People-Centric Urban Sensing

Andrew T. Campbell,* Shane B. Eisenman,† Nicholas D. Lane,* Emiliano Miluzzo,* Ronald A. Peterson*

*Computer Science, Dartmouth College
Hanover, New Hampshire, USA
{campbell,niclane,miluzzo,rapjr}@cs.dartmouth.edu

†Electrical Engineering, Columbia University
New York, New York, USA
shane@ee.columbia.edu

ABSTRACT

The vast majority of advances in sensor network research over the last five years have focused on the development of a series of small-scale (100s of nodes) testbeds and specialized applications (e.g., environmental monitoring, etc.) that are built on low-powered sensor devices that self-organize to form application-specific multihop wireless networks. We believe that sensor networks have reached an important crossroads in their development. The question we address in this paper is how to propel sensor networks from their small-scale application-specific network origins, into the commercial mainstream of people's every day lives; the challenge being: how do we develop large-scale general-purpose sensor networks for the general public (e.g., consumers) capable of supporting a wide variety of applications in urban settings (e.g., enterprises, hospitals, recreational areas, towns, cities, and the metropolis). We propose *MetroSense*, a new people-centric paradigm for urban sensing at the edge of the Internet, at very large scale. We discuss a number of challenges, interactions and characteristics in urban sensing applications, and then present the MetroSense architecture which is based fundamentally on three design principles: network symbiosis, asymmetric design, and localized interaction. The ability of MetroSense to scale to very large areas is based on the use of an *opportunistic sensor networking* approach. Opportunistic sensor networking leverages mobility-enabled interactions and provides coordination between people-centric mobile sensors, static sensors and edge wireless access nodes in support of opportunistic sensing, opportunistic tasking, and opportunistic data collection. We discuss architectural challenges including providing sensing coverage with sparse mobile sensors, how to hand off roles and responsibilities between sensors, improving network performance and connectivity using adaptive multihop, and importantly, providing security and privacy for people-centric sensors and data.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design, Experimentation

Keywords

Urban Sensing, People-centric, Mobile Wireless Sensor Networks

1. INTRODUCTION

To date, the bulk of work in the wireless sensor network space has focused on environmental, agricultural or industrial monitoring. Networks of static sensing elements are either physically placed or randomly distributed across a target area of interest, with a focus on application-specific deployments. A substantial body of literature exists addressing a wide spectrum of issues in such sensor networks. Such networks are of utility and offer research challenges to scientists and engineers, but do not currently directly benefit the general population. Furthermore, humans are disengaged bystanders in the sensing and communication processes, passively waiting on the fringe of the network for data to appear. In this paper, we move away from the traditional focus of wireless sensor networks and propose a new people-centric sensing paradigm for urban sensing at very large scale. While traditional sensor networks target remote and unattended deployments we target the urban setting, an environment possessing a rich diversity of lifestyles, activities, and thus potential applications.

As our focus is on enabling human-centric applications, requirements on the architectural solution include the ability to sense people and characteristics of their immediate surroundings, and the ability to sense data related to interactions between people and interactions between people and their surroundings. These requirements are made more challenging by human and vehicle mobility (e.g., cars, buses, bikes). Furthermore, the people-centric nature of the data collected implies a requirement for privacy beyond what is present in traditional wireless sensor network application targets such as forest microclimates. In addition, the urban environment presents a host of challenges absent, or only partially present in other sensing domains. The architectural solution must scale at least across a large metropolitan area, must be able to handle a diversity of hardware platforms, application heterogeneity, interactions between a

multitude of administrative domains, and a highly dynamic environment.

To meet these requirements and challenges a number of architectural alternatives are possible, including the Ubisense [1] approach (i.e., a provisioned, managed, single-hop hierarchical UWB networks targeted towards enterprise installations), tiered sensor networks, and sensor meshes. However, by considering the cost and fidelity trends of each as the network scales to metropolitan dimensions, we see that these approaches are not feasible. Cost includes the monetary expense of sensors and network access (e.g., gateways, sensor readers) and infrastructure (e.g., running wires to deployment points) to deploy the sensor network. Fidelity refers generically to the amount of data collected from the sensor field and delivered to an interested user (i.e., information sink) within a given time period. The exact notion of fidelity is application-specific. For example, for event-based applications fidelity can mean receiving enough packets to detect features of interest in a given event. For periodic monitoring applications, fidelity can be characterized in terms of data stream continuity. Figure 1 illustrates qualitatively how existing networks scale in terms of cost and delivered fidelity if deployed at increasing scale. Ubisense [1] represents the most costly deployment scenario but might continue to provide high fidelity as the network scales up. However, the cost of doing this is prohibitively high when considering complete coverage across an urban area. In addition, the Ubisense [1] platform is not programmable and only supports one sensing modality. Tiered static mesh networks such as ExScal [2], Tenet [3], and Siphon [4] all add hierarchies of powerful nodes with secondary radio overlays (e.g., WiFi). This allows the network designers to better provision the network to offer better fidelity in comparison to non-tiered multihop networks. However, the coverage and communication cost of deploying these static sensor and WiFi overlay networks is also considerable for large areas. In addition, while multihop WiFi networks offer considerably more bandwidth than sensor radios they still lead to similar scaling problems albeit at higher throughput. Non-tiered mesh networks present a lower investment cost but the fidelity of these network does not scale because of link unreliability [5], congestion [6], and the funneling effect implicit [4] in many-to-one multihop sensor networks.

We propose MetroSense, a network architecture for urban-scale people-centric sensing with a design goal of broad application and sensor heterogeneity support. MetroSense provides the network architecture that is lacking in current urban-scale pervasive systems. MetroSense has a symbiotic relationship between itself and the human community it serves, leveraging existing infrastructure and human mobility to opportunistically sense and collect data about people for people. MetroSense trades off real-time fidelity for improved cost and coverage, enabling sparse sensing across large areas using potentially very large communities of rechargeable people-centric mobile sensors (e.g., motes [7], sensor-enabled cell phones [8]). In so doing MetroSense hits the architectural sweet spot (see Figure 1) that provides reasonable fidelity at low cost at urban deployment scale. The network architecture is designed with the specific requirements of urban sensing in mind; it is not intended to be a general purpose communications infrastructure, but rather an extension from it - *a new sensing edge network for the Internet.*

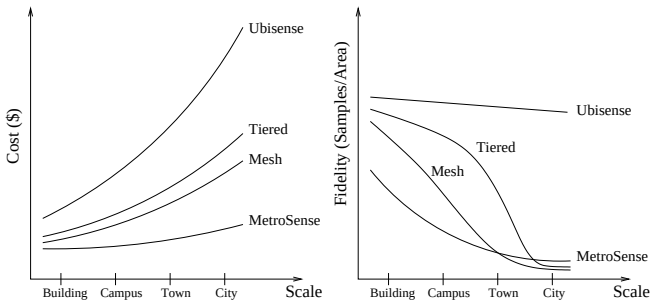

Figure 1: The Cost/Fidelity vs. Scalability Design Space

MetroSense assumes the ability of sensors to be recharged regularly like cell phones or PDAs of today. Thus, in comparison to embedded static sensor meshes and MANETs, MetroSense is less energy constrained and therefore targets a network lifetime of years rather than weeks or months. Furthermore, unlike static sensor meshes that leverage deep multi-hop, MetroSense adaptively limits multi-hop radio interactions in an effort to reduce complexity and wireless packet loss. MetroSense enables the general purpose programming of the infrastructure with the goal of supporting the execution of multiple applications in parallel across the network. The ability of MetroSense to scale to very large areas is based on the use of an *opportunistic sensor networking* paradigm. Opportunistic sensor networking leverages mobility-enabled interactions and provides coordination between people-centric mobile sensors, static sensors, and edge wireless access nodes in support of opportunistic sensing, opportunistic tasking, and opportunistic data collection.

In the next section, we outline our vision of the present and future of urban sensing, including descriptions of some practical leading edge applications whose requirements drive the design of MetroSense. In Section 3, we present an overview of the hardware and software architectural components of MetroSense, and discuss a number of design principles that underpin the MetroSense architecture. Section 4 provides focus on fundamental issues in the opportunistic sensor networking paradigm, namely, sensing coverage via mobility of sparse mobile sensors, selective responsibility transfer between sensors, sensor tasking and data collection. Section 5 addresses the important issue of security in a people-centric urban-scale sensor network. Section 6 discusses the gap between the work done by the sensor network and ubiquitous/pervasive computing communities that is bridged by MetroSense, before we conclude in Section 7.

2. URBAN SENSING

What is urban sensing? There are many elements that comprise the urban landscape, e.g., buildings, people, vehicles. We adopt a people-centric view of urban sensing where attributes of people, the immediate surroundings of people, and the way people interpret and interact with their surroundings become important. We view urban sensing as a departure from existing thinking on sensor networks because people are no longer just consumers of sensed data about some natural phenomenon or ecological process. Rather data about people is now sensed and collected such that the sets of producers and consumers of sensed data now overlap;

people are "in the loop" and may participate in both roles. In the following we outline our vision of the future of urban sensing, including likely interactions between elements of an urban sensor network. Issues that arise and the applications that are enabled as a result of these interactions are also discussed.

2.1 Present and Future Drivers

Currently, small areas of the urban landscape are sensed. Isolated deployments of private, static, application-specific sensor networks exist [9], focusing on the sensing of industrial or municipal infrastructure. For example, factories often monitor equipment health [10] as part of a preventative maintenance strategy. Municipal governments embed sensors in critical infrastructure to improve safety. For example, wired cameras are co-located with stoplights to detect/deter traffic violations and to support traffic congestion pricing strategies [11] and wireless vibration sensors are employed to monitor the structural integrity of bridges [12]. Such applications have successfully demonstrated the utility of wireless and wired sensor networking in the urban domain, but are designed to capture information about the infrastructure rather than people. Furthermore, existing deployments are mostly static and relatively small scale.

We feel this situation is likely to change, and that sensor data from a broader cross-section of the urban landscape will become available. Installations of traditional static wireless/wired sensor networks will continue to proliferate. Vehicles will be equipped with wireless sensors in an effort to capture information about air quality and traffic patterns. People will carry cell phones and other popular devices that are equipped with embedded sensors (e.g., temperature, acceleration, humidity) [8] and an ability to tag sensed data with location information. People may even carry key chain or vest pin wireless sensing platforms [7], motivated by the desire to be part of an interactive individual [13] or social [14] activity (e.g., real-time exercise analysis, sensor enhanced Instant Messenging), or to log the activities of a day for future playback [15]. Static sensing webs under different administrative domains will be integrated with a dynamic fluid of mobile sensors, forming a cooperative network. Depending on the application, sensed data will flow via gateways to back haul network infrastructure, or be processed in-network, possibly triggering immediate actuation (e.g., sound, light output).

2.2 Interactions and Characteristics

In an urban environment rich with sensor platforms of various types, with different mobility patterns, a number of interactions are likely to occur among these sensors, their custodians, their surroundings, and sensor network gateways to back end infrastructure or the Internet. In what follows, we list these interactions and their characteristics.

Static Sensor Peering. Following the paradigm of static wireless sensor networks, sensors are likely to establish long-term relationships to solve application-specific problems. For example, a static network may use a multi-hop routing tree to forward data to an information sink. In this case interactions are based on permanent physical proximity. Due to the static nature of the sensors and the resultant long term nature of relationships, extensive state exchange is feasible.

Mobile Sensor Peering. Sensor platforms carried by people are likely to interact as a result of interactions between humans themselves. The human interactions could be intentional like friends meeting according to plan, or unintentional like two people passing each other on the sidewalk or in the mall. The human interactions allow sensor interactions facilitating, for example, the exchange of sensed data between sensors for data muling [16], or the exchange of application/control information. Similarly, mobile sensors affixed to vehicles (e.g., cars, buses, bicycles) interact when vehicles pass each other on the roadway. People-mounted and vehicle-mounted sensors can also interact. A fundamental concern with mobile sensor peering interactions is the likely limited rendezvous duration that limits the amount of information that can be exchanged.

Static Sensor $\longleftrightarrow$ Mobile Sensor. A number of interactions are possible between static and mobile sensors. A mobile sensor can act as a data mule between isolated static sensors and a gateway. A mobile node can "visit" a gateway-anchored static sensor web in order to deliver sensed data or pickup application or configuration instructions. A mobile node can "reside" in a static sensor web for a time to participate in a collaborative application. However, recall that in general a mobile sensor platform has no control over the mobility of the human or vehicle custodian. Therefore, the terms "visit" and "reside" are qualitative descriptions of the interaction duration and do not imply intent on the part of the mobile sensor.

Static Sensor $\longleftrightarrow$ Gateway. Interactions between sensor network gateways and static sensors can be categorized as *tasking interactions*, or *collection interactions*. *Tasking interactions* arise when a sensor network user must convey instructions to the static sensor web about what activity to commence. Such instructions pass through the gateway. Sensor reprogramming, and the passing of system control messages fall into this category. On the other hand, sensed data must often be forwarded to a network user via the network gateway, either directly or indirectly via interactions with another static sensor or a mobile sensor. This activity is a *collection interaction*.

Mobile Sensor $\longleftrightarrow$ Gateway. Similar to the interaction between static sensors and gateways, interactions between a mobile sensor and a gateway include tasking interactions and collection interactions. Possibilities include delivery of sensed data for conveyance to an information sink, where the sensed data may be locally generated by the mobile sensor, or muled and delivered on behalf of another sensor. Programming, control and configuration operations also occur. Note that like other interactions involving a mobile sensor, the limited rendezvous duration implied by the mobility constrains the extent of possible interactions.

Gateway Peering. Gateways may be static or mobile. Static gateways, associated with static sensor webs, and mobile gateways form an overlay network that provides both the mobile and static sensors with a back end towards the Internet. Gateways interact, either directly or via an intermediary, to share information about collected data, and tasks currently running on mobile and static sensors.

Commonalities Across Interaction Classes. There are a number of practical issues that become important for all types of interactions. The owners/providers of sensing infrastructure are likely to be members of different administrative domains and may or may not trust each other. Furthermore, subsets of the sensing infrastructure may belong to economic competitors. Thus, the nature of interac-

tions between sensors may be affected by constraints beyond those imposed by physical proximity. For example, owners of static sensor webs and gateways may allow the "visiting" nodes just to share the sensed data with the local nodes, participate in the sensing activity of the "visited network" or become part of the visited network itself.

Aside from visitation rights, interactions between elements of the sensing infrastructure may require a set of resource discovery mechanisms so that the interacting principals can negotiate, for example, sensing roles in a peering application, or security parameters to use in subsequent communications. In fact, given the people-centric nature of the sensed data, protection, secure access and secure transport of data become main concerns of the urban sensing deployment.

Finally, achieving reliable transfer of data, whether it is related to tasking or sensed data collection, is of importance for all interactions between elements of the sensing infrastructure. Achieving adequate fidelity, in terms of the quantity of information delivered about an object or event to a sink, is an important challenge of people-centric sensing in the urban setting due to widespread mobility.

2.3 Enabled Applications

A city is more than buildings, people and vehicles. There are a large number of processes that govern a city and many ways of interacting with those processes. Those processes will change with time and location and can benefit from sensing and communications. Thus, wireless sensors will be woven into the fabric of existing processes as well as enabling new applications. It is too early to predict what application or class of applications may become popular or be of greatest utility or impact to society in the future. In what follows, we speculate on a mix of new urban sensing applications and how wireless sensor technology may be integrated with existing urban applications, both personal and public.

Personal Health Systems. By carrying/affixing one or more personal sensors people can monitor certain aspects of their own body. Sensors embedded in the home or public infrastructure can provide information about the environment surrounding people. Data collected from both personal body sensors and environmental sensors can be correlated to provide a trace of context-rich personal health information. This information can be monitored in real time or stored for longer term trend analysis. Particular applications include providing a digital memory assistant for the forgetful, recording activity/exercise level to estimate caloric expenditure [17] and joint stress, and detecting behavioral changes with health implications such as keeping the room warmer, stumbling more often, or a reduced activity level. A number of academic and industry projects have recently started up in this domain. These include Code Blue [9] and Alarm Net [18].

Pulse of the City. People like to know what's hot, where the crowds are, what's happening, what isn't. Data from mobile sensor nodes can be used to map crowds and infer activities. People themselves could also use a button on their mobile node to vote on the popularity of a location or event. How many people are watching the basketball game going on at "the Cage" at 4th Street/6th Ave street court where future professionals sometimes play? Is there an interesting street performer in Washington Square Park? Is that nut David Blaine's human aquarium stunt at Lincoln Center drawing a big crowd? How long is the queue for the Liberty/Ellis Island Ferry? Urban scale people-centric sensing can be combined with human activity inferencing to generate a general purpose location-based markup of a city, allowing various groups to map information of interest based on input from the people. Bar crawlers can mark the best bars, nirvana seekers can mark the best spots for mediation, and families can mark the best places to take children.

London Congestion Pricing. London Congestion Pricing [11] uses a network of wired cameras to sense a perimeter boundary around a central area of London, England. The goal of the system is to reduce automobile traffic congestion in this central area by charging drivers a fee to enter the central area during peak congestion periods. The system photographs the license plate of each car as it crosses the established perimeter. The image is processed to extract the license plate number, and levies the fee against the account of the car's owner. The system has been effective, markedly improving traffic conditions in the central area [11]. An evolution of this system might utilize wireless sensor technology to expand the scope and improve event detection, and might also provide finer-grained perimeter definition. For example, the wired camera network could be replaced with a network of wireless gateways, while cars could be equipped with smart tags. Tags could store status information about car and owner including inspection expiration date and registration status. Such a tag would replace the need for windshield/bumper stickers that currently display the same information. Tags could send periodic beacons, or respond to explicit queries to share information with the base stations as they near each other. This new system could handle not only the responsibilities of the existing system, but could eliminate the need for urban traffic policing by monitoring speed, illegal parking, expired inspection or registration and automatically issuing tickets to the owners of the car. Furthermore, unlike the current camera-based system, the wireless system would not be susceptible to visual obstructions such as pedestrians blocking the license plate.

Urban Planning and Usage Analysis. Assuming a future when both people and vehicles are equipped with mobile sensors, and wireless gateways are deployed at each intersection, a wealth of data is available. Aside from logging pedestrian and vehicular traffic flows [19], sound sensors can map noise levels in the urban environment [20] while air pollution levels can be mapped with more specialized sensors. Even with just these two sensors and the radio, analysis of collected data can be used to measure the wear and tear on infrastructure (roads, sidewalks), map urban area air pollution levels with fine spatial granularity and gauge how people "vote with their feet". Real estate businesses and municipal planners can make good use of the pedestrian and vehicle traffic flow data to learn of problem areas (e.g., congestion), evaluate business locations for foot and street traffic potential, and aid in establishing list value of property. Knowledge of air and noise pollution could aid buyers in choosing homes, and perhaps motivate municipalities to make quality of life improvements in neighborhoods that have consistently poor ratings.

3. METROSENSE ARCHITECTURE

Motivated by the promise and utility of the types of applications mentioned in the previous section we have designed the MetroSense architecture that meets the requirements (low cost, scalable, etc.) of the problem space and

takes advantage of the types of interactions between humans/sensors/environment that we envision. This section provides an overview of the physical and main software components included in MetroSense. We start by presenting a set of design principles on which MetroSense is founded.

3.1 Design Principles

Network Symbiosis. New sensing infrastructure and service deployment should leverage existing traditional networking infrastructure and services. The symbiosis between networks should be managed to maximize the benefit to the participants of all associated networks.

In a symbiotic relationship between two networks, at least one member benefits from the association. A sensor network architecture can benefit from the existing power and communications physical infrastructure, and from existing network services such as routing, reliable transport, and security. Paralleling the relationship between biological organisms, the symbiotic association between the sensor network and the traditional network may be parasitic, commensal or mutualistic. This means the traditional network may be injured, relatively unaffected or benefits, respectively, through the association. While there are inherent limitations in the services sensor networks can provide and in the processing and communications capabilities of the elements that compose them, we advocate mitigating these limitations through use of the capabilities of established networks to provide useful sensing services to the community. Relationships between networks and their constituents can be as defined by contractual policy or evolve in an ad hoc manner (e.g., via on-line cost/benefit analyses). Users of the established networks should experience minimal service degradation (e.g., through resource sand boxing) and may be provided service enhancements due to the association.

Asymmetric Design. Resource asymmetry that exists among members of the sensor network should be exploited by pushing computational complexity and energy burden to more capable nodes, while maintaining flexibility in the sensing applications that can be supported.

Network elements are classified into architectural tiers according to available computational, communications, sensing and energy resources. Service implementations should be aware of, and take advantage of such resource asymmetry that exists between the tiers by requiring the higher-capability tiers to handle tasks that consume more resources, or require a broader view of the network status. Leveraging resource asymmetry may result in sub-optimal process flow in the provision of a particular service or operation. However, we are willing to accept this sub-optimality for the benefits of a simplified service model and a network that is easier to manage. An architecture should push this trade off, while maintaining the ability to support the requirements of its target application classes.

Localized Interaction. Network elements should possess a highly constrained "sphere of interaction" within which they communicate with other network elements. We believe the loss of flexibility imposed by requiring localized interactions is outweighed by the increase in service implementation simplicity and communications performance.

Given the ephemeral nature of element rendezvous in a mobile network it is expensive and complex to maintain extended multihop interaction between elements. Even in a static network unlimited multihop is known to lead to poor communications performance. To keep interactions simple and efficient we advocate an adaptive multihop approach to communication, where a node's sphere of interaction grows and shrinks according to network conditions, rather than attempting to maintain continuous reachability of all elements. We rely on a probabilistic notion of reachability via opportunistic (mobility-enabled) interactions between nodes in the field.

3.2 Tiered Physical Architecture

While the MetroSense architecture does not define any particular set of hardware platforms, we do specify a three tier physical architecture based on a minimum required set of capabilities at each tier. We label these the *Server Tier*, the *SAP (Sensor Access Point) Tier* and the *Sensor Tier*. Members of the Server Tier are Ethernet-connected servers equipped with practically unbounded storage and computational power. These generic servers provide important service support to the architecture, as described in Section 3.3. In what follows, we describe the members of the lower architectural tiers, i.e., the SAP Tier and Sensor Tier.

SAP Tier. A SAP offers high performance, high reliability, and secure gateway access to MetroSense services for sensor tier elements while in range. SAPs provide secure, trusted interaction with the sensor tier. When sensor tier elements are not under a SAP then there are little or no such assurances given. A SAP performs the role of sensing, acts as a sink point for data gathered by the sensor tier and programs sensors by loading small application components onto them. Following the design principle of network symbiosis, we envision many SAP implementations will exploit readily available infrastructure such as WiFi access points, PCs/laptops, cell phones [8]. Network symbiosis raises a number of interesting challenges. For example, how can we exploit existing infrastructure without inhibiting or degrading its default operations? There is a need to design effective resource partitioning schemes to prevent the donor network being overwhelmed.

Sensor Tier. A mobile sensor is a wireless sensor device entrusted to a custodian, such as a person or a vehicle. These agents act as custodians of the device with the sensor device performing application functions as the agent moves within the sensor field. A static sensor (SS) is a wireless sensor device placed at a fixed location in the sensor field, typically to instrument infrastructure such as machinery or at specific locations to extend SAP coverage where wired SAPs are not possible. Mobile sensors (MSs) support sensing of the sensor custodian via applications run on behalf of the custodian or others, and provide sparse sensing via mobility. Examples of sensors in MetroSense are Zigbee-compatible motes [7] and sensor-equipped cellular phones [8] supporting a common protocol stack (e.g., Sensornet Protocol [21]. The key operations of a sensor are to collect sensed data and upload the data to a target data collection entity opportunistically. This is often a SAP, but could also be a sensor to which the SAP has delegated its collection responsibility for given data. Sensors also support muling [16] amongst each other to improve the delay and reliability of getting sensed data to a SAP and into the repository. Like any generic sensor used in sensing, sensors under MetroSense are capable of running peering applications such as tracking or other distributed multi-sensor applications where the sensors communicate on a hop-by-hop basis. MetroSense pro-

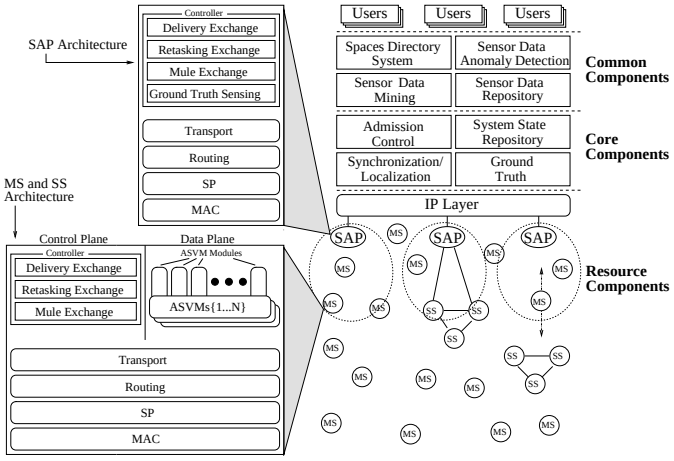

Figure 2: MetroSense Software Architecture

motes the use of small interpreted language applications [22] to reduce programming time. We believe like the Tenet [3] project that split applications may lead to greater efficiency for many applications. However, implementation of peering applications in this manner can be inefficient so we do not force this asymmetric execution model as Tenet does [3].

3.3 Software Architecture

At a high level, MetroSense can be viewed as comprising three classes of software components: *common*, *core*, and *resource*. To make clear when we are discussing implementation versus design, in the following we use the term *Metro* to describe an implementation and deployment of the MetroSense architecture. As such a Metro comprises a number of resource components each logically associated to a set of core components. Multiple Metros collectively can share the common components. A Metro tasks resource components to meet application requirements, with the core components internal to the tasking Metro providing management oversight. Common components are communal assets used to store and process the data output of sensing applications. The MetroSense software architecture is shown in Figure 2, where MS and SS represent mobile sensors and static sensors, respectively.

3.3.1 Core Components

The core components of a Metro are *admission control*, *system state repository*, *localization and synchronization*, and *ground truth*. These compose the central functionality of a Metro, enabling tasking and collection from the physical components, and supporting inter-Metro cooperation. Implementation of these services is distributed as appropriate across the physical architecture. Admission control supports a notion of fairness, and very loose service guarantees among multiple concurrent user applications. To support these features, admission control uses a hybrid soft/hard allocation of sensing resources (mobile sensors and static sensors), lazy tasking (see Section 4.3.2, and policy-driven resource reclamation. Furthermore, admission control protocols support execution of applications across multiple independent Metros. The system state repository contains both static (e.g., sensor physical address, sensor sensing capabilities, sensor custodian information) and dynamic (e.g., last known position of a mobile sensor, SAP load average, mobile sensor task set) state information. The information kept by the system state repository allows for proper allocation of mobile and static sensor resources to admitted applications, enables application fairness, facilitates inter-metro resource sharing, and supports a number of other network operations. The synchronization and localization component provides a consistent Metro-wide notion of time, and traces the mobility of each mobile sensor. Mobile sensor tracking allows for potentially smarter allocation of sensor resources to geographic sensing applications via mobility prediction. Sensed data is stamped with time and location information before storage in the sensor data repository so that sensed data is analyzed in both space and time. The "ground truth" data stream runs as a background process on each SAP to provide a steady trace of data from the SAP's on-board sensors collected from its own static field of view. Due to the wired nature of the SAP, collection of this ground truth data stream is nearly error-free. The data collected from the stream can be used by applications to augment data collected by mobile and static sensor resources. Furthermore, the Metro can use data from the ground truth stream for mobile and static sensor data integrity checking, sensor calibration, and sensor fault detection.

3.3.2 Common Components

The common components, shared across any number of Metros, are the *spaces directory system*, *sensor data repository*, *sensor data mining and sensor data anomaly detection*. These deal with name resolution and data storage and processing and in contrast to the core components possess few characteristics specific to sensor networks. In a Metro, users submit applications that collect data about spaces. A space describes the target of interest to the application (e.g., a geographic region, a demographic cross-section). Each Metro internally keeps a list of spaces it supports (e.g., defined by a system administrator). A record in this list comprises a descriptive string, and a list of resources (i.e., SAPs, static sensors, mobile sensors) that when tasked are most likely to return data consistent with the string descriptor. For example, a space entry that supports tasking of sensors whose custodians are University engineering students might look like {"University engineering students"; EngineeringBldg.SAP-1, EngineeringBldg.SAP-9, EngineeringBldg.SAP-10}. Internally, a query against a supported space string initiates tasking of the associated resources. Externally, the spaces directory system maintains a hierarchical mapping of space strings to the addresses of Metros that support the spaces (e.g., the IP address of the web server that accepts application submissions). The spaces directory system is updated in a like manner to the domain name system in the Internet. Users query the spaces directory system to discover which Metros support the spaces about which their application is designed to collect data. A more detailed usage case for application submission is described in Section 3.3.4. The sensor data repository provides a location for the communal amalgamation of collected data from one or more Metro. The sensor data repository may be realized by a physical centralized database, or virtually through the definition of appropriate mutual access policies to physically distinct databases maintained by each Metro. Such a communal resource provides the ability to users to analyze data from a broader set of results than would be otherwise pos-

sible. User access to particular data is a matter of user and repository policy negotiated by the constituent Metros. Sensor data mining and sensor data anomaly detection are services provided both to the core components and to the users. Sensor data mining implements a set of standard statistical functions a user can call by name to analyze repository data. Using these functions users can build interface applications to manage data retrieval and presentation. Sensor data anomaly detection implements a suite of algorithms in an effort to determine if a given data sample or set of samples lies outside the norm. This service is used, for example, by processes predicting data integrity that run in each Metro as part of the data collection process.

3.3.3 Resource Components

The resource components of a Metro are defined by the SAP and sensor software sub-architectures. Here we highlight primary elements of these sub-architectures with reference to Figure 2. MetroSense requires a programming model that supports multiple classes of applications, provides simple interfaces to application developers, whose runtime environment supports multiple simultaneous applications, and allows for light-weight tasking of sensors. Use of application specific virtual machines (ASVMs) [23] is one possible solution to meet these needs. These allow applications, transmitted as VM byte code, to remain small (tens of bytes). The approach provides a flexible and extensible environment for application developers, who may use instructions within an existing ASVM to construct applications, or can create ASVMs themselves tailored to the needs of a new class of application not previously supported. Three types of data exchange occur in the SAP and sensor sub-architectures: *retasking exchange*, *muling exchange*, and *delivery exchange*. In retasking exchange new application code is delivered from a tasking entity (e.g., a SAP) to a sensor; in muling exchange sensed data is transferred between sensors outside of the sphere of influence of the target data collection entity (e.g., SAP) to aid its progress; in delivery exchange sensed data is uploaded from a sensor to a target data collection entity (e.g., SAP). We present initial solutions to each of these activities in Sections 4.3.1, 4.3.2 and 4.3.3.

3.3.4 Usage Case: Running Applications in MetroSense

The MetroSense design is application agnostic and allows multiple applications to simultaneously share sensor resources. We believe use of an ASVM programming substrate on sensor resources is a promising approach to allow for flexibility and extensibility in developing and deploying applications.

Application Development and Submission. We envision that a standard set of ASVMs are supported by all nodes within a Metro. Developers are aware of these supported forms of ASVMs and can develop for the VM that is most appropriate for their requirements. Each ASVM is based on a master template VM that incorporates mandatory MetroSense "operations" (e.g., data buffer exchange, localization, synchronization, re-tasking exchanges, data muling exchanges, authentication and encryption). Once the application is written against the appropriate ASVM, a developer (more generally a user, post-development) deploys an application by constructing an application bundle and submitting it for execution, e.g., via a web-based portal. An application bundle is composed of a Space-time descriptor, a VM capsule and a Spec file containing the necessary parameters to fully specify the resource requirements of the application. The Space-time descriptor specifies the sensing target and the time window of interest. A space can have geographic-centric or people-centric semantics, or both. A User interested in sensing a particular space accesses the Spaces Directory System to determine the Metro or set of Metros that contain the sensing target of interest. If the target space is a single human, the resource requirements are unambiguous, the system must task the mobile sensor carried by that human, or sensors nearby. If the target space is "professors at Dartmouth", and the application is sensing the temperature of this space, the Spec file must specify that a temperature sensing instrument must be present on tasked sensor. Furthermore, the Spec file may specify a number of sensors to task, if the number is not implied by the Space definition. We are developing a small number of web-based user interface tools in support of application submission and a sensing description language to specify the requirements (e.g., space, time, tasking/sensing pre-conditions) of an application.

Admission Decision and Resource Selection. Admission control in MetroSense is based on target space validity and resource availability. If the space descriptor specified in the application bundle is not supported by the Metro then the application bundle is rejected and the user is notified. To check the availability of resources that match the requirements of the application, the Metro uses its internal Spaces directory to map the Space-time descriptor submitted in the application bundle to a list of candidate resources. These candidate resources (i.e., SAPs and MSs/SSs) are further filtered by additional requirements, such as sensor type, that may be included in the Spec file. To balance quality of service concerns with the ability to admit new applications, we use a hard/soft resource allocation negotiation. The system accepts the application if it has resources to meet the hard threshold immediately, and commit to trying to meet the soft threshold over time. Otherwise, the application is rejected with a notification of the current maximum hard threshold it can support. In this way, we aim to maintain system stability and provide a loose notion of fairness to multiple simultaneous applications. Because of the dynamism of the system, the admission and resource selection control is distributed across elements of both the Server and SAP tiers of the architecture.

4. OPPORTUNISTIC SENSOR NETWORKING

In MetroSense, we leverage the uncontrolled mobility patterns of mobile sensors, mobility that comes at no cost to the sensing/communication infrastructure, to bridge gaps in static sensor coverage. This mobility gives rise to a suite of opportunistic processes that facilitate urban-scale sensing. In this section, we discuss opportunistic sensing, opportunistic delegation, opportunistic tasking, opportunistic collection, and adaptive multihop.

4.1 Opportunistic Sensing

In order for a given sensing operation to be successful it is necessary that a particular sensor has the right instruments (e.g., temperature sensing device) for the required sensing

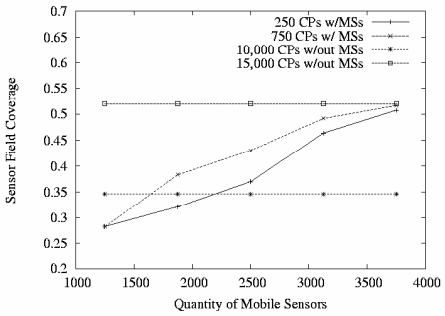

Figure 3: Coverage comparison of mobile and static sensor networks.

task, is loaded with the appropriate application, and has mobility characteristics that bring the sensor within the target area during the time window of interest. In an environment like MetroSense where most interaction between nodes is based on uncontrolled mobility we term the situation where the aforementioned requirements are met as *opportunistic sensing*. Strategies on how to task a sensor with a given instrument set to support a given application, and how to assign applications to sensors based on mobility characteristics are presented in Section 4.3. In what follows we discuss the ability of a network of mobile sensors to cover a sensor field so that properly tasked sensors may probabilistically be in the right place at the right time to sense the item of interest.

In our people-centric architecture, we are likely to have sensor coverage of our mobile sensor custodians, and swaths of coverage along their mobility paths. In contrast, a fully static network has a fixed coverage area determined by the initial deployment pattern. We conjecture that a sensor network founded on people-centric sensing, can provide a level of sensing coverage that can approximate that of a dedicated static network for many urban applications. In addition, we conjecture that without ubiquitous deployment (too expensive) static networks can not approximate the mobile sensing capability or fidelity of a people-centric sensor network. As an initial test of this conjecture we simulate (using NS-2) the complete Dartmouth Campus placing the data collection points on the actual positions of the Aruba WiFi access points we intend to use as part of our future SAP deployment strategy. We simulate a population of mobile sensors that periodically sense their environment (every 2 minutes) as they move around campus. The mobile sensors traffic patterns are based on actual traces from WiFi users on the Dartmouth Campus [24]. Mobile sensors deliver delayed data when they come across the simple data collection points. We unrealistically assume infinite buffering on each mobile sensor and a perfect wireless channel. We consider the sensing coverage area is 5m and the transmission range of each mobile sensor and data collection point is 40m. Figure 3 shows the percentage of the Dartmouth Campus area covered for varying numbers of mobile sensors (MSs) and collection points (CPs). It also shows two static sensor network deployments that are used as baseline comparisons, where 35% and 53% of the full campus area can be covered by 10,000 and 15,000 static sensors, respectively. Because NS-2 is not designed for such a large number of mobile nodes when using the wireless extension we ran the simulation for only a 20 minute simulation time. The important result shown in Figure 3 is that 750 collection points and 3750 mobile sensors are capable of covering the equivalent sensing area of a 15,000 static sensor network. The plot also shows the cross over point where a static network of 10,000 static sensors and {250 CPs, 2250 MSs} have roughly equivalent coverage after only 20 minutes. The observation from this simple simulation is that mobile sensors are effective at covering sensing areas over time and can match the coverage of static nodes with considerably fewer mobile sensors. A similar conclusion based on analysis for a random initial deployment strategy and a random mobility model is found in [25] However, what we do not characterize here is the delay cost. We are studying the relationship between coverage and delay through analysis and simulation, and plan to collect trace data from large-scale experimentation as part of future work.

4.2 Opportunistic Delegation Model

In what follows, we introduce the opportunistic delegation model used to coordinate the interactions between static and mobile elements in the network, allowing sensing to scale across very large networks in terms of both area and number of elements.

4.2.1 Model Definition

In any network architecture, principals have a set of designated responsibilities. Opportunistic delegation refers to the limited transfer of a subset of these responsibilities, when the transfer yields some advantage. Sensors may delegate responsibilities related to sensor tasking, and data collection. The transfer is limited in the sense that responsibilities are only delegated for a limited time or to perform a specific objective. The opportunistic element of the delegation is introduced by sensor mobility.

To make the notion more concrete, the following example presents a scenario where opportunistic delegation is used to extend the effective sensing range of a static sensor. Suppose an application requires data of type γ from region A of the field in the time interval $[t_1, t_2]$. Suppose, by some previous assignment, static sensor y has the responsibility to acquire this data, but its γ sensor range is such that region A is out of range. However, a mobile sensor z possessing a γ sensor exists with a motion vector v_z that intersects region A in the time interval $[t_1, t_2]$. Ideally z rendezvous with y prior to the intersection of v_z with A. In such a case, y delegates the target sense responsibility directly to z. Otherwise, "indirect delegation" can be used, whereby y delegates a third sensor w with the responsibility to task an appropriate sensor to execute the sensing. w, which can be a mobile or static sensor, in turn delegates the responsibility to sense the target to mobile sensor z. Such indirect delegation chains can grow as long as required. Assuming z is delegated to do the sensing in region A in time, it will acquire the γ data. At this point it is the responsibility of z to return the sensed data to the collection entity, say y. This responsibility can be fulfilled by z itself, or delegated to other sensors in a manner similar to the sensing responsibility just described.

4.2.2 Model Characteristics

The alternative to delegation is *direct responsibility*, where a sensor with a given objective does its best to achieve the objective without involving other participants. In what fol-

lows, we describe the advantages and disadvantages of opportunistic delegation in comparison to direct responsibility, with reference to the example presented in Section 4.2.1.

Sensing Coverage. Opportunistic delegation provides only a probabilistic notion of sensing coverage, due to its strong dependence on both the temporal and spatial aspects of mobility. In particular, in the context of the example from Section 4.2.1, a suitable sensor z (γ sensor equipped, v_z intersects A) for delegation must be available to y at the right time (relative to the intersection time of v_z with A) for the sensing delegation to be successful. Then, the data must make the return trip from z to the collection point, possibly using delegation. In contrast, with the direct responsibility approach, if the target region A is in range of sensor y the objective is successful (ignoring wireless channel loss for the sake of this discussion). Otherwise, y fails to acquire the desired data for the application. In cases where delayed data delivery is tolerable, the probabilistic notion of extended sensing coverage provided by opportunistic delegation provides an advantage over the direct responsibility approach.

Sensor Selection. In the previous example, sensor y may have several candidate sensors $\{z_1, ..., z_n\}$ for delegation of a given sensing task. In the general case, characteristics of the candidate sensor are not known to y, and consequently y may choose its delegate poorly. Section 4.3.2 discusses techniques that can be used to cull the candidates for delegation. However, if y is allowed to choose more than one delegate, the success probability of the objective goes up. In fact, when such multiple delegation is combined with delegate chaining, a larger platoon of sensors can be assigned any particular objective, increasing the fault tolerance of the data sensing and collection effort, if desired. A more typical case might involve a constant or linearly growing platoon. In contrast the direct responsibility approach assigns a single sensor or cluster of sensors responsibility to sense a given region; sensor loss has a permanent effect on sensing fidelity.

Sensor Data Collection. Opportunistic delegation in the data collection process, though semantically different, is functionally equivalent to data muling. As with muling, data delivery is likely to have a higher average delay with opportunistic delegation than a static network using direct responsibility for data collection. On the other hand, delegation in the collection process may enable collection where it is otherwise impossible (e.g., an isolated sensor). The best collection delegate at a given point is difficult to determine, yet the choice strongly impacts the ultimate success of the collection objective. Section 4.3.3 proposes a mechanism to improve the probability of success.

Sensor Data Fidelity. The impact of opportunistic delegation on the fidelity of the sensed data is related to the sensitivity of the sensing instruments (e.g., motion detector range) on the sensor. Mobile sensing delegates may be forced, due to mobile sensor custodian mobility, to acquire data at a distance from the intended target beyond the optimal sensitivity threshold of the sensing instrument. In this case the quality of the sensed data often decreases monotonic with the sensor's distance from the target. Static networks are often deployed to provide complete sensing coverage over all regions of interest. However, this is not feasible at urban scale. In this case, if the direct responsibility approach is followed, an out-of-range target region has no chance to be sensed and thus any data collected has null fidelity.

4.2.3 Opportunistic Delegation Primitives

Opportunistic delegation is a general purpose paradigm for mobile and static sensor coordination, using responsibility transfer to improve success probability for a given objective. The general model gives rise to a number of primitives of use in supporting applications that run in mobility heterogeneous networks. In what follows, we describe four primitives that we believe will be commonly adopted by applications.

Virtual Sensing Range. The ability to sense any arbitrary region within the sensor field is a fundamental requirement of a sensor network. Any mobile or static sensor has a sensing coverage area limited by the sensitivity of the sensing instrument. Application requests to acquire sensor data will sometimes exceed these limits. The opportunistic delegation example in 4.2.1 describes the interactions required to extend the sensing range of a mobile or static sensor to meet the application request. We refer to this extended sensing range as the *virtual sensing range*.

Virtual Collection Range. Some applications may impose a temporal filter on received data such that data received with a delay greater than the threshold is ignored. The *virtual collection range* primitive increases the probability of timely data collection by effectively increasing the size of the collection "target". The physical collection entity, e.g., a static sensor y, opportunistically delegates its collection responsibility to other sensors for a given data set. Upon collecting the data, the delegate then either directly or indirectly (using opportunistic delegation) communicates the data to y. Unlike the virtual sensing range which can be said to extend to the target region, there is no such equivalent limit to the virtual collection range.

Virtual Static Sensor. Many existing sensor network applications are designed for use in a static sensor network. The *virtual static sensor* primitive allows a completely mobile or hybrid mobile/static network to emulate a static network via opportunistic delegation. This primitive can establish an arbitrary number of virtual sensors at arbitrary locations. Virtual static sensors are realized by first partitioning the sensor field into subfields of interest at the resolution necessary to meet the requirements of the application. If a physical static sensor exists in partition p_i, it becomes the "virtual static sensor" for partition p_i. Otherwise, as mobility brings mobile sensors into partition p_i, these temporarily act as the virtual static sensor for p_i. Mobile sensors leaving p_i delegate the responsibility to function as the virtual static sensor for p_i to other sensors entering p_i, exchanging state as required, in a best effort manner. Data collection from virtual static sensors occurs via delegated collection, if necessary.

Virtual Mobile Sensor. In people-centric sensing, when the sense target is a mobile sensor custodian, depending on the type of sensor it may be easy to achieve sensor coverage of the target. When a human (or other mobile) target is not a sensor custodian the *virtual mobile sensor* primitive allows best effort coverage of the target. This primitive can be viewed as a generalization of the virtual static sensor primitive. Partitioning and delegation proceeds in a similar manner, except that with the virtual mobile sensor primitive a partition p_i moves around the field with the mobile target to which it is associated. The effectiveness of mobile sensor emulation is much more challenging than the static case, particularly because the motion of the mobile target is

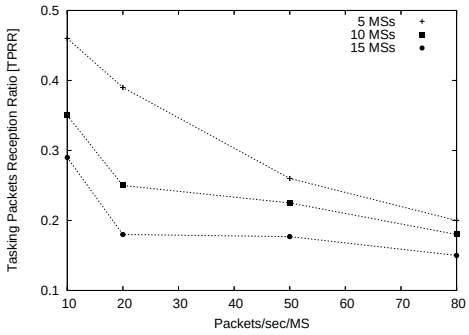

Figure 4: Tasking packet reception ratio (TPRR) versus load.

not known a priori and can only be predicted based on past history.

4.3 Opportunistic Tasking and Collection

Tasking of an appropriate mobile sensor for a given application by a particular SAP requires both that the mobile sensor moves within the sphere of interaction (e.g., radio range) of the tasking entity (e.g., a SAP, a mobile or static sensor that has been delegated the responsibility to task), and that the sensor remains within this sphere long enough for the tasking packet transfer to complete. Similarly, upload of a mobile sensor's data to a particular collection point (e.g., a SAP, a sensor acting as a data mule, a sensor with a multihop path to a SAP, a sensor that is in network end point or aggregator of particular data) requires both that the mobile sensor moves within the sphere of interaction of the SAP, and that the sensor remains within this sphere long enough for the data upload to complete. We introduce the terms *opportunistic tasking* and *opportunistic collection* to refer, respectively, to the methods by which sensor tasking and data upload can be completed in the face of uncontrolled sensor mobility. In this section, we discuss the effect of uncontrolled mobility on tasking and data collection operations and strategies to respond to this effect.

To evaluate the impact of sensor mobility on tasking and collection operations, we study the ability of a SAP network to authenticate, retask, and upload sensed data from groups of real people-centric sensors using an Aruba WiFi Access Point 70 as a SAP (running the OpenWRT customized Linux distribution [26]) connecting the Zigbee Sky Mote into the AP's USB port. A four SAP network on one floor of the Sudikoff Computer Science building at Dartmouth provides sparse coverage with a 30-40m coverage area per SAP. Data is collected for three experiments where mobile sensor groups (viz. 5, 10, 15) cross coverage areas at human walking speed at various trajectories. Mobile sensors upload traffic at 5 pkts/s while two static motes provide background traffic at various rates (viz., no, low, medium, high). Traces for each of the three group sizes and four loading conditions are collected. We use PSFQ [5], a reliable transport protocol, for authentication and retasking and unreliable communication for data upload to the SAP. Figure 4 shows TPRR, the ratio of tasking packets sent by the SAP that are received by the mobile sensors (MSs), against load for 5, 10, and 15 MSs. Even under no background traffic load the best results are only between 30-45% completion of the retasking activity. Once a sensor has authenticated (90% of sensors authenticate across all experiments) PSFQ attempts to reliably transfer an ASVM component (fourteen 36-byte packets). Under no load conditions only the 5-sensors group completes the retasking process. Increasing load gives poorer results. From these preliminary experiments we observe that sensor mobility reduces the likelihood that a sensor will complete a data upload operation (data omitted due to space constraints) or that a SAP will complete a tasking operation (Figure 4) in a single "session" before moving out of range, even for relatively small tasking payloads (e.g., ASVM application byte code). Mobility implies a relatively short rendezvous duration, changing multipath characteristics, and hidden terminal effects. Additionally, the relationship between body and sensor positioning has a mobility-dependent effect on radio signal attenuation due to shielding by human body parts. It is clear that a naive approach to tasking and collection when sensors are mobile is insufficient. This motivates the design of additional strategies to increase the probability that sensor tasking is successfully completed, the correct sensor is chosen for tasking, and that sensor data can be uploaded to the SAPs. Sections 4.3.1 through 4.3.4 discuss a number of promising strategies.

4.3.1 *Lazy Uploading*

Sensed data is collected in a delay tolerant manner. It is clear that depending on mobility patterns and buffering of muled and locally-sensed data mobile sensors may not be able to upload all buffered data to the target data collection entity in one session, i.e., before moving out of range. Furthermore, inter-contact times between mobile sensors and the target collection entity can vary considerably depending on mobility patterns [27]. Reliable upload of sensed data from the mobile sensor is thus problematic. Again, according to the particulars of the scenario, the target data collection entity could be a SAP, or a mobile or static sensor to which the collection responsibility has been delegated for given data. We introduce the notion of lazy uploading to facilitate sensor data upload across multiple collection entity visitations. When a mobile sensor enters a radio coverage zone (e.g., a SAP is the common case) of a collection entity zone it authenticates to establish a trust association. During a given established session, data upload occurs via a reliable point-to-point transport between the mobile sensor and the collection entity. Note that this could be tunneled over multiple hops in the case of adaptive multihop (see Section 4.3.4). We propose a transport protocol that uses a selective negative acknowledgment system with an unlimited window size. The collection entity with the support of the system state repository keeps state on the progress of the data transfer, allowing a mobile sensor to move away from a collection entity without worrying about first filling gaps in packet reception. When a mobile sensor authenticates with a new collection entity it receives a selective negative acknowledgment as part of its authentication process which allows it to continue its upload from the point where the system has received data, and requiring the mobile sensor to retransmit packets the system is missing. In this way, the need for any other wireless state exchange between mobile sensors and collection entities during the sensor data uploads is eliminated.

4.3.2 Lazy Tasking

In MetroSense, in the best case, the mobility of mobile sensors is such that reprogramming can be completed while the mobile sensor is in range of a particular tasking entity (e.g., a SAP is the common case). However, as discussed in Section 4.3, this will often not be the case. Therefore, MetroSense includes a lazy tasking mechanism similar to the Lazy Uploading strategy discussed in Section 4.3.1. The SAP tier and involved sensors keep state on active incomplete tasking operations should the tasking operation be incomplete when the mobile sensor leaves the range of the tasking entity. In this way, the architecture is able to "pause" and "resume" the tasking as a mobile sensor moves in and out of SAP coverage areas. According to the particulars of the scenario, the tasking entity could be a SAP, or a mobile or static sensor to which the tasking responsibility has been delegated. MetroSense uses a mobile sensor's direction, speed, and location with respect to the tasking entity with which the mobile sensor is currently interacting as an input to the candidate selection algorithm. This algorithm determines which mobile sensors to task and when, based on current application requests in the system. We have identified a number of heuristics for mobile sensor selection. For example, we might choose a momentarily stationary mobile sensor (longer rendezvous duration for programming) that will exit the tasking entity's zone of influence in the direction of the intended sensing space. Other policies include programming mobile sensors when they first associate with the tasking entity and allowing them to remove the task if they do not exit the zone of influence in the correct direction - we call this early tasking. A variant of early tasking has the mobile sensor informing the tasking entity that they are candidates for retasking. Here the tasking entity gives the mobile sensor the coordinates of the sensing space. Mobile sensors then elect themselves as candidates for retasking in an autonomous manner. An alternative is to wait until the tasking entity knows that the mobile sensor is a strong candidate based on its apparent exit trajectory - we call this late tasking.

4.3.3 Direction-based Muling

Mobile sensors are capable of picking up sensed data from static sensors that do not have a static single- or multi-hop connection to the target data collection entity (e.g., a SAP is the common case), and from other mobile sensors that it may encounter. The goal of muling in MetroSense is to get data to the target data collection entity in a timely manner. According to the particulars of the scenario, the target data collection entity could be a SAP, or a mobile or static sensor to which the collection responsibility has been delegated for given data. To improve performance we exploit the predictable nature of human mobility in a Metro (e.g., students walking across campus following a daily class schedule). In direction-based muling a mobile sensor determines prior to a reliable exchange which candidates are likely to be heading in the direction of the target collection entity. The direction-based muling transport works as follows. Once a source mobile sensor has sufficient data to mule (e.g., an event packet or set of event packets) it uses a mule discovery protocol. The protocol solicits muling services via rendezvous. It broadcasts a mule solicitation message with its data muling needs, its direction and speed and last location of the data collection point it encountered. Mobile sensors that hear the request only respond if they are either heading in opposite or orthogonal directions from the soliciting mobile sensor and they have sufficient buffering to carry the data. If this is the case then the candidate mules respond. If candidate mobile sensors overhear more than an a threshold number of responses they drop their response. A number of heuristics are possible to select the best mule candidate. For example, we may choose a mobile sensor whose motion vector compared to the advertising sensor gives the transport more time to complete. Alternatively, we may chose the sensor whose selection is likely yield the lowest delay data delivery given a predicted trajectory towards a SAP. Since there is no way to know exactly where mobile sensors may go in the future and where the target collection entities are, the best approach (with respect to reliability and timeliness of delivery) is to have multiple mules with the same data spread out in different directions. Many data muling proposals exist (e.g., exploring data replication strategies, choosing the best mule), a number of which are mentioned in the related work (Section 6).

4.3.4 Adaptive Multihop

Adaptive multihop is a mechanism that can improve the delay performance of both sensor tasking and data uploading by increasing the probability that these activities can complete in a single SAP session. Due to the increased complexity and probability of loss inherent therein, when considering a very large scale sensor network deployment we are strongly motivated to find solutions for sensed data transport and network retasking that do not rely on wireless multihop. However, under some conditions multihop may improve performance and/or enable necessary functions. When a mobile sensor's mobility pattern brings it within the radio reception range of a target data collection entity, data uploading may begin. Here, according to the particulars of the scenario, the target data collection entity could be a SAP, or a mobile or static sensor to which the collection responsibility has been delegated for given data. Single hop upload is preferable because of the reduced probability of packet loss. However, in sparse regions of a MetroSense deployment it may be advantageous to increase the size of the collection target by delegating collection responsibility to neighbors of the target collection entity. A collection entity whose average load (e.g., packets received per time interval, mobile sensors seen per time interval) remains below a value $Load.Threshold_{min}$ for N time epochs may attempt to grow its collection boundary by a single wireless hop. Several factors contribute to the decision of which collection entity neighbors become data collection delegates, including physical distance from the target collection entity, link quality with the target collection entity, and anticipated mobility pattern. For example, a target collection entity might select neighbors that are maximally distant from itself and each other, that possess link quality $> LQ.Threshold$, and that are likely to be stationary in the near future (e.g., $Mobility.Factor < Mobility.Threshold$).

On the tasking side, the requirements of a given application indicate which characteristics a mobile sensor must have to be a candidate for tasking with a given application. The pool from which candidates are chosen for tasking comprises static and mobile sensors within communication range of a tasking entity. Note that according to the specifics of the scenario, the tasking entity could be a SAP, or a mo-

bile or static sensor to which the tasking responsibility has been delegated. Normally, direct communication is desired since sensor tasking requires a reliable transfer of application code, and multihop wireless transfers increase the probability of packet loss. If requirements for a given application are particularly limiting, or if few mobile sensors happen to be within direct communication range with a tasking entity at a given time, it may be preferable to expand the search pool by relaxing the direct communication constraint. However, selecting a candidate from $Pool.expanded - Pool.initial$ introduces additional complexity because it requires a multihop tasking capability.

Other scenarios arise. Assume that a successful tasking for a given application requires the download of K packets from the tasking entity to the mobile sensor. If a mobile sensor moves out of direct range after $K - i$ packets have been successfully received the system must make a choice: purge state and start over with a new candidate; cache the state for the programming session and wait for the partially-programmed mobile sensor to come back into range to complete the download. Both of these options are unsatisfactory; the first wastes the resources of the tasking entity and mobile sensor already expended in partial programming, while the second ties up memory at the tasking entity. Allowing for multihop tasking provides the possibility to complete the download of the i remaining packets before the mobile sensor moves out of the multihop range of the tasking entity. Research questions here include how many hops to extend the tasking boundary, and which neighbors to choose as tasking delegates. Physical distance from the primary tasking entity, link quality with the primary tasking entity, and anticipated mobility figure prominently into tasking delegate selection.

5. PEOPLE-CENTRIC SECURITY AND PRIVACY

Given the people-centric nature of the data handled by MetroSense, protecting the security and privacy of the sensor tasking and data collection processes is of great importance. Initial work on providing security for static sensor meshes (e.g., [28] [29]) provides promising leads, but many open problems exist. Characteristics of sensing in the urban environment such as mobility and issues related to interactions across multiple administrative domains introduce further complexity. The overall security solution in MetroSense includes a number of general approaches, including: incorporating the use of trusted platform modules (TPMs) in the construction of sensors to provide hardware support for security; mechanisms to recover sensors once they are compromised; leveraging the capabilities of the SAPs and the server tier to detect problems; leveraging the people-centric nature of the network by including humans as part of the anomaly detection strategy; using human custodians to periodically refresh (e.g., plug in to PC) the sensor to a known good state; and managing data access via data creation, transport, and storage policies. Due to space constraints, we forego a treatment of these general approaches. Rather, in this section, we describe how the MetroSense brings one specific approach, the opportunistic delegation model (discussed in Section 4.2, to bear on the problems of security and data privacy. We first describe how the model offers a measure of protection by construction, followed by two active techniques that can provide additional assurances within the model's framework.

5.1 Intrinsic Security Properties of Opportunistic Delegation

The opportunistic delegation model offers forms of privacy and security protection due to its reliance on uncontrolled sensor mobility. Operations such as opportunistic tasking, opportunistic sensing and opportunistic collection thus lack the determinism of the direct responsibility model (discussed in Section 4.2.2. The uncertainty introduced by uncontrolled mobility results in an intrinsic robustness to potential attacks, and an increased notion of privacy, as described in the following.

Fault Tolerance. In a virtual static sensor grid no one single sensor is exclusively responsible for a specific geographic region p_i. Delegation occurs only as allowed by the unpredictable mobility of custodians. This results in a constant churn of responsibilities among sensors. Thus attacks that focus on capturing or otherwise limiting the capability of individual sensors to perform their responsibilities (e.g., sensing) have a limited effect on the sensor grid coverage. Performance degrades gracefully as a smaller set of mobile sensors opportunistically continue to cover region p_i in lieu of captured or disabled nodes. This is in contrast to a purely static sensor network where the ability to provide geographic coverage is tied to a specific set of sensors.

The same form of robustness extends to performing people-centric sensing when using the virtual mobile sensor primitive. By employing opportunistic delegation the ability to sense the individual is not tied deterministically to a specific mobile sensor but can be achieved opportunistically by the collective abilities of all sensors in the network.

Privacy. A mobile sensor custodian k may worry that her activities could be inferred based on a trace of the responsibilities her sensor assumes. The use of opportunistic delegation primitives acts to decorrelate responsibilities assumed by a particular sensor with potentially identifying characteristics of the sensor custodian (e.g., mobility pattern). Delegation depends not only on custodian k's mobility but also on the mobility of potentially all other custodians. This, combined with the ever changing set of requirements dictated by sensing applications, weakens possible inferences as to the exact activities of any particular custodian.

5.2 Security-aware Delegation

We envision an urban sensor network comprising a collection of heterogeneous sensing devices, each with potentially different capabilities to provide forms of privacy and security protection. Some devices have stronger means at their disposal for instance due to their location (e.g., a SAP located in a bank) or special hardware (e.g., incorporate TPM in a mobile sensor). However, a chain of security is only as strong as its weakest member.

To achieve a specified level of security, in the course of opportunistic delegation potential target sensors are filtered according their security capabilities. For instance, in establishing a delegation chain to perform data collection of particularly sensitive sensed data, delegation could be limited to only those sensors with trusted hardware. Given that the construction of such a chain relies on the opportunistic rendezvous of sensors that meet specified security requirements, performance in terms of delay is likely to worsen.

Furthermore, additional overhead is required to verify potential delegates have the necessary security characteristics (e.g., trusted hardware).

5.3 Privacy-aware Delegation

Privacy-aware delegation aims to limit the extent of information known about a particular sensor by inverting the opportunistic delegation procedure. In a typical opportunistic delegation interaction, a responsibility is delegated from a sensor x to a sensor y. In this case, x has information about the future objectives of y, and may be able to infer information about the custodian of y. This is particularly so if y happens to encounter x again in the future and shares information about the results of the previously delegated objective. One approach that reduces the information x knows about y is for the delegate sensor y to *choose* (e.g., at random) the responsibility it will assume (e.g., from a list of responsibilities outstanding at x) rather than x delegating the responsibility. This approach maintains a level of privacy at y but may seem to compromise that of x. However, this is not so if the list of responsibilities at x is also accumulated with the same privacy-aware delegation procedure.

Allowing the candidate sensor to choose the responsibilities has consequences for system performance. While increased signaling is required, the primary impact results from a possible tension between optimal responsibility delegation and custodian privacy. In particular, to increase the privacy of its custodian a sensor may choose a responsibility it is ill-equipped to fulfill while forgoing another that is may be uniquely positioned to fulfill. At critically large scale, such suboptimal system behaviour is probabilistically mitigated by the diversity of custodian privacy aims, though the fundamental trade off between privacy and performance remains.

6. RELATED WORK

To date, no published work presents a network architecture solution suitable for general purpose urban scale sensing. Proposals in the areas of tiered sensor networks, delay tolerant sensor networks, and sensor network and ubiquitous computing middleware architectures represent the most closely related work, and are discussed below.

Tiered Sensor Networks. ExScal [2] seeks to maintain delay performance at scale in classic static sensor networks using radio network tiering techniques. Others have proposed tiering (e.g., [4] [30]) to address different concerns in large scale static sensor networks. However, these proposals do not handle mobility and thus are not adequate to handle urban sensing. Tenet [3] is an architectural approach in which tiering extends even to the application execution model. While the strict approach is beneficial for some operations like aggregation, the rigidity of the model makes peering applications inefficient since all peer interactions are mediated by a higher tier proxy. We posit Tenet can not scale to urban area sensing because of the performance limitations imposed by the strict tiering model. Furthermore, [3] [2] [4] and [30] are all targeted at static sensor networks, and are thus inadequate to handle the dynamics of mobility in the urban sensing environment.

Delay Tolerant Sensor Networks. A number of projects propose the use of delay-tolerant networking via data muling within the context of sensor networks (e.g., [31] [32] [33]). However, proposals like [31] and [33] do so only to address partitions in the network with regards to the radio communications, neglecting the equally important challenges of sensor tasking and sensing coverage in the urban landscape. In [32], the authors discuss human-oriented sensing to improve fidelity, but present a limited solution space, exploring only efficient flooding variants. In MetroSense, communication, tasking and sensing are all addressed with a unified solution via the opportunistic interactions we have defined under the umbrella of what we call opportunistic sensor networking (Section 4).

Software Architectures. The software agent-based IrisNet [34], the Hourglass overlay model [35] and the stream-focused SONGS architecture [36] are representative of software architectures that attempt to address the challenges of coordinating data from heterogeneous sensor networks at large scale. These proposals begin with the assumption that the sensors and sensor networks necessary for application deployment and data collection already exist. Gaia [37], Aura [38], Endeavor [39] and Oxygen [40] are representative of the pervasive networks approach to systems architecture. These present middleware solutions to facilitate the dynamic negotiation between devices such as discovery and localization. The pervasive systems architectures again assume the presence of an enabling underlying network architecture for the urban environment at large scale. Achieving data collection of adequate fidelity in the urban environment is no small task, and this goal is not met with existing proposals. There is a need for new architecture solutions to fill this gap. MetroSense embodies a comprehensive solution that targets scalability of the network by proposing opportunistic sensing and communication infrastructure and mechanisms that are designed for scale, from the bottom-up.

Context-Aware Ubiquitous Systems. A number of people-centric pervasive sensing systems are notable successes. Examples of such work include, Sentient Computing [41], Active Badge [42], Ubisense [1], Cricket [43] and P3 Systems [44]. These projects focus on localizing people and objects in a defined environment to enable context aware applications. In such projects, the notion of sensing is confined to supporting location-based context-awareness; a more general architecture to support the diversity of applications and hardware platforms that are likely across a network spanning an urban environment is lacking.

7. CONCLUSION

We believe urban sensing is an important new research area with many interesting architectural challenges and problems to solve. Two principle trends support the move toward urban sensing, the integration of sensors into everyday personal devices [8] and the emerging recognition of the value of people-centric sensing in urban environments [45] [9]. The demands of urban sensing are distinct from the set of assumptions that underpin existing sensor network research. As a consequence the infrastructure to support applications of this type will bear little resemblance to the conception of a sensor network discussed in the literature to date. Recently, at the Urban Sensing Summit [46], a cross-disciplinary group of researchers and technologists have been discussing a broad agenda for citizen-initiated sensing in support of future civic, cultural, and community life in cities. In this paper, we proposed a wireless sensor edge for Internet based on a people-centric opportunistic sensor networking approach in support of urban sensing applications.

Acknowledgment

The authors would like to thank David Kotz and Sean Smith for their input on this paper. Shane Eisenman is currently supported by the Army Research Office (ARO) under Award W911NF-04-1-0311, and Nicholas Lane, Emiliano Miluzzo, and Ronald Peterson by grant number 2005-DD-BX-1091 awarded by the Bureau of Justice Assistance through the Institute for Security Technology Studies, Dartmouth College.

8. REFERENCES

[1] Ubisense. http://www.ubisense.net.

[2] A. Arora and et al. Exscal: Elements of an extreme scale wireless sensor network. In *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2005.

[3] R. Govindan and et al. Tenet: An architecture for tiered embedded networks. In *CENS Technical Report No. 53, Nov. 2005.*

[4] C.-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft. Siphon: overload traffic management using multi-radio virtual sinks in sensor networks. In *SenSys '05: Proc. of the 3rd int. conference on Embedded networked sensor systems*, pages 116–129, San Diego, California, USA, 2005.

[5] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. Pump slowly, fetch quickly (psfq): a reliable transport protocol for sensor networks. In *IEEE Journal on Selected Areas in Communications*, volume 23, pages 862–872, April 2005.

[6] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *SenSys '04: Proc.of the 2nd int. conf. on Embedded networked sensor systems*, pages 134–147, Baltimore, MD, USA, 2004.

[7] Moteiv tmote invent: http://www.moteiv.com/products-tmoteinvent.php.

[8] Sensorplanet. http://www.sensorplanet.org/.

[9] D. Malan, T. Fulford-Jones, M. Wesh, and S. Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *Proc. MobiSys 2004 Workshop on Applications of Mobile Embedded Systems*, 2004.

[10] L. Krishnamurthy and et al. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *SenSys '05: Proc. of the 3rd Intl. Conf. on Embedded Networked Sensor Systems*, pages 64–75, San Diego, California, USA, 2005.

[11] Victoria Transport Policy Institute. London congestion pricing. www.vtpi.org.

[12] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *SenSys '04: Proc. of the 2nd int. conf. on Embedded networked sensor systems*, pages 13–24, Baltimore, MD, USA, 2004.

[13] Tune your run. http://www.apple.com/ipod/nike/.

[14] M. Feldmeier and J. Paradiso. Giveaway wireless sensors for large-group interaction. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1291–1292, Vienna, Austria, 2004.

[15] G. Bell and J. Gray. Digital immortality. In *Communications of the ACM*, volume 44(3), pages 28–31, 2001.

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

[17] Ubifit project. http://dub.washington.edu/projects/ubifit/.

[18] Virone and et al. An assisted living oriented information system based on a residential wireless sensor network. In *Proc. of 1st Conf. on Distributed Diagnosis and Home Healthcare*, pages 95–100, April 2–4 2006.

[19] T. Ishida. Digital city kyoto. *Commun. ACM*, 45(7):76–81, 2002.

[20] Silent london - a map of the capital's quietest places. http://www.simonelvins.com/silent_london.html.

[21] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A unifying link abstraction for wireless sensor networks. In *SenSys '05: Proc. of the 3rd int. conf. on Embedded networked sensor systems*, pages 76–89, San Diego, California, USA, 2005.

[22] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks. In *Int. Conf. on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA*, Oct. 2002.

[23] P. Levis, D. Gay, and D. Culler. Active sensor networks. In *Proc. of the 2nd USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI), May 2005.*

[24] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 107–118, Atlanta, Georgia, USA, 2002.

[25] Liu and et al. Mobility improves coverage of sensor networks. In *Proc. of the 6th ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing*, pages 300–308, May 25–27 2005.

[26] Openwrt. http://openwrt.org/.

[27] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proc. of INFOCOM 2006*, April 23-29, Barcelona, Spain.

[28] E. Shi and A. Perrig. Designing secure sensor networks. *Wireless Communication Magazine*, 11(6):38–43, December 2004.

[29] C. Karlof, N. Sastry, and D. Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proc. of the 2nd int. conf. on Embedded networked sensor systems*, pages 162–175, Baltimore, MD, USA, 2004.

[30] T. Stathopoulos, L. Girod, J. Heidemann, and D. Estrin. Mote herding for tiered wireless sensor networks. In *CENS Technical Report 58*, December 7, 2005.

[31] M. Ho and K. Fall. Poster: Delay tolerant networking for sensor networks. In *Proc. of IEEE Conference on Sensor and Ad Hoc Communications and Networks, 2004.*

[32] 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.

[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] S. Nath, Y. Ke, P. B. Gibbons, B. Karp, and S. Seshan. Irisnet: An architecture for enabling sensor-enriched internet service. In *Technical Report, Intel Research*, number IRP-TR-03-04, Pittsburgh, June 2003.

[35] P. Pietzuch, J. Shneidman, J. Ledlie, M. Welsh, M. Seltzer, and M. Roussopoulos. Hourglass: An infrastructure for connecting sensor networks and applications. In *"Harvard University Technical Report TR-21-04"*, 2004.

[36] J. Liu and F. Zhao. Towards semantic services for sensor-rich information systems. In *The 2nd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (Basenets 2005), Boston, MA, Oct. 3, 2005.*, pages 44–51.

[37] M. Romn, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, pages 74–83, Oct–Dec 2002.

[38] J. Sousa and D. Garlan. Aura: An architectural framework for user mobility in ubiquitous computing environments. In *Proceedings of 3rd IEEE/IFIP Conference on Software Architecture, Montreal, 2002.*

[39] Endeavour. http://endeavour.cs.berkeley.edu/.

[40] Oxygen. http://www.oxygen.lcs.mit.edu/.

[41] Sentient computing. http://www.uk.research.att.com/spirit/.

[42] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

[43] The cricket location-support system. http://cricket.csail.mit.edu/.

[44] Quentin Jones and Sukeshini A. Grandhi. P3 systems: Putting the place back into social networks. *IEEE Internet Computing*, 9(5):38–46, 2005.

[45] G. Simon et al. Sensor network-based countersniper system. In *Proc. of the 2nd int. conf. on Embedded networked sensor systems*, pages 1–12, Baltimore, MD, USA, 2004.

[46] Urban Sensing Summit, CENS, UCLA, May 2006. http://bigriver.remap.ucla.edu/remap/index.php/Urban_Sensing_Summit.