Section 3.2.3 for the control protocol) , indicating the start of the ride, from the BAN's PersonalNode. Continuous sensing in BikeNet is appropriate for roles such as the TiltSensor where terrain mapping should be continuous. On the other hand, some sensing operations may be too expensive for the platform to do continuously, or may not have meaning except under certain contexts. As an example of the latter case, if the user preference is to capture wheel speed when going down hill, then sampling by the WheelSensor should only take place when the TiltSensor registers a negative slope. Alternatively, the user may wish to activate, or increase the sample rate of, the heart monitor when going uphill. Triggers can be arbitrarily complex, requiring thresholds, etc., on any number of sensing roles, time or location. Triggers, like sensing parameterization, are defined by user profiles executed by the PersonalNode that specify the conditions under which sensing should occur. Both continuous and triggered sensing are alternatively set up by system user queries that enter the BAN through a static or mobile SAP. As these queries enter the BAN through the PersonalNode (sensor roles are not tasked directly from the SAP), the origin of the continuous or triggered sensing request is transparent to the execution of the sensing request on the sensors themselves. Both continuous and triggered sensing requests can be embedded in resources that are either native to the PersonalNode's BAN (i.e., associated with the PersonNode), or are shared from another BAN.

**Resource Sharing.** The sensor roles that are necessary to collect the data preferred by the user may not be available within the BAN. In such cases, BikeNet allows for the sharing of resources (e.g., sensors, displays, long-range radio, a speaker, large storage) between BANs, where resources are said to be native to one BAN and borrowed by another; a GPS beacon can be viewed as a form of always-on sharing since the beacons are broadcasted. Resource sharing also takes places between BANs and SAPs, where for example a BAN might share a mobile SAP's camera or microphone). While a BAN most often consists of the sensor platforms mounted on a single rider/cyclist pair, and thus the BAN might often be equipped to meet all the sensing roles required by the rider, resource sharing is helpful in the case that the rider *prefers* data from more sensors than he or she *requires*. Resource sharing is also useful in the case of queries that enter the BAN via mobile or static SAPs after the start of the ride, and that ask for data from sensing roles that are not native to the BAN. In such cases, the requested sensors might be found in other BANs along the route (e.g., riding with a friend, participating in a cycling group, competing in a race). Resource sharing can be continuous or triggered.

### *2.2.3 Data Exchange Services*

Four types of data exchange occur with BikeNet sensor platforms: tasking exchange, uploading exchange, muling exchange and sharing exchange. The tasking and uploading data exchanges take place between sensor platforms and sensor access points. The muling and sharing data exchanges take place only between sensor platforms. In the BikeNet tasking exchange, a SAP interacts with available sensor platforms to first instantiate a PersonalNode programmed with a cyclist's BAN preference profile. Based on this profile, the PersonalNode assembles a BAN by tasking other available sensor platforms with the required sensing roles as discussed in Section 2.2.1. Aside from this BAN bootstrapping, the tasking exchange also includes the handling of user queries/requests for data by back end system users, received via the SAP. The PersonalNode responds to these queries by invoking the necessary continuous or triggered sensing (Section 2.2.2) within its BAN, possibly sharing out-of-BAN resources (Section 2.2.2) to complete the task.

In the uploading exchange, when a BAN comes within the radio range of a mobile or static SAP, the sensor platforms composing the BAN attempt to upload sensed data to the back end data repository.

In the muling exchange, sensed data is transferred between sensor platforms outside of the radio range of either a mobile or static SAP. The aim is to increase the probability of timely delivery of data as required by some application features (e.g., cyclist tracking). In the BikeNet system, data muling is done between native members of the same BAN to maximally increase the delivery probability. The reasoning is that native (i.e., not shared from another BAN) members of the same BAN are likely to be affixed to the bicycle and these platforms are likely to all be within range of a SAP for uploading or none of them will be in range. Generally, muling can be reliable or unreliable, and can use replication. With replication the sensor platform asks other nodes to mule *r* copies of each data record, and keeps the original to upload itself (*r* is termed the replication degree). In the BikeNet system we investigate replicated reliable muling.

There are two types of sharing exchange, direct sharing and indirect sharing. Direct sharing is a push-oriented exchange where data transfer is done BAN-to-BAN. Any authorization, authentication, or encryption is handled directly between the two communicating BANs. Indirect sharing is done with a two phase exchange, first BAN-to-rock and then rock-to-BAN to preserve anonymity of the data sharers. Data insertion and retrieval to/from the rock is authenticated and protected by access control to allow controlled sharing of specified data between specified cyclists. Aggregate data across all cyclists who use a given rock is available to all cyclists in the group (where the notion of group is minimally that of cyclists who ride by a certain location).

### *2.2.4 Ground Truth Sensing*

In the BikeNet architecture, SAPs are equipped with certain sensors to provide *ground truth* measurements. In this context of the BikeNet sensing system, ground truth refers to a trusted, high fidelity, always accessible stream of data. One use of ground truth data is as a filter applied to data uploaded from a sensor to a SAP before the data is passed to the back end repository. The ground truth filter can be applied to validate or invalidate uploaded data when the uploaded data samples and ground truth data samples have a high expected correlation (e.g., sampled at the same location, sampled at the same time, samples triggered by the same set of circumstances). Ground truth sensing can also be used to satisfy queries coming from a back end system user that require only a coarse view of the geographaphical extent of the deployed BikeNet SAP tier. A final use of ground truth data is to satisfy queries coming from a BAN in the radio range of a SAP. In this case the BAN can ask for readings from the SAP's ground truth sensors as part of a self-calibration routine or in the course of the resource sharing routine described in Section 2.2.2.

### *2.2.5 Query Management*

The query management component on the SAP handles queries both from the back end system user, and from the PersonalNode of a BAN; it invokes a sensing resource discovery protocol to determine what sensing resources are available to meet the sensing request (e.g., either on-SAP in the case of ground truth sensors [12], or in a BAN within radio range). A sensing resource selection subroutine decides which resources will be tasked in order to satisfy the request and invokes the tasking subprocess to execute the necessary request (i.e., a simple function call if the resource is on-SAP, or via the tasking exchange (Section 2.2.3) is the resource is in a nearby BAN. In the BikeNet implementation, we have experimented with handling queries to the SAP both for ground truth data from the BAN and from the back end. In particular, using a cellular phone as a mobile SAP, we have experimented both with event triggered capture of images, sound and videos requested by the BAN (e.g., an audio annotated ride); and with direct request from the back end BikeNet web portal for image, sound and video samples.

### *2.2.6 Back End Services*

**Query Submission Portal.** The BikeNet back end includes a web portal containing a graphical representation of the static and mobile SAP deployments, and a listing of which BANs are currently accessible through these SAPs. Using a simple mouse click the user can select the SAP or BAN of interest and assemble a query to submit to the query manager component of that SAP using a collection of pull down menus. A final mouse click transmits the query over the Ethernet directly to the selected static SAP or over a wide area wireless network to the selected mobile SAP.

**Sensor Data Storage, Mining and Visualization.** The sensor data repository provides a location for the long term storage of cyclist experience data on a per-cyclist basis and also provides a convenient location for the aggregration of all long term trace data for all participating cyclists. Access to particular data is a matter of a the policy that each cyclist registers with the repository (or a separate access control entity). The sensor data mining component provides a set of standard statistical functions and reusable calculations/ data transformations that a user (e.g., cyclist) can invoke to control the retrieval and presentation of data. For BikeNet we use a number of data interpretation and inferencing tools and techniques, including scatter plots to look for data correlation, Fast Fourier Transforms to look for periodicity, running averages to smooth data to look for trends, and interpolation to align samples according to distance. Particular examples include a method to filter spurious vibration data from the TiltSensor when calculating the slope of the cyclists path, infer gear ratio by combining readings from the WheelSensor and the PedalSensor, and (with knowledge of the starting location) estimate current location without GPS by combining reading from a WheelSensor and CompassSensor. Further, we have develop scripts to transform data values into *BikeView* visualizations. With BikeView (see [5]) we present summarized collected data sorted by user, and sorted by ride within each user account (akin to the presentation of "My Runs" on the Nike + iPod web page [9]). Detailed sensed data can be obtained by simple mouse hovers and clicks over the graphic reprentations of different rides. Back end sharing between users is facilitated by dynamic creation of group pages that are visible to all users in the group and to which all group members can publish data.

**System State Database.** The system state database contains both static state information (e.g., Tmote Invent physical address and sensing capabilities, PersonalNode human custodian information) and dynamic state information (e.g., last known position of a mobile SAPs, and BANs, SAP load average) about elements in the network. In particular, the system state database tracks which BANs are currently in radio range of mobile and static SAPs. This facilitates proper query routing from the back end query submission portal to particular BANs, for BAN-specific user queries. The information is also valuable more generally for debugging and management of the network.

## 3 System Implementation

We implement a complete system in line with the application domain requirements, and according to the architecture presented in Section 2. We build five fully equipped bicycles, implement all of the aforementioned sensing roles using Tmote Invent and N80 platforms, build a number of static and mobile SAPs, and implement a functional back end web portal offering query submission and data retrieval services. The following presents the implementation details of the various hardware and software components.

## 3.1 Implementation Hardware

BikeNet includes a number of bicycles, each outfitted with any of a suite of mote-interfaced sensing devices connected in a bike area network using the short-range radio (CC2420) of the Moteiv Tmote Invent [15]. The Tmote Invents are programmed with TinyOS [21], including the Boomerang package [15]. Untethered devices called rocks communicate via short-range radio with nearby bike area networks, provide storage and aggregation of location-specific sensor data, and facilitate the sharing of information between cyclists. Static and mobile SAPs act as Internet gateways, providing a point for the tasking of the motes in the bike area network, and an entry point for the collection of sensed data to back end data repositories. In the following, we describe the SAPs, bike area network components and rocks in more detail, and include a discussion of ruggedization of the sensing hardware and hardware used to calibrate/validate BikeNet sensor measurements.

### *3.1.1 Ruggedizing the Hardware*

Because of the outdoor nature of the BikeNet testbed we take steps to protect the sensor platforms from the weather (e.g., rain, snow). We enclose each sensor platform in an OtterBox 1600 GPS Case. The OtterBox

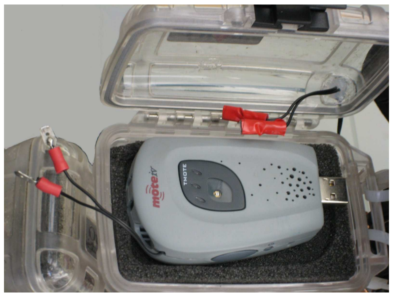


Figure 5: Waterproof OtterBox. wires connected to the user button of the Invent from the reed relay mounted next to the pedal go through a hole drilled in the waterproof Otter Box. The hole is then filled with silicone sealant. Wires have crimped connectors for easy disconnect and removal for mote recharging.

comes with adhesive foam that is customizable to a degree that allows us to secure the Tmote Invents inside the cases without any slipping. A number of sensors require running wires from the Invent out of the OtterBox to other places on bicycle or cyclist (e.g., the WheelSensor's reed relay is wired to the front fork; the GSRSensor's finger pads are wired to the cyclists finger tips). For these we drill holes through the OtterBox 1600 and filled the holes with silicone gel after passing through the wires to maintain waterproofing. For these wired in sensors, we cut the wires inside the box and crimp/solder on connectors for quick disconnect of the Invents for when they are removed for recharging (see Figure 5).

In addition to waterproofing, the sensor platforms have to be securely fastened to the bicycle, especially since bicycling implies often severe vibration and jolting. We are using a system of steel mounting bars affixed to the bicycle frame with steel hose clamps. The OtterBoxes are then screwed on to these mounting bars and the screw holes are sealed with silicone gel. In determining the geometry and placement of the mounting bars we have attempted to minimize vibration and unwanted degrees of freedom for the sensors (Figure 3). This is particularly important for the TiltSensor and LateralTiltSensor which both use carefully oriented accelerometers to obtain their measurements.

Despite our efforts to mount the accelerometers for the TiltSensor and LateralTiltSensor at perfect right angles (in two dimensions) with the ground, we find that a calibration of accelerometers has to be done for each bicycle in order to correctly understand the measured values. Even if the error angle of the mounting bracket is small it can lead to a large skew in the calculated slope. This is true both because the values of slope normally encountered in cycling on a road are relatively small ($<$ 15 degrees), and also because of the non-linear nature of the inverse tangent function through which the readings from the two dimensional accelerometer are passed to calculate the slope. We find that calibration is also necessary for the magneto-inductive sensors due to the steel frames of the bicycle, and the steel mounting bars.

### *3.1.2 Bike Area Network Components*

We implement the following hardware, based on the sensing and communication requirements of the BAN roles discussed in Section 2.2.1.

To measure angular velocity of the wheel and pedal, we augment the standard Tmote Invent by attaching a magnet-triggered reed relay mounted across the Invent's user button. Every time the relay closes a TinyOS [21] interrupt event is generated. For the pedal, the relay and magnet are mounted such that one revolution of the pedal around the pedal axle causes one interrupt event. Similarly, for the wheel, the relay and magnet are mounted such that one revolution of the tire around the front wheel axle causes one interrupt event. Further, knowledge of the circumference of the tire allows for the calculation of distance travelled. Bicycle speed is calculated with knowledge of elapsed time (i.e., the average number of interrupt events per second multiplied by the tire circumference gives the average speed).

To measure the angle of incline, and lateral tilt of the bicycle we use the two dimensional accelerometer integrated into the Tmote Invent. With knowledge of the angles the axes of the accelerometer are aligned with respect to the bike frame, the the slope of the terrain over which the cyclist rides and the side-to-side tilt of the bike is measured. The accelerometers are calibrated to compensate for their mouting characteristics on the bike. We sample the accelerometer at 160 Hz per channel.

To measure direction and deviation with respect to the Earth's magnetic field, we add a dual axis magneto-inductive sensor (Honeywell HMC1052L) to the standard Tmote Invent (Figure 6(a)), connecting the sensor output to two ADC channels on the Invent and connecting a free I/O pin from the Invent's MSP430 microcontroller configured as output to act as a digital control line. The magneto-inductive sensor is powered directly off the battery voltage of the Invent ($V_{batt}$), while the ADC conversion is driven by a divided reference voltage ($V_{ref}$). This requires that we prescale the output of the sensor before it enters the Invent ADC by $(V_{batt} - V_{ref})$ to avoid saturating the ADC channels.

To provide a common notion of absolute time and location within a BAN, we enhance the standard Tmote Invent by interfacing a Garmin Etrex 12 channel GPS unit via the UART0 port of the Invent's MSP430 microcontroller (Figure 6(b). The Garmin Etrex provides time and location data at the fixed rate of once per two seconds via its RS232 interface. Note that the relatively short range of the low power of the CC2420 radio used by the Tmote Invent confines the propagation of the time/location beacon broadcasted by the SyncSprinkler to an area on the same order of the positioning uncertainly of most price-accessible handheld GPS units [14], maintaining the usefulness of the broadcast for all recipients.

When a cyclist pauses his or her ride (e.g., to rest, to eat a snack) it is reasonable that the display of an available N80 mobile phone adopts the LocalDisplay role in the BAN. However, while in motion it may be unsafe for the cyclist to hold the N80, and further not every cyclist would tend to have a mobile phone available to them. To provide a mechanism for handlebar-mounted local BAN display we interface the standard Tmote Invent (Figure 7(a)) with an LCD using a SparkFun interface board and the Hitachi HD44780 LCD driver, via the UART0 port of the Invent's MSP430 microcontroller. The LCD module must have 5VDC to operate so it needs a separate power supply. For simplicity we use a rechargable NiMH 8.4V battery with an LM7805-based 5V voltage regulator circuit. Additionally, to achieve reliable communication over the serial line, a level conversion from the 3V output of the Invent to the 5V input of the LCD is required.

To measure the carbon dioxide levels in the environment surrounding the cyclist, we interface the standard Tmote Invent with the Telaire 7001 $CO_2$/Temperature Monitor, via an ADC port of the Invent's MSP430 microcontroller. The meter has a 0-4VDC output specifically for data logging. The Telaire 7001 uses a separate 9V battery. The Telaire 7001 uses as optical technique (dual beam NDIR) to provide a continuous but delayed measurement of the $CO_2$ level, with a response time of about sixty seconds.

To measure environmental ambient sound volume we use the microphone that is integrated into the Tmote Invent. Interestingly we observe that on the Invent platform, use of the LEDs causes a small DC offset in the microphone capture. This offset should be accounted for when interpretting the ADC values. In firmware, we convert the signal to an average power reading after first rectifying the signal.

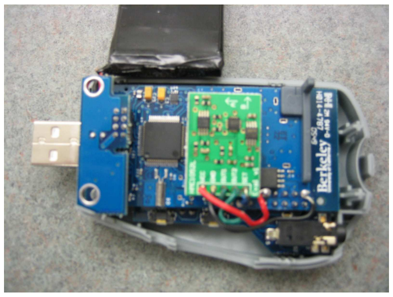
To measure the galvanic skin response of the cyclist, we use an ArcherKit Biofeedback Monitor. Wires connected to the fingers of the cyclist measure microcurrents on the skin. The Monitor normally represents these changes in resistivity of the skin by varying the pitch of an onboard speaker. We have re-routed a voltage level output into a spare port of a Tmote Invent ADC. The Monitor uses a separate 9V battery.

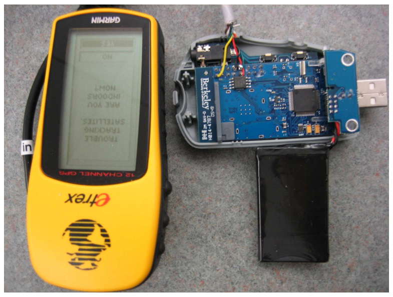
### *3.1.3 Rocks*

In our system, we implement rocks using an unmodified Tmote Invent enclosed in a waterproof Otter Box. While this provides somewhat less storage capacity than we envision for a final system, it does allow us to validate the indirect sharing communication protocols. The results of this validation are shown in Section 4.3.1.

### *3.1.4 Static and Mobile SAPs*

The static SAP is implemented using an unmodified Tmote Invent plugged into the USB port of an Aruba AP-70 IEEE 802.11a/b/g access point. The Aruba is running a customized version of OpenWRT, an embedded linux variant. The BikeNet SAP is implemented as an overlay of tools requiring only user priveleges. Certain kernel module support is needed; modules are loaded at runtime if necessary. Installation of the tools distribution occurs upon hotplug of a USB memory device making symbiotic deployment of a BikeNet SAP on to a standard WiFi access point easy to manage.


(a) Modifications to attach a two-axis magnetometer to a Tmote Invent via the ADC.


(b) Modifications to attach an external GPS unit to a Tmote Invent via the UART0 port.

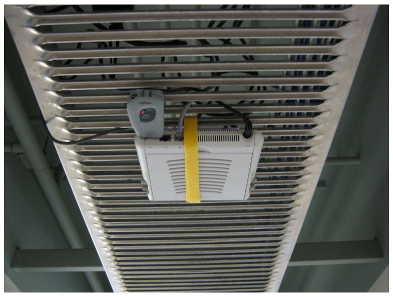
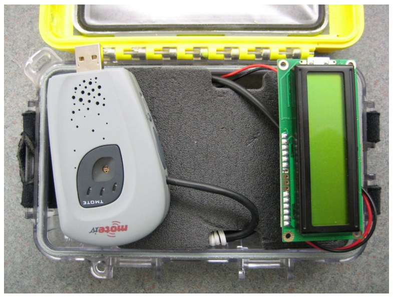
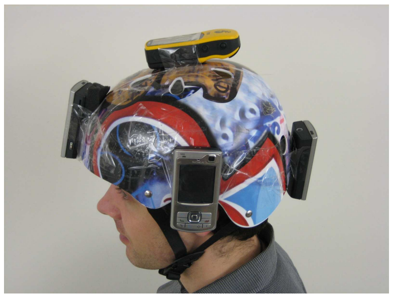

(c) BikeNet static SAP implementation.

Figure 6


(a) Initial prototype of a BAN local display.


(b) Ground truth video/sound/photo helmet with four N80s and GPS receiver. Each N80 can act as a mobile SAP.

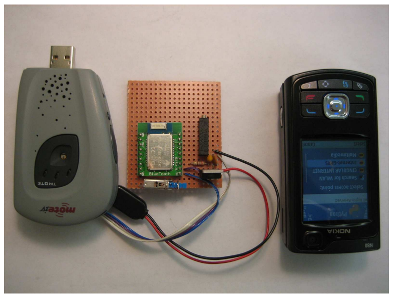

(c) BikeNet mobile SAP implementation. The Nokia N80 BlueTooth radio associates with a custom-built BlueTooth/802.15.4 gateway.

Figure 7

In the BikeNet system the mobile SAP is implemented using a Nokia N80 associated to a BlueTooth/802.15.4 gateway via its BlueTooth radio. The gateway is implemented by interfacing a BlueRadios BR-C40 serial radio modem to the UART0 port of the Tmote Invent's MSP430. The N80 SymbianOS uses a serial device emulation of the bluetooth SPP profile to read and write from the BlueTooth device using serial port semantics. The back end interface of the SAP uses GSM/GPRS to the BikeNet repository and back end services. This is done with a combination of SMS messages from the back end pushed to the phone, and TCP connections initiated by the N80 to transmit responses to the back end. Upon reception of an SMS message, the phone uses a python script to parse incoming SMS messages and acts according to the request. To upload data to the back end, the N80 opens a socket to a web service operating at the back end and uses HTTP POST to transmit both ground truth data (e.g., photos, videos, sound clips) and BAN data (e.g., $CO_2$ levels, road slope). The web server operates a parser for such incoming submissions and constructs SQL statements to insert the data into the back end user repositories.

The use of a personal device like a mobile cellular phone as a mobile SAP gives rise to an interesting dual role for the N80 in our system. Architecturally, there is a clean separation between SAP and sensor platform, but in the case of a mobile phone, the cyclist might own this mobile SAP and thus the BAN to which the cyclist's PersonalNode belongs might have nearly continuous access to the SAP services and resources. Thus, in our implementation, the N80 can be thought of as simultaneously carrying out its SAP role providing uploading, tasking, ground truth sensing, etc. to the BAN, and also acting like a member of the BAN and offering its sensing resources (e.g., camera, microphone, display, speaker). In fact, although in our initial implementation we have kept them separate, one can easily envision an incarnation of the BikeNet system that uses mobile cellular phones as PersonalNodes as well. As a mobile SAP the N80 offers real-time sensing service and access to the owners BAN. In scenarios where the GPRS service is not available, the N80 is used simply as another member of the BAN, and the service falls back to that of a delay-tolerant sensing system. For example, it can take on the LocalDisplay role when operating in this disconnected mode, or might provide audio cues from its speaker to the cyclist a la the training cues given by Nike + iPod [9] during a run.

### *3.1.5 Validation Hardware*

In order to validate the data collected by the BikeNet sensing system, we use a number of commercial off-the-shelf (COTS) hardware solutions. In the following, we discuss how the COTS hardware is applied to increase the confidence in the data collected by our BikeNet implementation.

The magneto-inductive sensor we connect to the Tmote Invent is used to measure heading and to detect distortions of the Earth's magnetic field caused by nearby ferromagnetic materials. In particular, we infer automobile sparsity based on signatures observed in the magneto-inductive sensor data. To validate the ability to determine heading, we simply compare our interpretation of the sensor data to known cardinal and ordinal directions. To validate our ability to detect automobiles, we create a small BAN comprising a GPS-based SyncSprinkler, a MetalDetector, and a standard Tmote Invent programmed to write the (time, location) 2-tuple to the Flash every time the user button on the Invent is clicked (termed *ButtonMote* for ease of reference). We bicycle a planned route and manually click the ButtonMote user button every time we pass a parked automobile or an automobile passes us. We compare the time/location-aligned MetalDetector trace with the ButtonMote trace to determine detection accuracy. In principle, the ButtonMote can be used in a similar manner to validate any detection-based measurement or inference.

To provide richer context for the sensor measurements and inferencing we do in the BikeNet, we attached four Nokia N80 phones at the cardinal points of a bicycle helmet, i.e., facing front, back, left and right (see Figure 7(b)). Using continuous video capture (both visual and audio) throughout the ride we are able to validate that events sensed/inferred by BAN sensors are at least reasonable/probable and depending on the measurement type we can definitively validate the data (e.g., car passing the bike or not). In fact, using the audio capture we are able to give context about the rider that is not obvious or at all detectable from a picture. For example, the cyclist describes his or her mental/physical state in reponse to car exhaust ("I am dizzy"), steep slope ("I am tired"), or overall cycling experience ("I am bored").

We infer cyclist fitness level using a combination of the lateral tilt, slope, and pedal speed to wheel speed ratio. To check our inferencing technique against a more direct physiological measure of cyclist fitness, we use the Garmin Forerunner 301 Heart Rate Monitor/GPS. A positive correlation between our inferred cyclist fitness level, and that indicated by the actual cyclist heart rate validates our technique.

In BikeNet, we calculate road slope by taking the inverse tangent of the ratio the measurements from properly oriented accelerometer axes. To validate the slope measurements using the accelerometer (the TiltSensor role), we measure at 30m intervals the slope of a 0.75km section of the road containing slopes from 0 degrees to 7 degrees using a laser level (model TUV EPT-97A, 650nm). We receive excellent correlation between the two methods.

## 3.2 Implementation Software

To address the design requirements derived in the previous subsection, we implemente a number of data storage procedures, communication protocols, and data visualizations. In the following, we describe a number of these software system solutions grouped according to logical function.

### 3.2.1 *Impact of Hardware Limitations*

Because of the hardware we use in our implementation (e.g., the various sensing instruments, the Tmote Invent) a number of limitations are imposed on the way the hardware can interact, and what software may be written on top of it. In the following we discuss the more interesting limitations we have encountered.

On the Tmote Invent platform, the Flash storage device, the radio, and the UART all share the same physical lines which can be used for either SPI (radio, Flash) or UART. Device access on this bus is mutually exclusive, requiring an explicit time management approach for communication and storage, and since we use the UART as in interface to a number of external sensors and display devices these facets of system operation are also affected. The LCD, the BlueTooth/IEEE 802.15.4 bridge used to network the N80, and GPS receiver all face this problem.

Another characteristic of the Tmote Invent is the limitation imposed by the number of spare ADC channels and free GPIO pins on the MSP430 microcontroller. The result is that multiple Invents are used in order to support all the required sensing roles in the system. Not only does this increase cost by requiring several Invents per BAN, but also increases contention on the wireless channel, especially intra-BAN, since not all data acquisition devides can be "wired in" to a single controller. With more capable hardware, we will be able to condense our current implementation by assigning multiple roles to a single sensor platform.

The Garmin GPS unit we use pushes location over the UART at a non-adjustable rate of once per two seconds. This limitation mandates either a location interpolation scheme, such as using dead reckoning with a combination of data from the CompassSensor and WheelSensor, or accepting a loss of location accuracy of the collected data. Given that even recreational cyclists can easily reach speeds over 50km/hr, especially when going down steep hills, the location error can be substantial - up to 25m or more.

The CC2420 radio used by the Tmote Invent has a relatively high raw bit rate (250 kbps) compared to other mote class platforms, and equivalent transmission power (0 dBm). On the other hand, due to the challenging environment caused by mobilty and by the attenuation and reflections unique to a bicycle area network, radio contact time among BANs and SAPs can be limited and variable. This argues for a maximally efficient transport protocol and application of appropriate compression techniques. At the same time, sensed data is often unique and reliable upload of data is prefered by system users. In the experimental results presented in Section 4 we use an acknowleged transport and no compression to get baseline results for the BikeNet system and defer an investigation of appropriate transport protocol and compression algorithms to future work.

### 3.2.2 *Tasking and Role Assignment*

It is assumed that each cyclist possesses a mobile personal computing device (e.g., mote, mobile phone) at all times that can be tasked by the SAP. In our system, each cyclist carries a Tmote Invent implementing the software architecture shown in Figure 4(a). The SAP tasks the cyclist's personal Invent to take on the PersonalNode role described in Section 2.2.1. The PersonalNode role includes a list of user preferences that dictate what additional sensing roles are desired to quantify the cyclist fitness/performance/environment. These desired sensing roles are split into two lists, required and preferred, represented as 16 bit maps (16 roles are currently supported). The bit maps are included into a *hello* beacon periodically broadcasted by the PersonalNode. Each potential BAN sensor mote in the system running the BikeNet virtual machine (VM) replies to the beacon with a *hello reply* if its sensing capabilities match those of either a required or preferred role as requested by the *hello* beacon, indicating in the *hello reply* which role(s) (e.g., any of those listed in Section 2.2.1) it is offering to fill. If it has already associated with another PersonalNode a sensor will not reply to the *hello* regardless of its fitness to fill a requested role. Upon receiving a *hello reply*, the PersonalNode registers the respondant as the provider of the role(s) offered in the *hello reply*, sends a *hello reply ack* to complete the association, and updates the required and preferred sensing role bit maps in subsequent *hello* beacons to reflect the change. One can imagine a more sophisticated exchange wherein the PersonalNode sends not just a *hello reply ack* but includes also the required sensing parameterization (e.g., sampling rate). In our implementation, for simplicity all sensing parameterization for the sensor that supports a given role is fixed in the BikeNet VM at compile time. Association is reliable in the sense that if a *hello reply ack* is not received in response to a *hello reply*, the *reply* is retransmitted up to $N$ times. If after $N$ times the *ack* is not received, then it is assumed that mobility has carried the PersonalNode and potential sensor apart and the partial association state is purged.

### 3.2.3 *Sensing Control*

When the PersonalNode has assigned BikeNet roles and established an association with sensor platforms to meet all the roles specified by the user preferences (reflected in the required bit map descibed in Section 3.2.2), an LED on the PersonalNode indicates a 'Ready' state. A button click on the PersonalNode when in this 'Ready' state sends a *start* message broadcast from the PersonalNode indicating that the ride is beginning and sensors should start collecting data with their prescribed parameterization. This message is acted on by BikeNet nodes that have associated with that PersonalNode (see Section 3.2.2) and moves both the PersonalNode and the associated sensors into the 'Started' state. If associated sensors do not receive a *start* message within a timeout period, the association times out and the sensors are free to associate with another PersonalNode at that time. A subsequent PersonalNode button click while in the 'Started' state sends a *stop* message broadcast, signaling the end of the ride. The *stop* message is acted on by BikeNet nodes that have associated with that PersonalNode and are in the 'Started' state, and results in a cessation of sensing.

### 3.2.4 *Event Triggered Sensing*

Sensing triggers and the actions that result when the requisite conditions are met are both a matter of user profile. The BikeNet implementation support of triggered sensing includes methods to define and submit sensing triggers and actions to the PersonalNode for execution within the BAN. Upon receiving the triggered sensing definition, the PersonalNode breaks apart the conditions that must be met for the action to take place, and reliably transmits each condition (e.g., "slope $>$ 5 degrees") to the BAN member suited to evaluate the condition (e.g., the TiltSensor). When a condition evaluates to true, the BAN member signals the PersonalNode. The PersonalNode initiates the action when all conditions for a given triggered action are met. We are currently focusing on triggered photograph, video and audio capture using the camera and microphone on the N80, when certain conditions in the BAN are met, but we have also implemented support for a number of other actions such as sending data to be displayed on the BAN LCD, sensing something at a different parameterization than the current one, playing a sound on the N80 or Invent speaker, transferring sensed data from one sensor platform to another, and trivial actions like blinking an LED on the PersonalNode.

### 3.2.5 *Resource Sharing*

A PersonalNode requests to share the resources of another BAN if it does not have all the preferred sensing roles as required by user profiles. In order to find sensor platforms that can fill the desired roles, the PersonalNode executes the same protocol decribed for sensor association for native sensor roles in Section 3.2.2. However, each message contains a one byte field that indicates whether this is an exchange for native association or resource sharing. In practice, in our implementation, the difference between native and borrowed resources is that native associated resources are controllable via *start* and *stop* messages. Borrowed resources that hear a *stop* message from a PersonalNode to whom they are loaned do not stop their own sensing (only a *stop* message from their native PersonalNode can cause this); instead they terminate the sharing assocation and purge any outstanding triggers or streaming data transfers executed solely on behalf of the sharing BAN.

### 3.2.6 *Synchronization/Localization*

The SyncSprinkler periodically broadcasts location/time data obtained from a GPS receiver to the bike area network. Each mote in the bike area network keeps a local estimate of its current location and the current time. The local time estimate is updated externally with the values contained in the SyncSprinkler broadcasts, and internally via a local clock to provide higher time resolution between SyncSprinkler broadcasts. Similarly, the local location estimate is updated with the values contained in the SyncSprinkler broadcasts. In Section 4.3.2 we present results on the feasibility of using the combination of distance value from a WheelSensor and a heading value from a CompassSensor to provide an estimate of location within a bike area network, as an alternative to the GPS-driven SyncSprinkler solution. Each mote in the bike area network stamps its sensed data with its current location and time estimates before storing the data in its local cache.

### 3.2.7 *Indirect and Direct Information Sharing*

Indirect information sharing occurs via a Bike-to-Rock protocol, in which any mote in the BAN may participate. The protocol comprises three phases: discovery, establishing session context and data exchange. Since a rock is untethered (no line power) we reduce its energy burden in the discovery phase by having it passively listen until it overhears intra-BAN messages from an approaching bicycle. Upon overhearing, it becomes active and begins beaconing rapidly to notify passing BANs of it's presence. This beaconing times out after a period of no response, after which it is assumed the BAN has moved out of range or is not interested. BANs that wish to use the beaconing rock establish a session context through an authenticated exchange, after which a meta-data exchange occurs. The meta-data contains information about data that the BAN, based on the cyclist preferences, aims to store/retrieve to/from the rock. The session concludes with the data exchange, an acknowledged stream from the rock to the BAN mote. Sessions time out after a period of inactivity. For example, a cyclist may store her average speed of the prior 3 km of trail every time she takes the trail. As she passes the rock during the current ride she both inserts her sensed data from the current ride, and retrieves data of the same type from her personal history, the history of her friend, and the average of all cyclists that have taken the path. A representation of the information retrieved from the rock can be shown in real time to the cyclist via her LocalDisplay, if desired.

The Bike-to-Bike communication protocol for direct information sharing is indentical to that of the Bike-to-Rock communication protocol, though no rock resources are used. While indirect information sharing via rocks is done with a focus on sharing location-specific data, direct information sharing is rendezvous-specific; you share information with a person rather than about a place.

### 3.2.8 *Real-time Feedback/Display*

The BAN LCD protocol is used by the LocalDisplay to query nearby sensor platforms for values to display. The LocalDisplay is provided either by an N80 mobile phone, or by a handlebar-mounted LCD. Because the Invent's hardware design shares the SPI bus between radio and flash, and the same physical microcontroller pins are used for UART0, SPI bus arbitration is necessary to write data to the LCD display module. The same limitation exists with our N80/mobile SAP implementation since the BlueTooth-to-Serial converter is also connected to the Invent UART0 port. This precludes simultaneous radio communication and display updating, and hence BikeNet uses a simple TDMA-like time slot assignment on top of the CC2420's CSMA MAC to improve communication between the sensor platforms and the LocalDisplay. The LocalDisplay broadcasts a query for data and the sensor motes register the first LocalDisplay they hear a query from as the only LocalDisplay they will reply to thereafter. This association times out after a period if no queries are heard from the LocalDisplay. The data that a given sensor role returns is a matter of user policy, but a typical display might include speed, distance traveled, bike frame tilt angle, pedal RPMs, and time of day. Data from rock motes and debug messages from sensor motes can also be displayed (this is helpful for troubleshooting in the field). A simple extension to the protocol allows displaying select data from other nearby BikeNet equipped bicycles. Future work will include adding graph displays of the cyclists performance and profiles of road characteristics.

### 3.2.9 *Sensed Data Muling and Upload*

A simple muling protocol is implemented on every mote, but the option to activate muling (i.e., spend Flash space to carry others' data) is by cyclist preference. The protocol uses an *advertisement-accept-data* exchange, where the *advertisement* specifies the number of data records the data provider has available in Flash for muling, the *accept* message indicates the number of records the data consumer is willing to mule (based on Flash constraints) and the *data* message represents a burst of data packets from the producer to the consumer. In addition, Stop-and-Wait ARQ with a maximum of three resends provides for reliable transfer of the data packet burst. If a producer still receives no acknowledgement after three resends of the same packet it will assume the session established with the *advertisement-accept* exchange is over (most likely due to mobility) and begins advertising anew. Our implementation includes support for replication of sensed data via the muling process but re-muling (the replication of muled data) is not currently supported. Disallowing re-muling allows the data source to maintain control over the number of copies of its data that are circulating and more importantly with whom the data is been shared. The upload protocol message exchange is identical to that of the muling protocol. When a SAP receives data packets, they are forwarded via serial port (in both the mobile SAP and static SAP cases) to the back end repository. The decision to accept new upload sessions by the SAP is made based on channel congestion around the SAP. Results on the muling and uploading protocol efficiency are shown in Section 4.3.

## 4 System Evaluation

In this section, we present results from several groups of experiments repectively targeted at: quantifying the cyclist experience by sensing and inferencing from sensed data collected about a single cyclist and his or her environment; investigating the correlations between data collected by cyclists traveling in a flock (e.g., a group of

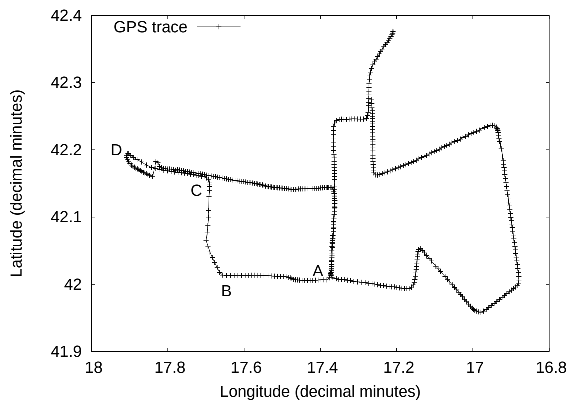
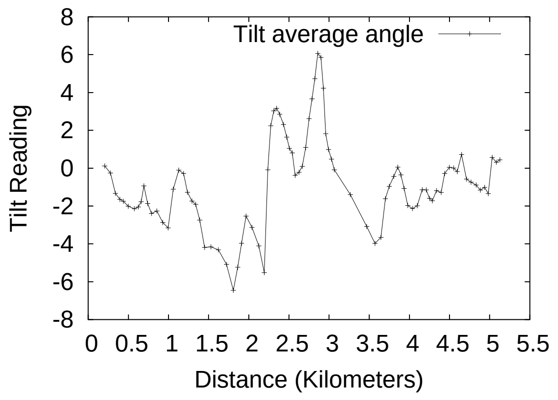
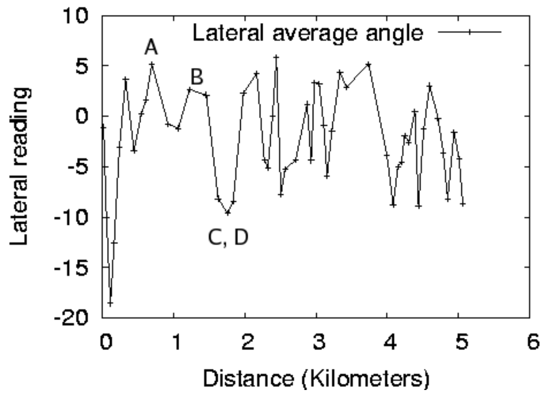


Figure 8: The mapped GPS trace of the cyclist route. The route comprises roads in the vicinity of Dartmouth College in Hanover, NH, USA.



(a) Tilt vs. distance



(b) A plot of lateral tilt versus distance. Periods of increased lateral tilt can be correlated with corner turns expected from the mapped GPS trace shown in Figure 8.

Figure 9

riders); looking at performance aspects of key BikeNet subsystems; and measuring the real-time performance of a deployed system across the Dartmouth campus and in adjacent areas of Hanover, NH. For the single bike and flock experiments we use a common path that we call the ground truth route. This route includes a variety of urban cycling terrain, including built up busy roads in the town center with lots of cars and pedestrian traffic and quiet back roads with little or no traffic. The route exposes cyclists to a variety of flat terrain, gradual down hill and steep uphill sections. Typically the ground truth route takes 25-30 minutes to ride and is nearly 5 km long. The experiments are conducted at rush hour and in the middle of the day when there is less traffic and activity. We conducted many experiments over the period August - November 2006 collecting a typical data set of 0.8 MB per ride per bike. We plan to make the data traces available [4]. For all experiments we record the experiments using video from the video helmet (Figure 7(b)). This allows us to collect 4-directional video of a ride, and correlate events in the video with sensor data in terms of ground truth - see [5] for an example of video-to-sensor correlation that we use as part of our experimental methodology.

## 4.1 Cyclist Experience Mapping

### *4.1.1 Inference and Cyclist Fitness Sensing*

In this section we present a series of plots characterizing cyclist behavior and the environmental conditions encountered during a ride. We collect data from each of the sensing roles mentioned in Section 2.2.1, and apply data fusion and trend analysis to infer higher level context from the raw data.

Figure 9(a) shows the slope calculated from TiltSensor readings versus distance. The slope is measured by feeding the *arctan* function with the TiltSensor's x- and y-channel accelerometer readings. We register accurate measurements when the bike is stationary; error increases with speed and terrain roughness due to unfiltered vibrations and cyclist behavior. Figure 9(a) shows the profile of the route with a 90% accuracy when compared to ground truth measurements.

Figure 9(b) shows the readings obtained from the LateralTiltSensor versus distance. The lateral tilt is measured by feeding the *arctan* function with the LateralTiltSensor's x- and y-channel accelerometer readings. A cyclist's aggressiveness in turning is inferred. From the plot we correlate the increases in lateral tilt shown on the y-axis, with corner turns expected from the mapped GPS trace shown in Figure 8. Positive increments of the lateral angle indicate a right-side lean whereas negative angles are for left-side leans. The letters in Figure 9(b) mark the readings corresponding to the turns indicated with the same letters in Figure 8, where at A, B, C, and D the biker makes, respectively, a right, a right, a left, and a left turn. The very sharp left tilt (almost -20 degrees) is due to mounting the bicycle at the beginning of the ride. Figure 9(b) shows that it is possible to infer biker's turns (positive angles for A and B, negative angles for C and D).

The quantitative aspects of the cyclist fitness include the slope of the road/trail that the cyclist covers on her

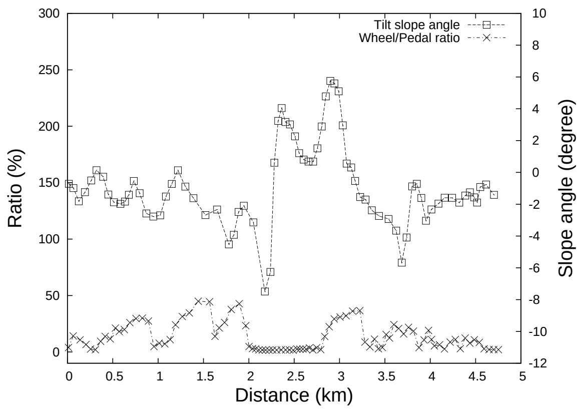


Figure 10: A plot of road slope and wheel speed to pedal speed ratio along the traversed route. Cyclist fitness can be inferred by correlating the gear ratio inferred from the wheel/pedal ratio and the measured road slope.

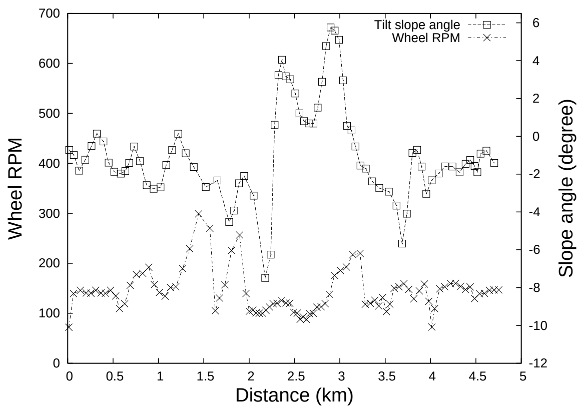


Figure 12: A plot of road slope and wheel RPM along the traversed route. Periods of strong braking can be inferred by slowing wheel speed while on a increasing or stable downhill slope.

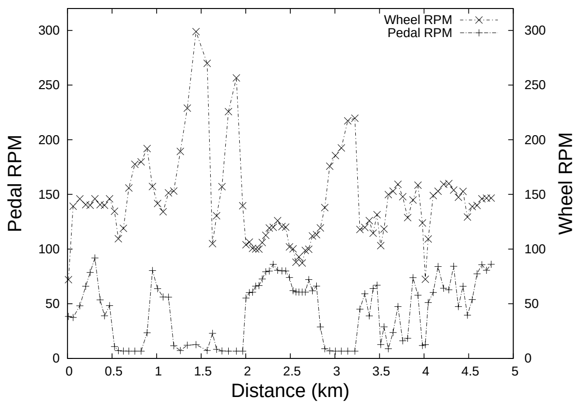


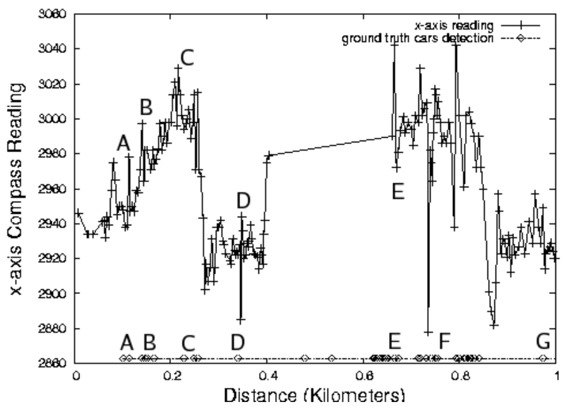
Figure 11: A plot of pedal RPM and wheel RPM along the traversed route. Periods of coasting can be inferred by zero or near zero pedel RPM with non-zero wheel speed.

ride, the speed profile of the cyclist, the gear used when traveling up a given slope, and the location of the route. Figure 10 shows a plot of the slope of the road traversed on the trip of the cyclist, and the ratio of the the tire/wheel speed, as measured by the WheelSensor, to the pedal speed, as measured by the PedalSensor. This ratio infers the approximate gear the bicycle is in at a given point in the route, or equivalently provides a notion of the fitness of the cyclist. This fitness indicator is most accurate when the cyclist is going uphill, since when going downhill the pedals may not be moved much at all (coasting). A cyclist with strong legs can use a higher Wheel/Pedal ratio when climbing hills. On the other hand, recently the use of a high pedaling cadence (which gives a higher aerobic workout) at all times throughout a ride has come to prominence [37]. In Figure 10 it is possible to infer intervals where the cyclist changes gears to climb hills (from roughly 1 to 1.25 km and from roughly 2 to 2.7 km), where the Wheel/Pedal ratio is close to 1.

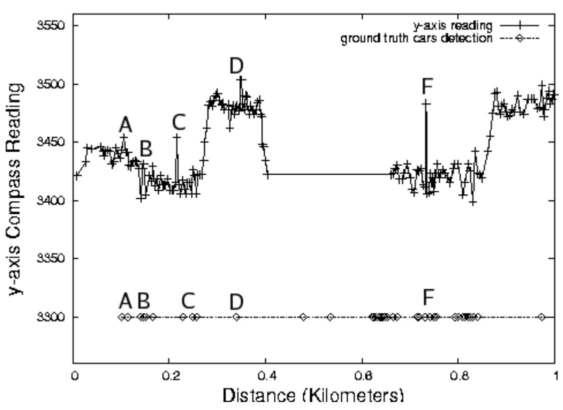
Knowledge of the sensed path slope combined with the measured pedal speed and wheel speed allow us to infer when a cyclist is coasting or braking. On a given bicycle there is a finite discrete set of allowable pedal speed to wheel speed ratios when the bicycle chain is engaged with a gear and providing thrust to the bicycle. The cardinality of this set is equal to the number of gears the bicycle has. If the measured ratio of pedal speed to wheel speed does not match one of the allowable values we can infer that the cyclist is coasting. In particular, if the pedal speed is zero and the wheel speed is non-zero, we can easily infer the cyclist is coasting. Figure 11 shows the pedal speed to wheel speed ratio versus distance. In the figure, from roughly 1.25 to 2 km and 2.7 to 3.3 km, the pedal speed is approximately zero while the wheel speed is high, indicating coasting.

Braking can be inferred in a similar fashion to coasting. It is likely a cyclist is braking if the measured wheel speed slows while the slope is negative (downhill). Further, braking is likely when going uphill if the measured wheel speed slows faster than dictated by the slope of the hill, though this is more challenging to detect. Note that inference of uphill braking is dependent on knowledge of the combined mass of the bicycle and the cyclist. Further, in all cases, but particularly when the slope is flat, inference of braking is complicated by the fact the generally the route surface composition is not known exactly and thus the appropriate coefficient of rolling friction to assume is also unknown. This fact precludes detecting slight braking, but hard braking can still be detected in most cases. Figure 12 shows the wheel speed and slope versus distance along the route shown in Figure 8. In the figure, applying the simplest inferencing technique of observing decreasing speed when the downhill slope is increasing or level, we see sharp braking intervals at $\approx$ 1.6 km, $\approx$ 3.3 km and $\approx$ 3.9 km. In these cases, we see a sharp decrease in wheel speed concurrent with a sustained downhill slope.

Aside from route topography and personal perfor-



(a) x-axis magnetometer readings



(b) y-axis magnetometer readings

Figure 13

mance metrics, cyclists are interested in the ambience of a route as a determinant in the overall enjoyment of the cycling experience. We take steps towards quantifying the ambience in terms of automobile traffic, air quality, sound level and GSR readings. The presence of vehicles is often undesirable for cyclists who have concerns about safety, noise, and pollution. To infer automobile traffic along the cyclist route (Figure 8), each BAN is equipped with a MetalDetector. When the MetalDetector passes close to any large metal body the Earth's magnetic field is deformed and is interpreted as an anomaly on the magnetometer; the time and location of the inferred car presense is stored in Flash. We write embedded software that runs automatically to calibrate the magnetometer once mounted on the bikes to take into account the effect of the metal of the bike and the rack where it is mounted. The calibration routine stops once the ride data capture starts. To collect ground truth data for comparison against the inference from the MetalDetector, the event of passing by a car is registered by the cyclist by means of a ButtonMote click. Each click generates a record of GPS time and location information, and video captures. The ButtonMote records are stored into its Flash memory, as well as uploaded once the BAN comes in contact with a SAP. We use the ButtonMote data and data from the video stream discussed above as ground truth for car detection. Figures 13(a) and 13(b) show, respectively, the x-channel and y-channel of one MetalDetector's magnetometer readings plotted versus the distance covered along the ride. ButtonMote events are shown overlayed on the same plots for comparison with the ground truth. To better capture the details of the series, we report in the graphs only a portion that includes the first kilometer of the ride. It is evident that the magnetic field anomalies measured by the MetalDetector (as seen by sharp simultaneous spikes on x and y channels of the magnetometer) occur at the same locations as the ButtonMotes events. We have annotated Figures 13(a) and 13(b) with letters to highlight the matches. Despite the high correlations we see, additional filtering and signal processing algorithms are needed to detect and eliminate false positives.

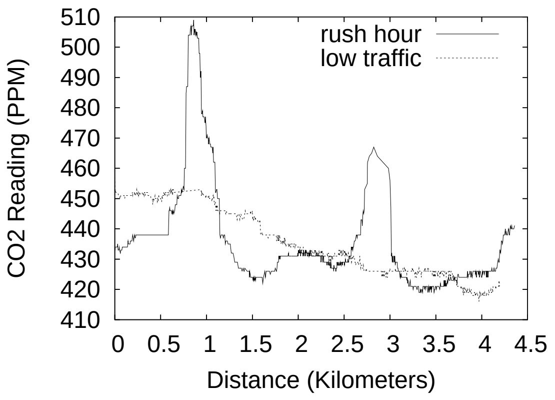


Figure 14: A trace of the carbon dioxide readings along the route shown in Figure 8.

To provide a measure of air quality along the cyclist's route we conduct experiments using a sensor measuring the level of carbon dioxide in the air surrounding the cyclist. Figure 14 shows a trace of the carbon dioxide sensor readings along the route shown in Figure 8 for two different cases, namely, rush hour and low traffic. The peaks in the rush hour case occur when the biker was cycling on downtown roads with a considerable presence of cars. In fact, variation in carbon dioxide levels measured on roadways is likely to be the result of automobile exhaust. While carbon dioxide has low toxicity at all levels we recorded during our experiments, it can act as a predictor of other noxious automobile exhaust gases such as hydrocarbons, nitrous oxides, and particulate matter. Thus, from readings of the carbon dioxide sensor we can infer how enjoyable the traveled route is for a cyclist from the standpoint of pollution.

Another way to detect the presence of vehicles is by the adoption of a microphone. For this purpose we in-

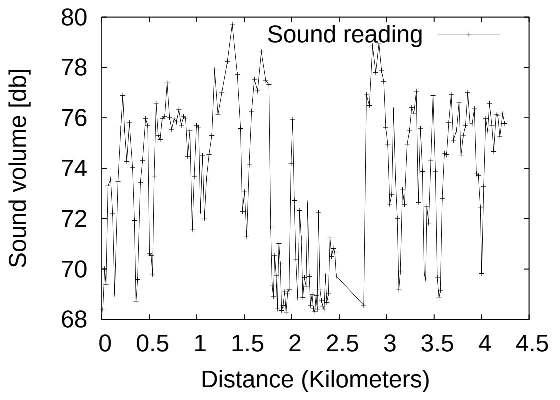


Figure 15: Sound vs. distance

clude in the BAN a SoundSensor. The readings derived from the SoundSensor are reported over distance in Figure 15 for a ride along the path of Figure 8. As expected, the sound volume is higher as the number of cars increases and again this occurs in downtown areas from 1 to 1.7 km and 2.7 to 3 km when the biker rode on the same location twice along her ride.

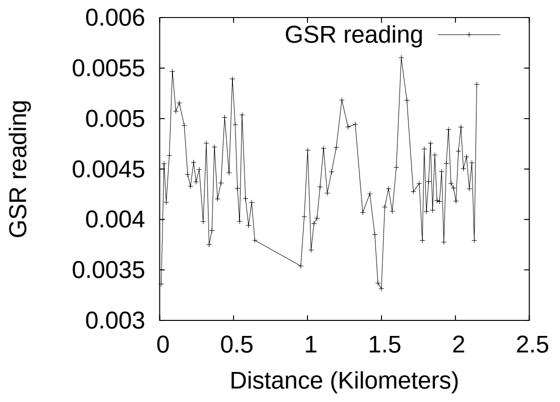
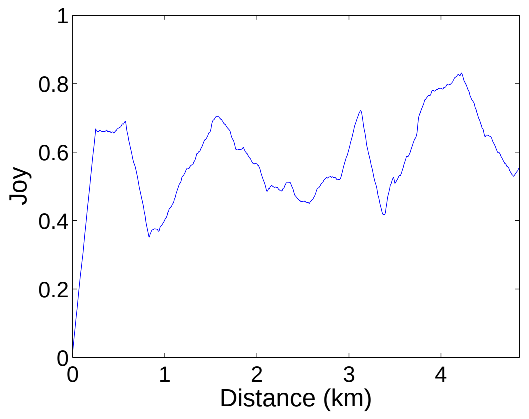
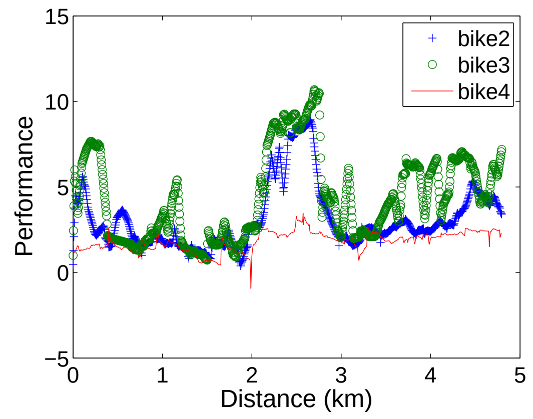


Figure 16: GSRSensor readings

By using the GSRSensor we aim to measure the stress level of the biker. How tense is the biker when she rides between cars? Is that incline a struggle? We conduct one of the experiments by equipping the cyclist with a GSRSensor along the path of Figure 8. The result is shown in Figure 16 where we report only the first 2 km of the ride to better catch the details of the GSR readings on the y-axis. It is possible to identify three segments of the path where the GSR registers the higher values, namely, from 0 to 0.5 km, from 1 to 1.3 km, and from 1.6 to 1.7 km. These roads correspond, respectively, to a cars populated urban area, medium steep downhill and medium steep uphill. From this result we can infer that the cyclist was experiencing a certain degree of stress while riding in town, possibly because of the presence of cars, she was tense while traveling downhill being



(a) A plot of enjoyment index for a single rider



(b) A plot of performance index for three riders with different preferences.

Figure 17

carefull given the high speed, whereas her workload was high during the uphill.

### 4.1.2 Quantifying a Cycling Experience

From the raw values obtained from the magneto-inductive sensor and the carbon dioxide sensor, along with sound levels from the microphone and slope values calculated from the accelerometer, we derive values for the *enjoyment index* of routes that a cyclist travels. At the back end data repository, these index value are then mapped to colors and routes are visualized as a color-coded playlist (see [5].) We quantify the enjoyment using the following equation:

$$Joy = 1.0 - a_1 * HillAngle - a_2 * CO_2 Level - a_3 * SoundLevel.$$

When the HillAngle is positive cycling is more difficult, reducing the Joy of the casual cyclist. When HillAngle is negative the cyclist can coast, a pleasurable experience for most people, resulting in an increased Joy index. A higher $CO_2$ level implies there are more cars near the cyclist, creating an unpleasant experience due to exhaust, noise, and increased danger, driving the Joy index down. Similarly an increase in noise level indicates more traffic, people, wind, shouting, etc., reducing pleasure and the Joy index. The selection of weighting factors is cyclist dependent since each person has different levels of dislike for these annoyances. Figure 17(a) shows a plot of the Joy index for a given rider over the course of the route shown in Figure 8. The weighting factors chosen here were $a_1 = 0.5$, $a_2 = 0.3$, $a_3 = 0.2$, indicating the cyclist dislikes steep rides the most, is somewhat annoyed by cars, and doesn't care much about the sound level. Accordingly, the Joy index goes down during the steep and car congested parts of the ride. By aggregating data from multiple cyclists over time, the relative The joy index of paths is computed to predict where the good cycling is. Optionally, the maximally enjoyable path is computed between two endpoints.

Some cyclists primary purpose in riding is for exercise or for competition and they may be less interested in the joy index of a ride. For these riders we calculate a *performance index*, using the raw values obtained from the WheelSensor, PedalSensor, and TiltSensor. We compute a unitless measure of performance ($P$) using the following equation:

$$P = a_1 * HillAngle + a_2 * WheelSpeed/PedalSpeed + a_3 * Distance.$$

When HillAngle is positive the performance index goes up; when it is negative the index goes down. When the wheel/pedal ratio is high this indicates the bike is in a higher gear (the wheel goes further with fewer pedal turns) and the index increases. The further a rider travels (larger *Distance*) the higher the performance index. The graph in Figure 17(b) shows the comparative performance of three cyclists, all traveling the same route at the same time. Two of the riders perform similarly but the third rider put his bike in a lower gear when going up the steep hill and thus has a lower performance between the two and three kilometer marks. The weighting constants where chosen such that traveling five kilometers, a wheel/pedal ratio of 0.1, and a tilt of five degrees were all considered to have equal effect. Unlike the joy weighting constants, the performance constants should remain the same for all riders. By aggregating data from multiple cyclists over time, the relative performance index of paths a cyclist is considering is computed to predict where the most challenging paths are. Optionally, the most challenging path is computed between two endpoints.

### 4.2 Bicycle Flocks

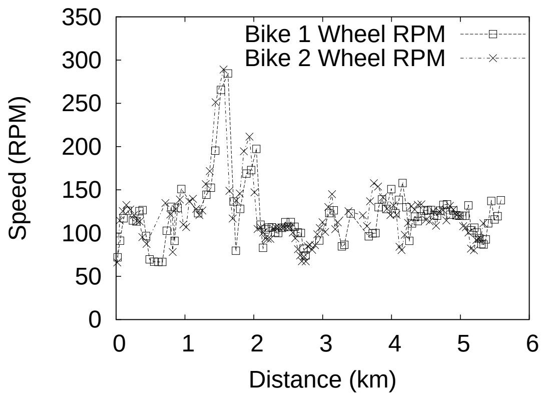
There is a strong social element to bicycling, with many cyclists often riding in groups or flocks. If all of the cyclists have BikeNet sensors then we would expect certain types of sensors to return very similar readings for all the cyclists in the flock. This correlation amongst readings may have several applications including: automatic calibration checks of sensors, noise reduction in sample data using shared information, and sharing of sensor data with those who are missing sensors (e.g., the more expensive ones.)

We examine the correlation between the sensor data in one of the multibike experiments to explore this facet of BikeNet. In this experiment the cyclists usually remain within about 10 meters of each other, though there are occasional separations of as much as 50 meters. The route taken by the cyclists is the same as shown in Figure 8. We compare results from a representative two bicycles in the following plots. Figure 18(a) shows the speed profile of two bikes, demonstrating a strong correlation in speed throughout the experiment. Figure 18(b) shows the tilt profile of the same bikes, again demonstrating a strong correlation in tilt. Figure 19(a) shows the compass readings from the bikes are strongly correlated. (Note that one compass stops reporting data between 0.4 and 0.65km. Note also that one of the cyclists has a tendency to wander from side to side, especially near the 0.9 kilometer mark.)

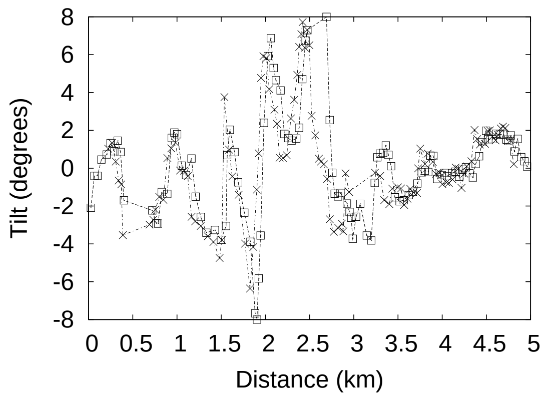
Figure 19(b) on the other hand shows that the sound sensors on the two bikes are only somewhat correlated. This graph is computed from the sound data from both bikes, interpolated so that the samples are aligned at the exact same distance along the path. Thus we are comparing the sounds at the same location (and not necessarily at the same time.) The weak correlation is due to several factors including multipath reflections of sound, wind noise variations over time, one or the other riders yelling comments, inverse square fading of volume with distance, cars passing each cyclist at different times, and phase differences from small differences in each bikes path.

Some sensor data exhibits no correlation at all. The lateral tilt of each bike is dependent on the way each cyclist rides and varies a lot from bike to bike. GSR is unique for each cyclist. Gear ratio is loosely correlated depending on several factors including the strength and condition of the cyclists, the gear ratios available on each bike, the wheel size, and the choice of each cyclist as to when they pedal and when they coast. Pedal speed has a similar loose correlation for the similar reasons. $CO_2$ data is highly correlated between riders in a flock since the exhaust of cars becomes well mixed by the turbulence created by the cars motion. GPS data is strongly correlated between cyclists depending on how close together they ride.

Thus the sharing of some types of sensor data is feasible and likely to be quite useful. In the course of this study we learn that the sample rates for several of our sensors reduce the potential usefulness of shared data. For example, we want to try correlating GSR to pedal RPM's to see if the periodicity of the pedaling shows up due to the shifting pressure the cyclist put on the GSR electrodes. However the 1Hz sample rate of the GSR output is too slow to capture this phenomenon if it exists because people often pedal at somewhere near 1Hz. For future work we are searching for methodologies for examining the correlation of data that may be skewed in time or location, and which may have only a fuzzy sort of correlation. For example, correlating $CO_2$ concentration with car detections from the magnetometer is complicated by the fact that the $CO_2$ sensor reacts slowly to changes in $CO_2$ concentration, taking about a minute to respond to changes.

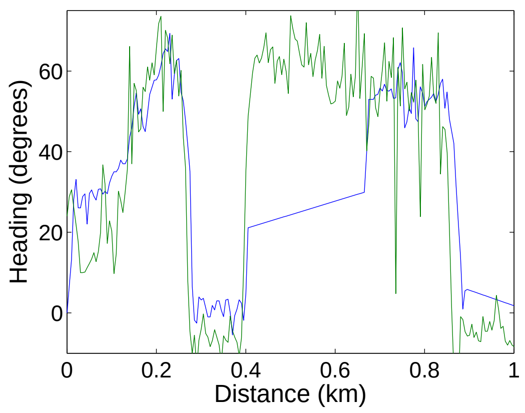


(a) Comparison of speed profiles of two bikes.

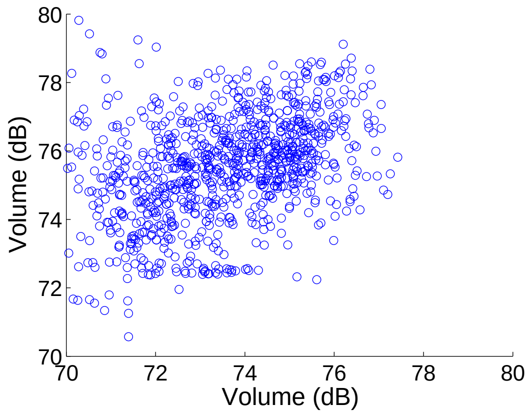


(b) Comparison of tilt profiles of two bikes.

Figure 18



(a) Comparison of compass profiles of two bikes.



(b) Correlation test of sound data from two bikes.

Figure 19

## 4.3 Services Performance

### *4.3.1 Data Sharing*

In what follows, we evaluate two types of BikeNet data sharing. We first consider a bike-to-rock experiment that analyzes sharing between a mobile bike that passes a stationary rock device and attempts to publish event data. Second, we evaluate a bike-to-bike sharing experiment where both bikes are mobile, presenting a limited rendevous window for the BANs to interact and share sensing modalities. Data sharing under these conditions is challenging because of multipath conditions and radio propagation issues, and due to the short range nature of the radio (between 30m-50m outside), communication opportunities are short lived in many cases (e.g., on the order of seconds for bikes traveling in opposite directions that initiate sharing). Therefore, BikeNet is designed to quickly assert communications and attempt to complete operations in a timely and reliable manner as discussed in Section 3.2.7.

We first consider the bike-to-rock experiment. The

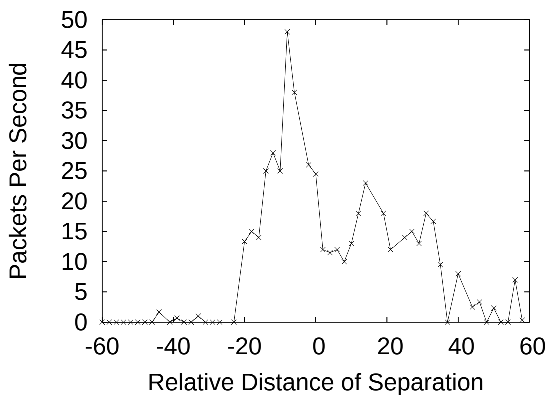


Figure 20: Effect of bicycle separation on session packets received.

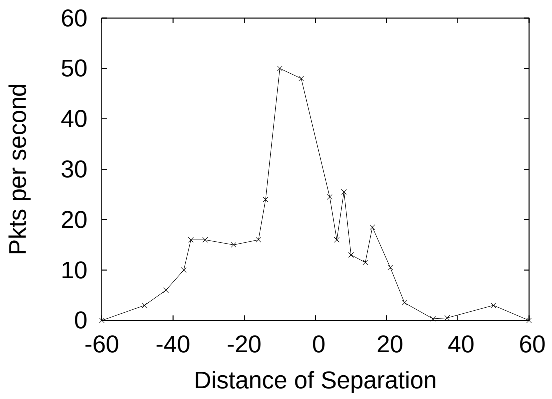


Figure 21: Effect of separation between bike and rock on session packets received.

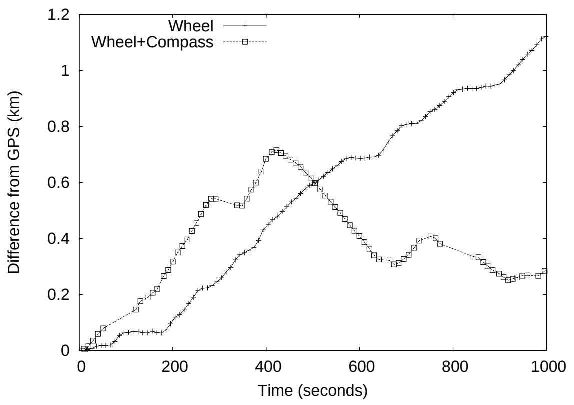


Figure 22: Deviation from GPS of measured distance by WheelSensor; deviation from GPS of absolute location by WheelSensor+CompassSensor using dead reckoning from a known starting location.

experimental set up is as follows. The rock is placed 0.3 m off the ground at the mid point of a 240 meter long path that the bicyclist traverses in a straight line at an average speed of 2 meters per second. The BAN attempts to detect the rock and upload data. We record the time series of the packet reception at the rock for ten separate experiments. Figure 21 shows the packet delivery rate per second against distance for the experiment. Distance is interpreted as follows, a negative number indicates that the bike is moving toward the rock, while a positive number means the bike is moving away from the rock along path. Distance zero indicates that the bike and rock are at the same point (as measured by the WheelSensor). From the plot we observe an average packet transfer rate over ten runs of 22.7 packets per second. This is for reliable ordered data transfer as discussed Section 3.2.7. As part of the experiment we record the contact time between the bike and rock which represents the interval in seconds that bike and rock are capable of exchanging data. In the experiment the bike attempts to upload far more data than is possible, meaning that the bike always has more data to exchange assuming that there is an association (i.e., the bike and rock are in radio range). From the results we observe an average contact time across all ten experiments of 9.25 seconds. This results in an average transfer of 637 bytes between the bike and the rock.

Next, we consider a bike-to-bike sharing experiment. The same experimental setup discussed above is used. We conduct ten independent experiments for two bikes that start down the same path starting at opposite ends of the path. We record the average speed of each bike and then average each bike's speed over ten runs. To measure distance between bikes we select one bike as the reference bike from which the distance is computed and use the same signed distance semantics as in the bike-to-rock experiment. Results for the bike-to-bike experiment are shown in Figure 20. The plot shows the reliable data transfer of the reference bike's CompassSensor data operating at a very high sampling rate to produce sufficient traffic to share with the other bike. In this scenario the reference bike always has data to transfer if there is an association between the bikes. The average speed of the reference bike and other bicycle are recorded as 3.45 meters per second and 2.55 meters per second, respectively. The average packet transfer rate recorded for an average contact time of 10.75 seconds is 19.8 packets per second. This results in an average reliable transfer of 556 bytes between the bikes.

From both experiments we can make a number of similar observations. We observe asymmetry in throughput performance between the transmitter and receiver in both experiments (i.e., the bike is the transmitter and the rock the receiver in the first experiment, and the datum bike is the transmitter and the other bike is the receiver in the second experiment). This throughput skew or asymmetry is clearly shown in both plots. We believe this is an artifact of the placement of sensors on the bikes. The radio transceiver is located on the front of the bike therefore, we conjecture, that the drop is due the the additional interference presented by the body of the cyclist rather than an other issues (e.g., we discounted doppler as a factor). For future work we will study changing the transmitter on the fly to use the sensor platform with the best channel to the receiver to provide increase throughput and to remove this throughput assymentry.

#### *4.3.2 Cyclist Localization*

In the BikeNet system, each data record is tagged with location (and time) to provide context. Section 2.2.2 describes the BikeNet localization design. While GPS is a natural solution for the implementation of the SyncSprinkler role, other lower cost implementations warrant an investigation given that not every BAN may be equipped with a GPS receiver, and GPS is limited in its application since it must have vision of at least three satellites. In particular, it is not always functional in some common bicycling environments (e.g., among tall buildings in a city, under dense overgrowth (tree

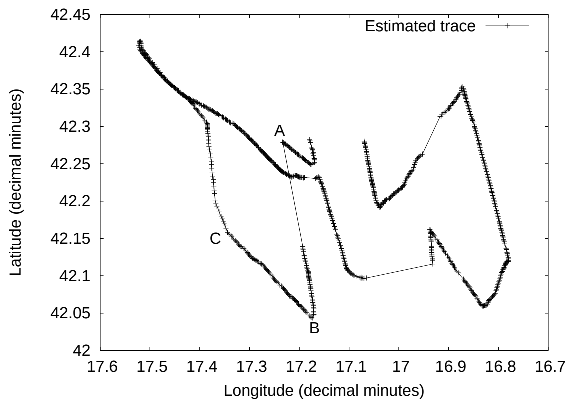


Figure 23: Map of estimated absolute location using WheelSensor+CompassSensor using dead reckoning from a known starting location. The skew compared to the actual location trace shown in Figure 8 is the results of a miscalibrated magnetometer channel.

canopy)). In the following, we present results determining the accuracy of a WheelSensor in measuring total distance traveled by the cyclist. We also present the accuracy of using the combination of WheelSensor and CompassSensor for measuring localization (i.e., dead reckoning). In both cases, a known starting location is required. Both results are shown with respect to the GPS solution.

We present the absolute error of these localization alternatives in Figure 22. As seen from the curve labeled 'Wheel', the deviation from the GPS-derived distance grows continuously over the course of the ride. A closer look at the data shows that WheelSensor in consistently underestimating the distance traveled. The WheelSensor technique requires that the circumference of the tire be known; we initialized the WheelSensor with the value we measured with a tape measure. We conjecture that over the course of a ride, small errors in this measurement accumulate to cause the continuously increasing error. As seen from the curve labeled 'Wheel+Compass', the deviation in absolute localization from GPS has a rather symmetric growth and recovery shape. We conjecture this is due to a calibration errors in one channel of the CompassSensor's magnetometer. Comparing the localization error trace to the route taken by the cyclist, we see that error always increases when heading west or south, and always decreases when heading east or north. This is consistent with miscalibraion of one of the two analog outputs of the magnetometer, which have the following trigonometric relationship to the heading: $arctan(Y/X)$. We can view the manifestation of the compass miscalibration another way by mapping the absolute coordinates derived from the 'Wheel+Compass' solution and comparing the route to the GPS trace shown in Figure 8. Figure 23 shows this location trace estimated using dead reckoning.

While the GPS-substitutes we evaluate here may not be accurate enough to replace GPS, we note that in the case of both the 'Wheel' and the 'Wheel+Compass', the deviation from GPS, for example, is relatively small from time 0s to time 200s (e.g., $< 200$m). Therefore, we conclude that these GPS-substitutes may be used as complementary solution that interpolate between occasionally received GPS beacons, e.g., once every 2 minutes.

The ability to localize a cyclist (via GPS or a CompassSensor/WheelMote alternative) gives rise to a "lazy tracking" feature of BikeNet. Since each data record is stamped with time and location metadata when it is sampled, any packet that is delivered to the data repository contains information that can be used to reconstruct the track of individual cyclists.

### *4.3.3 Data Uploading and Muling*

Throughout the conducted experiments, sensed data is always stored in the local Flash memory of the sensing platform (i.e., Tmote Invent). Ultimately, this data must be transferred to the back end data repository. Depending on bicycle and cyclist mobility this transfer may happen directly from BAN to SAP using the upload protocol (Section 3.2.9), or may happen indirectly via the muling protocol (Section 3.2.9). There are two main scenarios to consider: (i) the bicycle enters in range of a SAP in which case the sensor nodes in the BAN upload their data to the SAP directly or (ii) the bicycle is travelling out of range of the SAP, and must rely on probabilistic mobility of other people or BANs to mule the data to a SAP.

Since the Stop-and-Wait ARQ reliable tranfer mechanism used in the muling and upload protocols is well known we omit any evaluation of this mechanism per se. Rather, we aim to characterize the opportunistic sensor networking environment provided by BikeNet. In Figures 24, 25 and 26, we show results from a multi-bicycle experiment where each cyclist follows a prescribed path; the paths intersect giving rise to inter-bicycle muling opportunities. The paths followed by each of the four bicycles used in the experiment is around the perimeter of the Green field in front of Baker Library at Dartmouth College. The size of the Green field is approximately 150 meters by 100 meters. Two cyclists ride clockwise around the green field and the other two cyclists ride counter clockwise. The transmission range of the sensor motes are less than 50 meters so the connections among the cyclists are not always present but there are transmission opportunities when the cyclists who are moving the oppposite direction pass by each other. After ten minutes of circling around the green field, each cyclist leaves the green field one by one with fifteen minutes interval and parks the bicycle within the radio range of the SAP installed at the Sensor Systems Lab in the Computer Science building which is 250 meters away from the northeast corner of the green field.

In Figure 24, we show the number of data packets directly uploaded and the number of data packets muled to the SAP. The x-axis of the plot is the id of the sensor motes. The node id from 1 to 6 belongs to bike one, 11 to 16 belongs to bike two, 21 to 26 belongs to bike

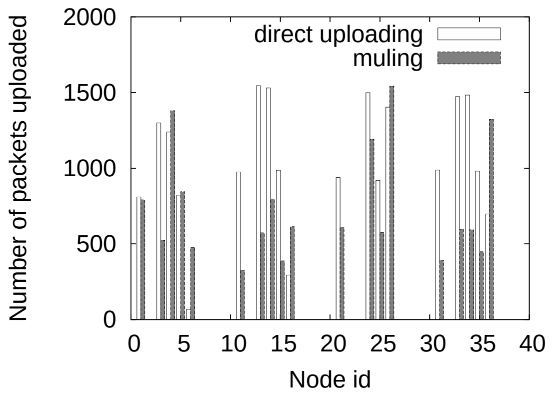


Figure 24: The Green. Muling/upload splits.

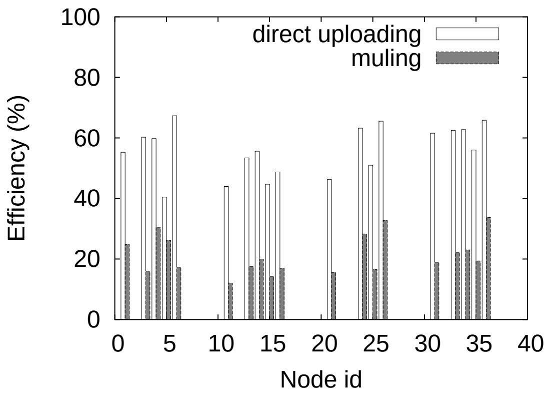


Figure 26: The Green. Delivery efficiency.

three, and 31 to 36 belongs to bike four. This x-axis is the same for Figures 25 and 26 as well. Overall, more data packets are directly uploaded than muled. However, a considerable amount of data are muled.

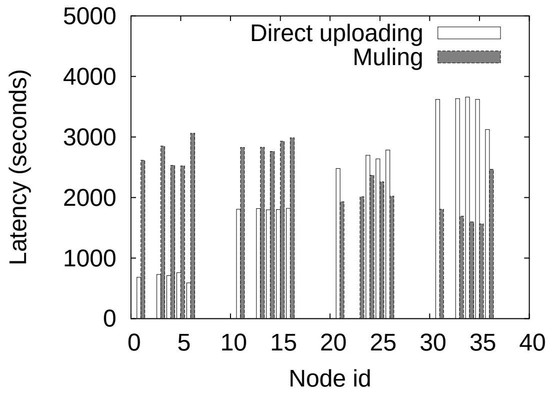


Figure 25: The Green. Delivery delay.

Figure 25 presents the latency of the direct uploading and muling. We measure the latency as the time difference from the time the sensor data packet is generated to the time the packet is uploaded to a SAP either by the originator of the packet or by a mule. The result clearly shows the benefit of muling. The sensor readings from bike three and four were muled by the motes on bike one and two which entered within radio range of the SAP earlier than bike three and four. As a result, the sensor readings from bike three and four gets less latency with muling than direct uploading. Especially, bike four gets on average 1500 seconds less latency with muling than direct uploading. If we assume one of the bikes never went within the radio range of the SAP, the benefit is not only the improvement in latency but the enabling of the data acquisition form the bike.

The muling comes with cost. The cost of muling is that it requires transmission from the origin of the sensed data to a mule, and then from the mule to a SAP. Figure 26 illustrates the transfer efficiency for both muling and direct uploading as a function of the packets transferred. Here we define efficiency as the ratio of data bytes transferred to the total bytes sent (including data packet replications and retransmissions). A higher efficiency for a given transferred packet reflects a more stable, less congested connection between sender and receiver. The sensed data origin always stores a copy in its local Flash hoping to upload the data to a SAP. It also can transfer a copy of the data to another mobile sensor via the muling protocol. In our experiments, the muling replication degree is one; we do not allow multi-hop muling. Figure 26 shows that the muling efficiency is less than uploading efficiency as expected. While the uploading efficiency ranges from 40% to 68%, the muling efficiency ranges from 12% to 33%. The difference between the muling efficiency and the uploading efficiency represents the cost for improving latency and enabling data acquisition from the bikes that rarely enter within the radio range of a SAP. We are currently studying the effect of replication degree on performance; we expect to see that more replication leads to improved delivery delay performance, but also lower efficiency.

## 4.4 Dartmouth BikeNet Testbed

We built five sensor bikes and implemented a small-scale BikeNet testbed with seven static SAPs at a number of points across the Dartmouth College campus and in the town of Hanover, as a means to validate and evaluate an operational BikeNet system. The system was initially built and debugged in the summer 2006 and is now operational [5]. In what follows, we present results from data collected by a group of three cyclists on the morning of November 20, 2006. We have collected a significant amount of data from over 50 different BikeNet experiments starting in summer 2006 but here only present data from a single-shot experiment with the three cyclists. The three cyclists routes and the times they started their rides is pre-planned. Cyclists 1 and 2 live near each other and ride much of the way toward Dartmouth Campus from the town together. Before getting to the campus they rendezvous with cyclist 3 before cyclists 1 and

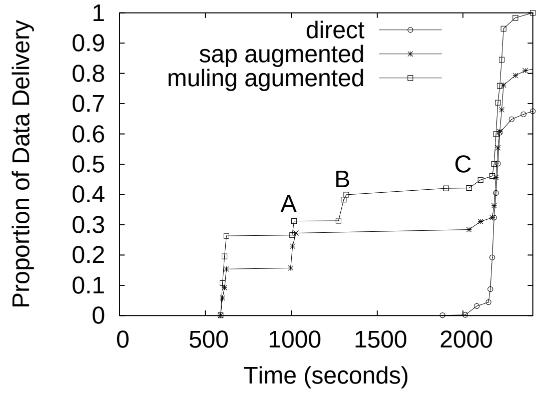


Figure 27: Rider Centric Delivery

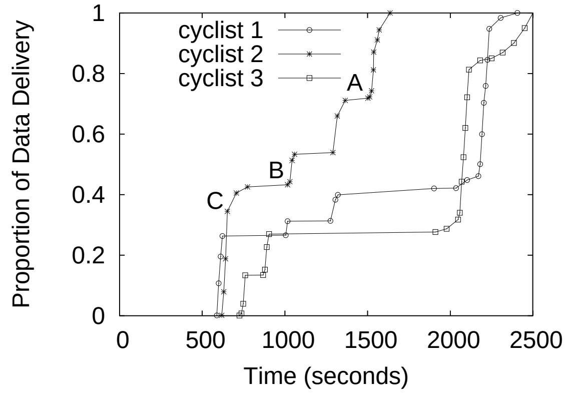


Figure 28: BAN direct Delivery

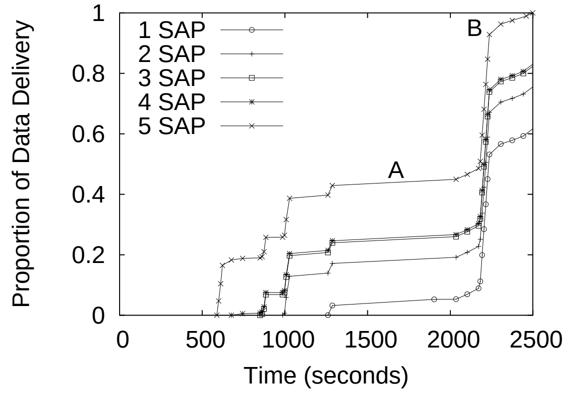


Figure 29: Multi-SAP Delivery

3 depart toward the library while cyclist 2 heads to the Computer Science building. The longest journey time for a cyclist is 40 minutes. The results presented in this section provide insights into how live sensor data collected by each of the bikes is either muled or directly uploaded to a passing SAP, in an opportunistic manner.

We first consider the time taken for each cyclist to upload its data into the backend data repository via direct upload to a SAP and present the results in Figure 28. Note that in this case the data uploaded to a SAP by a cyclist includes its own sensor data and any data muled by it. Figure 28 shows the "proportional data delivery" over the journey time. We define proportional data delivery as the amount of data received by the backend as the journey proceeds (e.g., at 520 seconds into the experiment cyclist 2 has delivered 40% of its data including any data it may mule on behalf of cyclists 1 and 3). From the plot we can observe that there is a noticeable period of time before any data is actually delivered. This is due to the lack of SAPs along the route from the homes of cyclists 1, 2 and 3. Cyclists 1 and 2 recounter a SAP along Main Street in Hanover at approximately 520 seconds into their ride. Even though only a modest number of SAPs are deployed we can see from the plot that all bikes are capable of delivering sizable portions of their collected data before their journeys end. For example, cyclists 1 and 2 deliver approximately 50 % and 40 % of their data to the backend depository before the half way stage of the route, respectively. From the plot we observe that data is delivered between bikes and SAPs in a quick burst of data transfer. As discussed before the amount of data transfered in strongly influenced by the short contact times which are a product of the short-range radios used by the BikeNet experiment.

Figure 27 shows the delivery of sensor data that is generated by a single cyclist only - in this case cyclist 1. We consider three scenarios of possible transfer between cyclist 1 and the data depository: *direct*, which is where cyclist 1 keeps all its data and does not mule and only uploads it at the final destination SAP in the Computer Science building; *SAP augmented*, which is where cyclist 1 opportunistically transfers data to SAPs it encounters along the way but also in this scenario no muling of cyclist 1's data occurs; and finally, *muling augmented*, which exploits muling and opportunistic use of SAP to transfer cyclist 1's data to the depository. Figure 27 shows the performance of these three types of communication. From the plot we can observe direct benefits of muling: cyclist 1 is disconnected for approximately 15 minutes between points A and C as shown on the plot but if we look at the "proportion of data delivery" at the intermediate point B we can observe that cyclist 1's delivery performance increases at point B yet it is disconnected. This shows the benefit of muling.

Next, we evaluate the impact of SAP availability (which are opportunistically encountered en-route) on the data delivery latency of the sensed data from the three cyclists to the backend data repository. In this experiment, we consider the incremental increase in available SAPs (i.e., from one SAP to a maximum of five SAPs) that are encountered by the three cyclists. By incrementally making available SAPs we study the "proportion of data delivery against time for five independent runs of the three cyclists from their respective homes to the Computer Science department. Figure 29 shows the results for a truncated time series up to the point when

100% of data is delivered by all cyclists for the case of 5 SAPs; the plot shows that when the complete data delivery for five SAPs occurs only 78% and 70% of data is delivered to the backend for the cases of three SAPs and one SAP, respectively.

Figure 29 shows that when the cyclists return to the Computer Science building they become stationary and upload their remaining data, which, is denoted by steep step in the curve at point B representing a large delivery of the final data from the bicycle to the SAP. In contrast, the flat portions of the curves, e.g., denoted by point A, represent a period when cyclists 2 and 3 are disconnected from the network with no other BikeNet nodes acting as mule data.

From the plot, we can also observe that the addition of a new SAP does not always result in a similar incremental improvement in the data delivery performance; for example, it is clear when inspecting the plot that the difference in the incremental improvement in the "proportion of data delivery for each of the curves in non-uniform. The performance increase of adding SAPs is highly dependent on many factors including the density, location of SAP in relation to routes traveled and the mobility of cyclists.

## 5 Related Work

Mobile sensing systems have been proposed in prior application contexts. Zebranet [1] was one of the first of these types of systems and was deployed to monitor Zebra populations in their natural environment. More recently the Cartel project [3] from MIT has deployed a mobile sensing platform based on exploiting cars and open wi-fi access points in a city. SATIRE [2] presents work more directly focused on people centric applications by proposing an general software architecture for smart clothing.

Hobbyists have assembled elaborate sensor-laden bicycles for fun, the earliest and most elaborate being the Winnebiko [22] which had multiple embedded and laptop computers, several VHF and UHF radio modem network links, and a variety of onboard sensors including GPS, security system, video camera, altimeter, battery monitors, and more. Like the current commercial sensor systems for bicycles mentioned in the introduction, the wireless communication in these systems is either strictly local to the bicycle or provides a link to the internet. None of these systems have explored bike-to-bike or person-to-bike communications.

Wireless sensors have long been used in sports to monitor performance, for example in training for the Olympics. The UWEN project [23] and the Sesame Consortium [24] are bringing new technologies such as UWB localization and advanced sensing, combined with improved biomechanical models of the human body, into play. Wireless sensors are also starting to be used to enforce the rules of contest in the martial arts [25] and other sports contests. These systems focus on the individual and are not used to network data between individuals.

There is much interest in monitoring first responders at the scene of an emergency using wireless sensors to alleviate some of the difficulties in their working environment [26]. These systems include peer-to-peer sharing of data so that, for example, a medic can follow a SWAT team into action while monitoring their lifesigns. They tend to be intolerant of delay and require high reliability, accuracy, and connectedness which comes at high cost.

The BikeNet "rock" has some similarities to collaborative augmented reality markup of which Websigns [27] is an example. However BikeNet's augmentation is via locally sensed and shared information rather than human authored markup.

The Wearable Computing and Personal Area Networking (PAN) fields have produced numerous examples [28], [29], [30], [31] of wireless networks that operate on and near the human body and which interact with the wearers surroundings or other peoples PAN's. There has also been much work in delay tolerant networking [32], [33], [34] to improve data transfer in networks that are often disconnected (as BikeNet is). BikeNet is a synthesis of all these ideas, using opportunistic rendezvouz of human wearable PAN's and bicycle PAN's (both human-to-bike and bike-to-bike) to add new dimensions to the bicycling experience.

## 6 Conclusion

In this paper we have presented detailed design, implementation and evaluation of the BikeNet mobile sensing system. BikeNet is a first experimental testbed for exploring the opportunistic sensor networking principles and techniques of the MetroSense people-centric sensing architecture. Our initial results are encouraging and demonstrate some of the value that ubiquitous wireless sensor networks can bring to our lives. We plan to further expand BikeNet to more deeply investigate the sharing and opportunistic aspects of personal sensing, and also the effects of radio propagation dynamics caused by mobility and interaction with the human body. We will also be exploring the value that BikeNet can bring to a community as a platform for large scale sensing and scalable muling of sensor data. While our current experiments have concentrated on sensing for the cyclist, bicycle mounted sensors could also serve a community, using mobility for spatial coverage in urban areas.

## Acknowledgment

The work is funded by the Institute for Security Technology Studies (ISTS), Dartmouth College, Intel Corp., and Nokia. The authors thank X. Zheng and H. Lu for their help with data collection.

## 7 References

[1] T. Liu, C. Sadler, P. Zhang, M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with Impala and ZebraNet. In *Proc. of MobiSys '04*, pages 259–269, Boston, MA, USA, 2004.

[2] R. Ganti, P. Jayachandran, T. Abdelzaher, J. Stankovic. SATIRE: a software architecture for smart AtTIRE. In *Proc. of MobiSys '06*, pages 110–123, Uppsala, Sweden, 2006.

[3] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *Proc. of 4th Int'l Conf. on Embedded Networked Sensor Systems*, pp.125–138, Boulder, Nov 1-3, 2006.

[4] CRAWDAD: A Community Resource for Archiving Wireless Data At Dartmouth. `http://crawdad.cs.dartmouth.edu/`.

[5] BikeNet Web Portal. `http://www.bikenet.cs.dartmouth.edu/`. username: Emiliano, password: bikes

[6] K. Fall. A delay tolerant network architecture for challenged internets. In *Proc. ACM SIGCOMM, pages 27–34, Karlsruhe, Germany, Aug. 2003.*

[7] The Bio Mapping Tool. `http://www.biomapping.net/`.

[8] Beating Heart Blog. `http://turbulence.org/Works/beatingheart/blog/`.

[9] Nike + iPod. `http://www.apple.com/ipod/nike/`.

[10] Carmichael Training Systems. `http://www.trainright.com/folders.asp?action=display&uid=2348`.

[11] CicloSport HAC5. `http://www.ciclosportusa.com/hac.html`.

[12] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. People-Centric Urban Sensing. In *Proc. of the 2nd Annual Int'l Wireless Internet Conf.*, Boston, Aug 2-5 2006.

[13] VeloTrend BikeBrain system. `http://www.bikebrain.com/`.

[14] R. Grisso, R. Oderwald, M. Alley, and C. Heatwole. Precision Farming Tools: Global Positioning System (GPS). Virginia Cooperative Extension, Publication 442-503, 2003.

[15] Moteiv Tmote Invent and TinyOS Boomerang package. `http://www.moteiv.com/`.

[16] ChipCon CC2420 Radio. `http://www.chipcon.com/`.

[17] Garmin Emap, Garmin International, Inc. `http://www.garmin.com/products/emap/`.

[18] Garmin Edge 305, Garmin International, Inc. `http://www.garmin.com/products/edge305/`.

[19] P. Levis, D. Gay, and D. Culler. Active Sensor Networks. In Proc. of 2nd USENIX/ACM Symp. on Networked System Design and Implementation, 2005.

[20] Adventure Planner, Trimble Outdoors. `http://www.trimbleoutdoors.com/biking.aspx`.

[21] TinyOS. `http://www.tinyos.net/`.

[22] Steven Roberts N4RVE. The winnebiko 1, winnebiko 2, and behemoth expeditions (1983-1991), and microship (1992-2002) expeditions. *http://microship.com/index.html*, 1983.

[23] I. Oppermann, L. Stoica, A. Rabbachin, Z. Shelby, and J. Haapola. Uwb wireless sensor networks: Uwen - a practical example. *IEEE Communications Magazine*, pages S27–S32, 2004.

[24] Stephen Hailes, George Coulouris, Andy Hopper, Alan Wilson, David Kerwin, Joan Lasenby, and Dipak Kalra. Sesame consortium. In `http://www.sesame.ucl.ac.uk/index.html`, Started April, 2006.

[25] Ed H. Chi, Jin Song, and Greg Corbin. 'killer app' of wearable computing: Wireless force sensing body protectors for martial arts. *In Proc. of 17th Annual ACM Symposium on User Interface Software and Technology*, pages 277–285, 2005.

[26] David Malan, Thaddeus Fulford-Jones, Matt Welsh, and Steve Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. *International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[27] Salil Pradhan, Cyril Brignone, Jun-Hong Cui, Alan McReynolds, and Mark T. Smith. Websigns: Hyperlinking physical locations to the web. *Computer*, 34(8):42–48, 2001.

[28] Mark Feldmeier and Joseph A. Paradiso. Giveaway wireless sensors for large-group interaction. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1291–1292, New York, NY, USA, 2004. ACM Press.

[29] Rich DeVaul, Michael Sung, Jonathan Gips, and Alex "Sandy" Pentland. Mithril 2003: Applications and architecture. *ISWC '03: Proceedings of the 7th IEEE International Symposium on Wearable Computers*, 2003.

[30] Thomas Zimmerman. Personal area networks: Near-field intrabody communication. *IBM Systems Journal*, 35(No. 3,4), 1996.

[31] B. Gyselinckx, C. Van Hoof, J. Ryckaert, R.F. Yazicioglu, P. Fiorini, and V. Leonov. Human++: autonomous wireless sensors for body area networks. In *Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005*, pages 13–19, 2005.

[32] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: A networking architecture designed around mobile users. *IFIP WONS*, 2006.

[33] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In *Proc. of MobiSys '04*, pages 111–124, Boston, MA, USA, 2004.

[34] Y. Wang and H. Wu. Dft-msn: The delay fault tolerant mobile sensor network for pervasive information gathering. In *Proc. of INFOCOM 2006*, April 23-29, Barcelona, Spain, 2006.

[35] San Diego Wind Tunnel Testing. `http://www.multisports.com/windtunnel_camp.shtml`.

[36] Nokia N80 Mobile Device. `http://www.nokia.com/nseries/index.html`.

[37] Lance Armstrong cycling cadence. `http://en.wikipedia.org/wiki/Lance_Armstrong#Riding_style`.