Collaborative Research: CRI: Large-Scale Open Sensor Network Testbed for Urban Monitoring

1 Overview

Wireless sensor networks are attracting increased interest for applications in large-scale monitoring of the environment [46], civil structures [5, 47], roadways [1], and animal habitats [6, 24, 42]. Sensor networks also have applications in homeland security, including biochemical hazard detection and tracking [31]. In K-12 education, networked weather sensors stimulate higher levels of achievement in teachers and students [43]. This technology has the potential to revolutionize the harnessing of data from the physical world. One key factor, however, has greatly limited the development of this field - the daunting logistical challenge of experimenting with thousands of small, battery-powered nodes.

We propose to eliminate this problem by building *CitySense*, a large-scale, urban sensor testbed of 100+ nodes powered by city electricity (from streetlights), and then operating it as an open research facility for academic and industrial groups who want to experiment with novel distributed sensing architectures. We have reached a preliminary agreement with the City of Cambridge, Massachusetts on terms for using their streetlights. (See attached letter of support.) Figure 1 shows a conceptual laydown of nodes in Cambridge. Each node is linked to its neighboring nodes via a high-speed radio. We will employ ad hoc routing protocols to create a mesh network enabling all nodes to directly communicate with each other at the IP (network) layer.

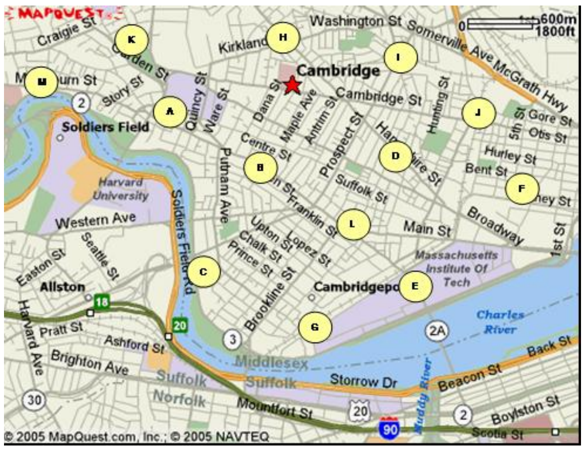


Figure 1: **Conceptual Deployment of Sensor Nodes in Cambridge, MA. Actual distance between nodes will be smaller.**

Unlike the many existing low-power, low fidelity mote sensor networks, CitySense provides high-throughput nodes with long-range 802.11 radios and high-fidelity, low maintenance sensors, powered by the electricity for city streetlights. As such CitySense will provide unprecedented access to data from the physical world, enabling research that is not possible via simulation or small, lab-based testbeds.

Research activities: CitySense is designed to support a broad range of research projects in large-scale sensor networks. This testbed will be open to researchers, allowing them to collect data, experiment with networking protocols, and reprogram nodes over a web-based interface.

We plan to use CitySense as an essential component of projects involving high-level programming models for sensor networks; effective techniques for resource allocation and sharing; and integration of sensor networks with Internet-based information systems. We will explore these projects in the context of a concrete application, that of monitoring air pollution transport in a dense urban environment. In addition to the projects that we will undertake ourselves, we intend to open up the testbed to other research groups to enable many as yet unforeseen projects.

Educational activities: Development of a large-scale sensor network testbed will also benefit educational activities at the graduate, undergraduate, and K-12 levels. We intend to use CitySense as a tool to teach distributed systems concepts at the graduate and undergraduate levels. Together with our unfunded collaborators from the Tufts Center for Engineering Educational Outreach, we plan to develop a curriculum for K-12 in sensor networks that makes complex engineering concepts such as sampling and averages concrete. (See attached letter of support.)

Why this team? Harvard and BBN Technologies bring world-class expertise in distributed sensor nets – academic and industrial – and we are in close physical proximity. We will collaborate very closely in this project, and plan for many visits between our organizations. Ever since BBN developed the original ARPA-net routers, BBN has been at the forefront of Internet research and development. As one example, BBN's mobile ad hoc networking technology, the Near Term Digital Radio (NTDR), is the only such operational system in active usage by the armed forces.

2 Infrastructure: The CitySense Testbed

Realistic sensor network applications are now demanding much larger networks, distributed over wider areas, with higher data rates and more sophisticated processing requirements. As an example, large-scale monitoring of pollutants in an urban environment is one area where sensor networks could be brought to bear. By distributing a large network of high-quality sensors over an entire urban area, monitoring airborne pollutants as well as related metrics such as wind velocity, humidity, temperature, rainfall, and automobile traffic, it will be possible to develop detailed, high-resolution models of the impact of pollution down to the specific street and neighborhood level. Existing approaches to pollution monitoring make use of very few, widely distributed sensors or handheld monitors used to manually collect data along streets and sidewalks [21].

In such urban applications, individual sensor nodes can be placed atop streetlights and powered continuously, avoiding the requirement of low-power operation. In addition, the distance between any two sensors must be generally greater than the (ideal) 100m obtained by low-power WPAN (802.15.4) radios, necessitating higher-power solutions. Finally, the application's data requirements involve multiple channels, high data rates, and complex processing at the node level. In the spectrum of sensor network designs, we see a compelling need for higher-power, more capable sensor nodes than those used in low-power, battery-operated deployments.

We will deploy 100+ streetlight mounted nodes distributed over an urban area. Each node is linked to its neighboring nodes via a high-speed radio. Ad hoc routing protocols running on each node will create and maintain a mesh network, enabling all nodes to directly communicate with each other at the IP (network) layer. Our deployment plan calls for building and fully testing a small indoor testbed in the first year of the proposed effort. In the second year, we will contract the city of Cambridge to deploy 10-20 nodes along Concord Avenue, which connects the Harvard campus to BBN. Each subsequent year we will deploy an additional 25-30 nodes, until 100 nodes have been deployed throughout the city.

The CitySense testbed will include the sensor nodes, wireline gateway nodes linking the wireless mesh to a data repository and a web-server for external Internet access. These are described in the following two subsections. Afterwards we discuss important considerations of an urban deployment, followed by a description of the sensor node software environment and the sensing equipment.

2.1 Sensor nodes and radio base system

The CitySense testbed design and node hardware is powerful and flexible enough to support both ends of the sensing application spectrum, from long term sensor data collection studies to real-time sensor data monitoring. Each sensor node will consist of a reprogrammable, reconfigurable base platform and CPU, multiple environmental sensors (including temperature, humidity, windspeed and direction), and a high-bandwidth 802.11 radio outfitted with a high-gain omnidirectional antenna.

A unique feature of the CitySense design is the the availability of power from the streetlight mounting. While past sensor networks have by necessity mandated very lightweight processing and communications power [35], we are not constrained by battery longevity. This freedom enables us to opt for the powerful Soekris net4526 CPU. Soekris has been used in prior mesh network deployments such as MIT's RoofNet.[1] Matched with a Linux based OS, the Soekris CPU enables rapid development using standard software

[1] `http://pdos.csail.mit.edu/roofnet/doku.php?id=hardware`

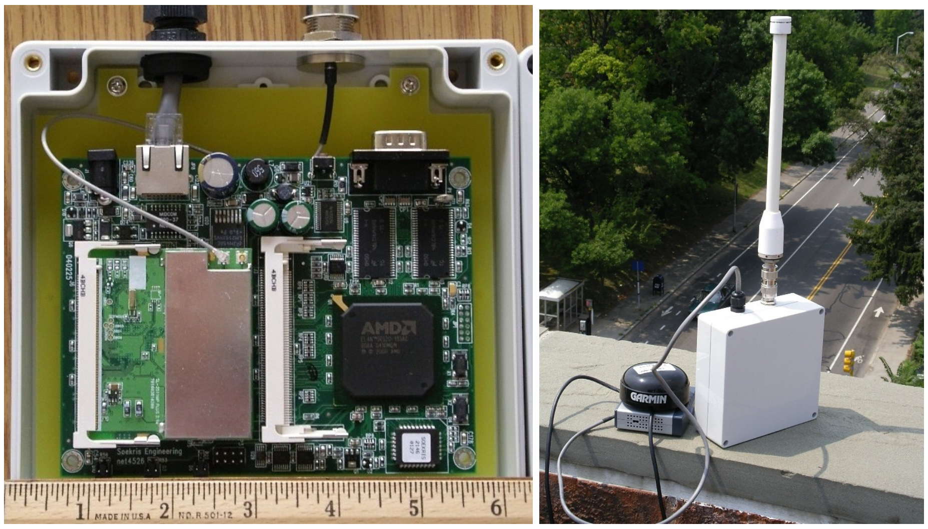


Figure 2: **Sensor Node Hardware. Left: The Metrix Mark I with 802.11b Radio, Ethernet (power), Serial and miniPCI Ports. Right: Unit in Weatherproof Mounting and a GPS Sensor Attached via a Ethernet Hub.**

tools. Coupled with a high transmit power 802.11b miniPCI radio card, our sensor nodes combine ample local processing power with high-bandwidth wireless connectivity.

Our base node consists of the Metrix Mark I shown in Figure 2. This unit contains a Soekris net4526 motherboard with an Intel x86 compatible 133 MHz AMD Elan processor packaged in a weatherproof enclosure.[2] We selected the NL-2511MP 802.11b mini-PCI adaptor with 200 mW of transmit power for long range wireless connectivity. We chose the radio because of its extended range capabilities. Although 802.11g offers a higher peak bandwidth (54 Mbps) than 802.11b, studies by MIT have shown that 802.11g radios rarely outperform 802.11b radios unless they are close together [25]. Power is supplied to the unit via Power over Ethernet (PoE). A custom power supply will be designed to convert the line voltage into the power needs for the sensor node. We provide power to those wishing to add custom sensors to the CitySense nodes via PoE. We will have a five port Ethernet hub that will serve up to 4 extra PoE sensors. External sensor devices can be connected to the node via either the serial port, the wired Ethernet port or the extra mini PCI slot available on the Soekris motherboard.

2.2 Wireline gateway nodes and servers

In addition to the sensor nodes, a few desktop and/or laptop computers equipped with 802.11 radios situated at both Harvard and BBN will serve as gateways to the sensor radio network. These gateways will connect to a wired Ethernet linked to machines hosting the sensor data repository and CitySense resource management systems including a web-server to support remote access to the testbed.

2.3 Urban Deployment Considerations

The outdoor urban environment presents several challenges to the design of a wireless network of sensors. These include: physical environment factors, network coverage, and network security.

2.3.1 Physical environment

Cold temperatures, thieves and malfunctioning hardware are realities that face computing equipment deployed in outdoor urban environments. We plan to install CitySense nodes atop of streetlight fixtures, as shown in Figure 3. Each streetlight has a standardized, screw-in connector for the photo-sensitive switch that turns on the light when it is dark. A standardized screw-in tap can be attached to draw off electrical power (AC) for the CitySense node.

[2]`http://metrix.net/metrix/products/packages/MET-KIT-MARK-I.html`

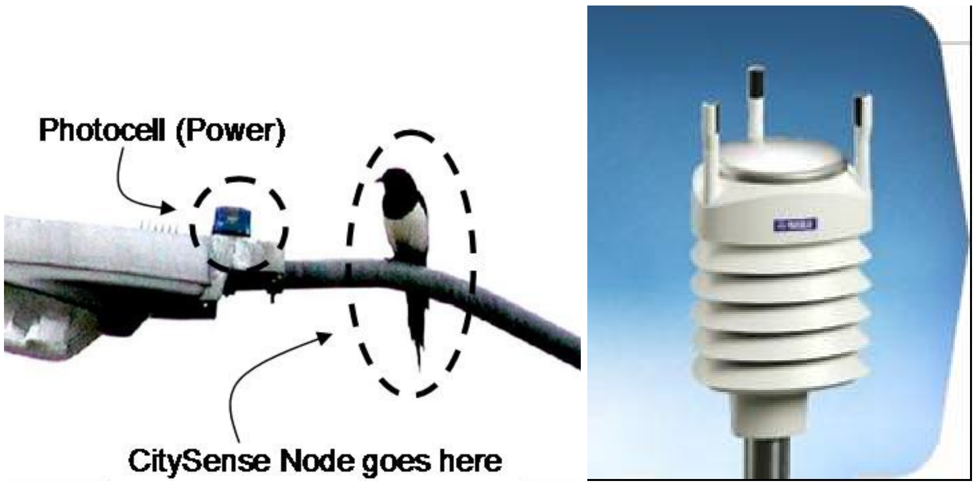


Figure 3: **Left: Sensor Node Positioning and Power Source. Right: Vaisala WXT510 Meteorological Sensor.**

We have come to an agreement with the City of Cambridge, which owns the streetlights, for installing and maintaining CitySense nodes (see attached letter of support). Obviously each node installation will require skilled installers, and we expect that the installations will be carried out by city workers, as will any replacements of failed nodes. Of course we must also pay for the electricity used by all nodes (which strongly motivates a ”sleep” mode for inactive periods).

For streetlight deployment, a weatherproof housing is needed for both CPU and sensors. Such a housing comes with the Metrix Mark I CPU (see Figure 2). The environmental operating temperature range of the CPU (0 .. +60 °C) may require a small heater to be placed in the weatherproof mounting for operation through winter. Our baseline meteorological sensor, the Vaisala WXT510 (see Figure 3), ensures low-maintenance by having no moving parts and by providing a heater that allows operation in an air temperatures ranging from -52 to +60 °C.

To prevent the theft of hardware, ground mounted sensors will be secured in a cagelike enclosure. The CPU, radio and baseline sensors are mounted 10 meters above the street and are therefore protected.

2.3.2 Network Coverage

CitySense’s streetlight mounted nodes benefit network coverage in two ways. First, we can leverage the natural line-of-sight paths provided by roadways to achieve large inter-node spacing despite the presence of many RF obstacles. Second, the uniform mounting height, roughly 10 meters above the street level, helps RF reception range by reducing the strength of the ground reflected signal. An additional benefit of the uniform mounting height and antenna orientation is the reduced probability of interference from non-CitySense RF emitters such as WiFi radios in private laptops and access points.

To ensure that mesh connectivity is maintained in the face of unplanned node outages, our inter-node spacing will be less than half the range for the radios. This ensures RF overlapping such that two nodes will not loose connectivity should a single intermediate node fail (see Figure 4). Our initial spacing will be based upon estimates, however, we will experimentally validate our estimates prior to deployment.

To estimate our inter-node spacing, we follow a two step approach, similar to [15]. First, we use a free-space, r^2, model to calculate the received signal strength of our radio at its maximum free space range. Then using this receiver sensitivity, we apply a model for urban RF propagation to derive the expected range for our deployment environment where multi-path fading is significant.

The Friis free space propagation model, defines the received power, P_r, at d meters from the transmitter as,

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{16\pi^2 d^2 L}, \tag{1}$$

where P_t is the power of the transmitter in Watts, G_t is the gain of the transmitter, G_r the gain of the

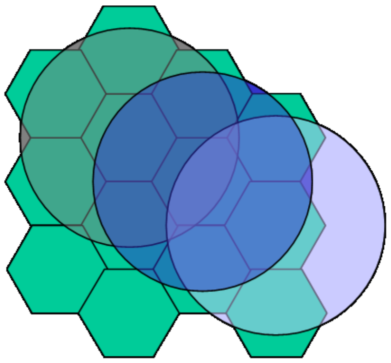


Figure 4: **Planned Inter-Node Spacing and RF Coverage Allow for Single Node Failures.**

receiver, L is the system loss factor and $\lambda = \frac{c}{f} = 0.125m$ for 802.11.

By applying Equation 1 to our radio manufacturer's suggested maximum free-space range at 11 Mbps of 300 meters with a unity gain omnidirectional antenna, we obtain a received RF power of 220 pW or -67 dBm. Applying Equation 1 again using the 8.5 dBi gain antennas in both send and receive mode yields a theoretical free-space range of 2.12 Km.

Next, we apply the Hata empirical model [13] which predicts RF signal path-loss given various parameters of the urban environment and radios. Using a web-based tool,[3] we determine a theoretical urban range of 175-600 meters. We expect the actual range to fall between our two estimates, with the average being 388 meters. Thus, on average, an inter-node spacing of 194 meters achieves our goal of overlapping coverage. Given, the streetlight spacing of 30 meters, a node will be placed on every 6th streetlight along each roadway. However, exact node placement will depend upon the specific environmental constraints of the roadways as well as the requirements of specific research activities, such as pollution monitoring.

Note: We believe our range estimates to be conservative since the radios afford even greater range at lower throughput rates, e.g., 1 and 5 Mbps, than that used to make our estimates, 11 Mbps.

2.3.3 Network Security

Security and interference are critical concerns in urban environments where many private 802.11 networks may exist. CitySense will use a two-layer approach to security. At the link layer, we employ 128 bit Wire Equivalent Protection (WEP) encryption to ensure that passive listeners cannot snoop on the CitySense network. At the transport layer, applications will use a secure transport layer protocol, e.g., SSL or SSH, to perform all inter-node communications. Due to the potential for snoopers to crack the WEP keys over time, we will periodically distributed new keys to the nodes using a secure transport protocol. We will mitigate RF interference via the uniform mounting height of the nodes and the high-gain antennas that will be relatively insensitive to emitters out of the plane of the nodes.

2.4 Software environment

The Metrix CPU hosts the ad hoc wireless networking protocols that form the backbone of the testbed. BBN Technologies has extensive experience deploying and maintaining multi-hop wireless ad hoc networks [36].

We expect to run the Optimized Link State Routing protocol (OLSR), as it has shown, via simulation, to scale well in networks with more than 100 nodes [17], and thus appears well-suited to our application. This open-source solution uses multipoint relays to efficiently flood routing information across the network.[4] Its design allows TCP/IP based applications to run unchanged across a multi-hop ad hoc network.

These baseline protocols can be replaced with user defined configurations so as to support the goal of providing an open framework in which to develop and test new wireless networking protocols in a realistic testbed. Updates to the networking software will be made via the web-based management interface. To

[3]`http://www.antd.nist.gov/wctg/manet/prd_propcalc.html`

[4]`http://hipercom.inria.fr/olsr/`

prevent manual intervention (literally climbing the pole) in the case of network software failure, the nodes operate in a failover mode. Each node monitors its connectivity to the web management system and to its neighbor nodes. A node reboots into the baseline configuration when connectivity to the network is lost for more than a predefined period of time.

2.5 Sensor Hardware

We will select sensor hardware based on the following criteria: form-factor; low-maintenance; cost; ability to access the sensor data in real-time from the Metrix CPU, and relevance to research activities of our partners. For the baseline testbed, we have selected the Vaisala WXT510 weather transmitter, shown in Figure 3, that measures wind-speed, direction, relative humidity, temperature, pressure and precipitation. It uses no moving parts to lower maintenance and has a heater to ensure year round operation. The WXT510 unit will connect to the Metrix CPU via its serial port.

Additional sensors, including seismic, radioactivity, carbon monoxide and PAH detectors may be added to the baseline depending on the needs of supported research projects. New sensors will connect via a Power over Ethernet (PoE) hub or via the extra miniPCI port. Sensors co-located with the CPU unit will also draw their power from the streetlight via PoE. Ground-mounted sensors may connect to the PoE hub on the streetlight with a cable.

Open Sensor Interface: An important goal of the testbed is to incorporate new sensor technologies as they emerge. As such we will define an open sensor interface that standardizes the methods of data capture and sensor configuration. This interface will also define an XML schema for exchanging sensor data. Adding a new sensor to the system entails writing an adaptor for the sensor that conforms to the standard interface. This ensures that the base software does not require changes when new sensors are added.

2.6 Advantages of Infrastructure over Current Systems

Never before has there been a large scale testbed explicitly designed for sensor network experimentation. It offers many advantages over the alternatives: simulation, small testbeds, and customized infrastructure. As with simulation, CitySense supports repeatability by allowing the experimenter to specify the set of nodes and the network topology. However, because of the real world deployment environment, experiments performed using CitySense will reflect the natural complexities that are frequently lost in a simulated model.

While small scale testbeds may be less expensive to initially setup, they are harder to maintain over time. Typically they rely on battery operated nodes and low-cost, uncalibrated sensors that must be frequently re-placed and do not allow for easy experimental design or repeatability. Achieving desired coverage can also be challenging when dealing with ground-based motes whose typical maximum range is 100m [46].

Remote programming and access in small mote testbeds is severely limited due to the low-bandwidth and low-processing power of each node. Each node performs simple pre-processing and relaying of data collected from the sensors. The richness of the data that can be communicated through the sensor network is severely limited by the low-power 250 Kbps radios (802.15.4). Whereas CitySense would enable high-speed access to each node, supporting remote reprogramming. It would also support data capture, signal processing and wireless transfer of data from high resolution sensors. Although, each CitySense node is less expensive, the expected 500m range of the high-power radios (802.11b) and the high-quality, high-reliability of the sensors means fewer nodes and lower maintenance costs over the life of the testbed.

CitySense opens the door to experiments and applications that were previously only available via simulation. For example, monitoring the spread of pollutants or bio-chemical agents across an urban area in real time could be supported by a future expansion of the testbed, *CitySense 2.0*. Figure 5 shows a plume of pollutants spreading across the field of sensor nodes. Details of the specific research activities that will be enabled by the proposed infrastructure are described in the next section.

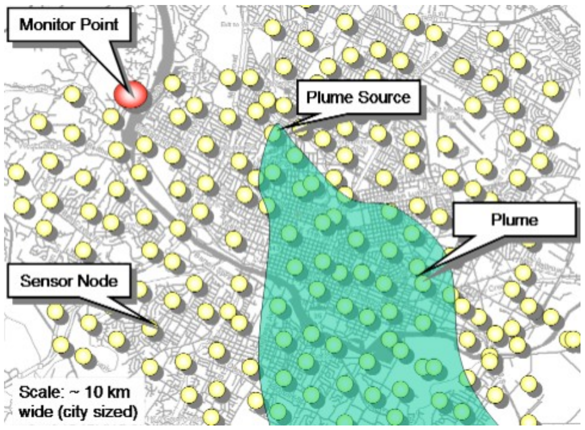


Figure 5: **Candidate Application: Tracking Spread of Pollutants in Urban Environments.**

3 Research Projects Enabled by the Infrastructure

CitySense is designed to support a broad range of research projects in large-scale sensor networks. We plan to use CitySense as an essential component of projects involving high-level programming models for sensor networks; effective techniques for resource allocation and sharing; and integration of sensor networks with Internet-based information systems. We will explore these projects in the context of a concrete application, that of monitoring air pollution transport in a dense urban environment. In addition to the projects that we will undertake ourselves, we intend to open up the testbed to other research groups to enable many as yet unforeseen projects.

3.1 High-level programming models for sensor networks

Developing distributed applications for sensor networks is very difficult. Existing programming models for sensor networks are very low-level, providing raw hardware access and radio messaging, forcing application designers to build up a great deal of machinery to perform sampling, routing, time synchronization, and data aggregation. The programming challenge is amplified by the event-driven nature of sensor application design and the unreliability of radio communication. The core difficulty with building sensor network applications is that high-level, global behavior must be expressed in terms of complex, local actions taken at each node.

We are developing a high-level programming language, called *Regiment*, that allows the sensor network's behavior to be described in terms of global operations on the entire network [28, 27]. Such a programming model allows application developers to interact with the sensor network at a high level, without having to concern themselves with node-level details of sensing, communication, and resource management. A *macroprogramming language compiler* generates efficient per-node code from the global macroprogram, automating the process of decomposing global programs into complex local behaviors. This language is supported by a *runtime system* that dynamically optimizes the placement of macroprogram components in the network.

We have received NSF support for the Regiment project (see Section 6) in the context of monitoring volcanic eruptions with mote-like sensors. However, Regiment is not tied to this platform, and through this proposal we wish to translate the system onto the CitySense testbed. In doing so we will have an opportunity to explore the use of Regiment on a very different platform and application domain.

Regiment language model

Regiment is a high-level language based on the ideas of *functional reactive programming* (FRP) [11, 30]. In Regiment, the principal objects that the programmer manipulates are *signals*, which represent continuous streams of time-varying values. For example, the magnetometer sensor of a given mote, which returns a floating-point value, is a `Signal(float)`. Regiment provides a number of operations for manipulating and building new signals out of existing signals. For example, one can write "`smap (`λ`x.x`

`> thresh) s`" to convert a sensor's floating-point signal `s` to a boolean signal that is *true* whenever the sensor value is above a threshold.

Of course, in a macroprogramming environment, programmers often want to manipulate collections of signals. Thus, another central concept is the *region*. We can think of a region as an indexed table of signal values. For example, the collection of all nodes' acoustic sensor values (over time) indexed by the associated node's position would be a `Signal(Region(pos,float))`. The collection of values that make up a region in a signal can change with time due to node failures, interference, or the arrival of new nodes in the network. Thus, programming at the region level insulates the application from the low-level details of failure in the network.

There are four key operations on regions: `rmap`, `rrestrict`, `rfilter`, and `rfold`. The `rmap` operation is the region-level version of `smap` and applies a function to each value in the collection. For instance, "`rmap (λx.x > thresh)`" allows one to convert a region of sensor readings to a region of booleans. The `rrestrict` operation can be used to filter out signals based on their index. For example, one can use `rrestrict` to cut a region down to those signals emanating from the frontier of the region. Because indices do not vary with time, `rrestrict` can be used to prune a region in a static fashion. In contrast, `rfilter` is a *dynamic* filter that is evaluated continuously to determine whether a given signal should remain in the region. As a simple example, one can use `rfilter` to remove sensors from the region whose acoustic sensor readings do not lie above some threshold:

```
thresh r = smap (rfilter (λv.v < threshold)) r
```

The `rfold` function is used to aggregate all of the values in a region to a single value using an associative and commutative combining function. For example:

```
srsum r = smap (rfold (+) 0) r
```

defines a function `srsum` that maps a signal of regions to a signal representing the sum of all values in the region. It is important to note that at any point in time, the nodes participating in a region could fail, and thus the sum will only include those values that were successfully communicated.

Regiment examples

Although a complete discussion of the Regiment language is beyond the scope of this proposal, we will give a sense of the language applied to our pollution monitoring application described in Section 3.4. The first Regiment program samples the data from each CitySense node equipped with a carbon monoxide (CO) sensor, for CO values that are above a given threshold:

```
main (world : Signal(Region(Node,Sensors))) : Signal(Region(Node,(float,float))) =
  let co_nodes = srrestrict has_cosensor world
      co_data = srmap get_co co_nodes
  in srfilter (λv . v > THRESH) co_data
```

The function `main` takes a signal of regions representing all nodes in the sensor network. The `Node` type is a record of static information about a node, such as whether the node has a particular sensor type. The `Sensors` type is a record that holds the time-varying sensor values for each node. The program first selects the CO sensor nodes using a combination of `smap` and `rrestrict`. It then uses `srmap` to sample the CO data from these nodes at each point in time. The data is then filtered according to the predicate $v >$ THRESH before being returned.

A more sophisticated example aggregates data from multiple carbon monoxide sensors and only delivers a report when enough sensors in the network detect a high-CO condition. To write this program, we introduce the notion of an `Event`, which is conceptually just a `Signal`(`bool`) that transitions from false to true at a single point in time and henceforth, always returns true. The operator

$$\texttt{edge} : \texttt{Signal}(\texttt{bool}) \rightarrow \texttt{Event}$$

converts a signal to an event that fires at the first transition from false to true, whereas the operator

$$\texttt{until} : \texttt{Event} \rightarrow \texttt{Signal}(U) \rightarrow \texttt{Signal}(U) \rightarrow \texttt{Signal}(U)$$

is used to switch from one signal to another based on the occurrence of an event.

We can now write the program that samples data from all CO sensors and tallies "votes" for each node that exceeds the threshold condition. When enough votes are counted, the data from *all* CO sensor nodes is returned:

```
main world =
  let co_nodes = srrestrict has_cosensor world
      co_data = srmap get_co co_nodes
      detected = srfilter (λv . v > THRESH) co_data
      votes = srfold (+) 0 (srmap (λx.1) detected)
      enough_votes = edge smap (λv . v > VOTE_THRESHOLD) votes
  in until enough_votes nullsig co_data
```

Here, the "vote" from each node (which can be contained in a small radio message) is counted using the `rfold` primitive over the `detected` region. The `until` keyword is used to trigger data collection when `enough_votes` fires: it returns the null signal until the event, and the CO sensor data thereafter.

More generally, one can develop event-driven loops that allow a signal's behavior to cycle from among a set of other signals. For instance, we could modify the above program so that the output signal becomes dormant again after a certain number of sensors fall below the detection threshold.

Role of the CitySense testbed

Our current work on Regiment [28, 27] has been focused on mote-based networks running in a dense area. The CitySense testbed will provide a vastly different development and evaluation environment for this project. Generalizing Regiment to this new domain will yield significant insights into appropriate language and runtime models for relatively high-powered, widely distributed sensor networks. In addition, we plan to work directly with the other users of the CitySense testbed (including the public health studies described below) to employ Regiment as the core programming interface to the sensor network. Not only will Regiment greatly simplify the ability for domain scientists and other users to program the CitySense network, but it will also provide us with significant real world experience for evaluating the language.

3.2 Resource allocation in shared sensor networks

We intend to allow multiple users and applications to share the CitySense testbed. Existing sensor networks are designed for running a single application at a time, time- and space-sharing has not been a serious concern until now. CitySense offers a very different environment and raises many interesting questions for resource management. We assume that users will want to schedule long-running applications that operate on fine time scales across a significant set of the testbed nodes. As a result, it is inappropriate to allow any individual user to "lock down" testbed nodes for significant periods of time. As a result we face a very interesting resource management problem: how to allow multiple applications to share CitySense nodes while preventing any application from consuming an excessive amount of CPU, memory, or network bandwidth.

The PlanetLab [34] testbed faces a similar problem in terms of resource sharing. PlanetLab consists of several hundred servers connected to the Internet, allowing users to conduct experiments with a real, large-scale distributed system. A wide range of proposals for effective resource allocation in PlanetLab have been developed. As currently deployed, PlanetLab allocates each user a "slice" of each node and prevents resource conflicts with a set of OS patches to provide proportional share scheduling and resource quotas. Still, this leaves open the problem of determining the appropriate resource allocation for each slice.

In order to seed interest in the CitySense testbed, we plan to initially impose no resource constraints on applications. That is, users will be given accounts on CitySense and expected to use the testbed in a cooperative manner. This approach was taken by the initial deployment of PlanetLab and was very effective at getting users involved with the system before any sharing policies were introduced. As the popularity of the testbed increases, we expect the resource management problem will become more severe and will require mechanisms for limiting resource use.

Resource management in CitySense requires research on both *mechanisms* and *policies*. Mechanisms are required that can monitor resource usage of individual applications and enforce resource constraints when needed. The policy dictates the amount of resources provided to each application under an appropriate sharing discipline.

Resource allocation mechanisms: Our first observation is that the radio channel is the most constrained resource in CitySense. We do not expect users to perform processing on individual CitySense nodes that consume a large amount of CPU or memory relative to the node capacity. Therefore, existing approaches to resource compartmentalization in server systems (e.g., proportional-share scheduling and memory quotas) can be used to limit the node-level resource consumption of individual processes.

Limiting radio use is somewhat more challenging. A simple approach is to rate-limit transmissions by individual applications on each node. In effect, this treats the radio channel as a fixed per-node resource that can be partitioned across applications. While such an approach is a natural fit for wired networks (where link capacity is fixed), in wireless networks we must take into account interference and contention that introduce "resource crosstalk" across CitySense nodes.

Variations in node density impact the effective channel capacity: nodes with many radio neighbors will experience greater contention than those with few neighbors. As a result, simply allocating, say, 1 Mbit/sec to each of 5 processes on each CitySense node does not ensure that each node will actually achieve those rates, nor that the channel will be shared fairly across nodes.

Resource allocation policies: The more difficult research question deals with what policies should be used for allocating resources across multiple applications. In general, we expect that the resource needs of CitySense applications will be *elastic*: rather than requiring a constant amount of resources during the length of a run, applications will expand and shrink their requirements based on external stimuli. For example, an application that collects air pollution data may only trigger high-rate sampling and aggregation when pollution levels rise above a certain threshold. Therefore policies that provide a fixed resource budget may be inappropriate for many applications.

This elasticity can also be exploited to yield much greater allocation efficiency using statistical multiplexing. Much work on sensor network programming has yielded algorithms that can adapt to limited resource availability [44, 23]. As a result, it is also acceptable to expect applications to respond well to resource limits at runtime. As long as the allocation mechanism is able to give applications feedback on varying resource availability, an application should be able to adapt to changing conditions, for example, by varying the sampling rate or degree of in-network aggregation performed.

One approach to managing shared resources in sensor network testbeds draws on microeconomic techniques [2, 29]. In such a system, users submit bids for resources through a combinatorial auction. A bid might consist of a specification of the number of nodes and CPU, memory, or radio bandwidth requirements during a given time window. This approach is designed to prevent users from gaming the allocation mechanism and reserving resources that applications do not actually require. Bellagio [2] is an auction-based system for allocating PlanetLab nodes, while Mirage [29] has been used in a mote-based testbed.

While auctions are an attractive model from a resource efficiency perspective, they raise a number of research challenges for large-scale use that we plan to explore through this project. First, auctions are often counterintuitive, especially for new users who are unsure how much to bid for resources. We intend to develop an easy-to-use interface for submitting bids that provides feedback on recent auction results and calculates default prices based on recent activity.

Second, the bidding language must provide enough flexibility for users to express elastic and time-variable resource requirements. One approach is to use a continuous bidding process where applications must periodically submit new bids if resource requirements change. Using a programmatic interface to the auction server we can allow a rich interaction between applications and the resource allocation mechanism at runtime.

3.3 Hourglass: A Data Collection Network Infrastructure

A core challenge that emerges in the sensor network domain is that of the *infrastructure* that connects many disparate sensor networks to the applications that desire data from them. To date, work in the sensor network community has focused on collecting and aggregating data from specific networks with an associated base station. The problem of delivering this data to external systems is typically left open. At the same time, work in continuous query (CQ) systems [16, 8, 9, 10, 26] peer-to-peer overlay networks [38, 41] and publish/subscribe systems [4, 3, 7, 12, 33, 39] offers a range of techniques for similar application domains. However, these approaches do not directly address the needs of capturing and querying data

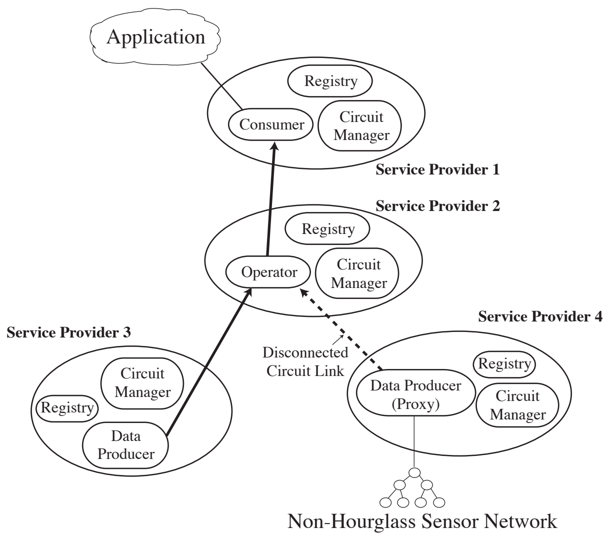


Figure 6: **The Hourglass Data Collection Network architecture.**

from a large number of sensor networks with different data schemas, access mechanisms, and resource considerations.

The Harvard Hourglass project focuses on the problem of developing an Internet-based infrastructure that handles aspects of naming, discovery, schema management, routing, and aggregating data from potentially many geographically diverse sensor networks. We call this infrastructure a *data collection network (DCN)*, which consists of an overlay network of Internet hosts that collaborate to provide a common interface for constructing pathways from sensor networks to applications. In this context, a sensor network might be a single node that collects data from a discrete physical location or a large collection of nodes capable of complex internal processing. In either case, the DCN supports data flow between applications and multiple sensor networks from different administrative domains with differing degrees of connectivity.

The essential data model in Hourglass is a *circuit*, which can be thought of as a set of network links that connect one or more sensor network sources to one or more recipients of that data. The circuit idea separates short-lived, small control messages from long-lived, stream-oriented data. Control messages are used to set up pure data channels that travel over multiple services. This is a useful idea because different types of data messages can be routed and processed using different tools.

Data flowing along a circuit may be filtered, aggregated, compressed, or temporarily buffered by a set of *services* that exist in the Hourglass infrastructure. These primitives allow applications to construct powerful ensembles of sensor networks and infrastructure services. Circuits are constructed with the aid of a *registry* that maintains information on resource availability and a *circuit manager* that assigns services and links to physical nodes in the network.

Given that sensor networks or applications receiving data from them may be mobile or connected through poor-quality wireless channels, support for *intermittent disconnection* is a critical requirement for a DCN. Hourglass is designed to handle intermittent disconnection gracefully by instantiating *buffer services* along links in a circuit that may cross a “connectivity domain,” and by incorporating a reliability model that permits either buffering or loss of data items being routed along a circuit.

Role of the CitySense testbed

CitySense will benefit the Hourglass project directly, by providing the back-end sensor network environment that will allow us to demonstrate query optimizations that span the data collection network and wireless sensor networks. CitySense can also be physically partitioned to emulate multiple physically disjoint sensor networks. In addition, Hourglass also ties together the other applications described earlier into a common infrastructure, and leverages the CitySense testbed through these efforts.

3.4 Application: Urban-scale air pollution monitoring

Apart from Computer Science research, we envision CitySense as a novel resource for environmental studies of the urban environment. The ability to instrument urban spaces at a high resolution can yield a tremendous advance in our ability to understand this challenging environment and its effect on the health of urban populations. To seed efforts in this area, we are working directly Harvard University's School of Public Health (Co-PI Majid Ezzati, Ph.D.) to leverage CitySense for a study of ambient and indoor air pollution.

In the year 2000, an estimated 800,000 deaths and 0.4% of the global burden of disease were attributable to urban ambient air pollution, as measured by ambient particle concentration (i.e. excluding lead and low-level ozone). Although urban ambient air pollution has been commonly defined at the level of a city in most epidemiological studies, recent research has illustrated the variation of exposure to this risk and the associated health effects in considerably smaller microenvironments [19, 20, 14]. This variability occurs because: (i) the ambient concentrations, composition, and dispersion of pollutants depend on the characteristics and location of pollution sources (e.g. mobile or stationary, combustion fuel and technology, etc.), meteorological factors, and urban physical characteristics, (ii) indoor concentrations (e.g. in buildings and vehicles) as a result of ambient pollution depend on the locations, type, and structure of indoor environments, and (iii) individuals and groups spend various amounts of time in different indoor and outdoor urban microenvironments [32].

For example, in a study of selected microenvironments in Boston, USA, it was found that PM_{10} (particulate matter less than 10 microns in diameter) concentrations varied more across selected indoor microenvironments where people regularly spend time (e.g. subways, buses, restaurants, etc.) than outdoor environments [19]. A recent study of increased mortality associated with ambient air pollution in the Netherlands highlights the importance of accurate characterization of spatial distribution of exposure. The study found that cardiopulmonary mortality was more affected by long-term exposure to air pollution due to living near a major road than from exposure to larger-scale urban and regional air pollution [14].

Air pollution exposure, accounting for spatial and temporal "exposure microenvironments," can be represented by the following relationship:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{m} w_j t_{ij} c_i$$

where c_i is the pollution concentration in the ith period of the day (n is the total number of time-periods for each individual); j is the spatial microenvironment defined based on the distance from pollution source (m is the total number of microenvironments); t_{ij} is the time spent in the jth microenvironment in the ith period, and w_j the conversion (or dilution) factor for the jth microenvironment which converts the concentration at the measurement point to concentration at the jth microenvironment. Therefore the two summations together represent all the time-location pairs for each individual. w_j often comes from air pollution models. Alternatively, if pollution concentrations are measured in a large number of microenvironments, the product of the terms c_i and w_j can be from actual measurement. At small scales, the measurement approach is preferable because the details of pollution source (e.g. mobile versus stationary sources), urban topology (e.g. height and distributions of buildings), and meteorological factors (wind speed and direction) affect the dispersion of pollution, possibly differently for various pollutants (e.g. coarse particles versus fine particles versus ozone). With actual measurements, the product $w_j c_i$ can be replaced with the measurement term c_{ij}. An additional advantage of a small number detailed studies with good measurements of pollution in multiple microenvironments is that the data can be used to improve the performance of air pollution dispersion models for subsequent analyses.

Using the CitySense testbed, we will undertake a detailed study of pollution levels in Cambridge, Massachusetts and be able to derive very detailed models of pollution transport. CitySense is the first project to provide a large-scale, high-resolution monitoring system for an urban environment. Such a detailed study would not be possible without this infrastructure — this is truly a unique opportunity to leverage wireless sensor technology to enable a new kind of scientific study. By applying CitySense to a concrete application involving pollution monitoring, we will strive to ensure the system is easy to use and that the acquired data is meaningful to the scientific community. Furthermore, we expect that our experiences with CitySense will inspire many more studies of this type.

3.5 Supporting external research efforts

Our ultimate goal with CitySense is to build an infrastructure to support a broad range of research projects by groups other than our own. Building and supporting CitySense as a shared testbed we will enable many novel research activities that can leverage the existence of a permanent, urban-scale sensor network. In this effort we draw on the experience of the PlanetLab [34] community which has in many ways revolutionized distributed systems research, allowing essentially any research group in the world to make use of the infrastructure. We will similarly open up CitySense and encourage external research groups, both academic and industrial, to undertake studies using this environment.

CitySense is designed to be extensible in many directions. The initial node platform and sensor suite requested in this proposal are by no means set in stone and we expect that new devices and sensors will be introduced over time. The use of a Web-based interface for scheduling, programming, and monitoring CitySense will in many ways mask users from this platform evolution. Using Linux as the host operating system, which is relatively stable and well-supported by the research community, will facilitate application portability.

Augmentation of CitySense by external groups: We will also encourage external research groups to augment CitySense with additional hardware capabilities. For example, a group that wishes to deploy a specialized sensor package on certain CitySense nodes will be allowed to do so provided that (a) the new sensor does not interfere with existing hardware and (b) the entire CitySense research community is given access to the sensor. In this way external groups can leverage the existence of CitySense nodes as a base platform for sensor deployment and

Public software and hardware releases: All of the software developed under this project will be released with an open source license, and the hardware designs will be published with no restrictions on use. This will permit external groups to replicate the CitySense environment for themselves, seeding an active community effort to continue developing the platform. We will endeavor to collaborate closely with external groups deploying CitySense networks. Tying all of these systems into a single, coherent Web-based interface will allow external users to program all of the CitySense deployments using a single set of credentials.

Expected external research projects: Issues that were previously only open to simulation studies can be experimentally validated using CitySense. Research areas include, wireless ad hoc networking, and distributed systems. Specifically, issues such as routing in RF noisy urban environments [22] and the effects of power control on network throughput [37] can be studied in a real-world environment. We have received support from Prof. Little from Boston University. He intends to use CitySense to further his research in multimedia access and retrieval across wireless networks. (See attached letter of support.)

4 Educational Activities and Broader Impact

Development of a large-scale sensor network testbed will also benefit educational activities at the graduate and undergraduate level as well as the K-12 level. We intend to integrate CitySense into assignments for graduate courses on sensor networks, operating systems, and hardware architecture. The undergraduate Operating Systems curriculum will be extended to cover distributed systems and will use CitySense as an environment for experimentation. In addition, we have a strong track record of integrating undergraduates into the research activities that CitySense will support. Finally, for the K-12 curriculum development we intend to work with the Center for Engineering Education Outreach (CEEO) at Tufts University who has extensive experience working with classrooms. (See attached letter of support.)

4.1 Graduate teaching and research

We plan to use CitySense as part of programming assignments and research projects in three graduate courses: Wireless Sensor Networks (Prof. Welsh), Operating Systems (Prof. Seltzer), and Computer Architecture (Prof. Brooks). Each of these courses involves paper readings as well as an independent research project. The programming assignments in the Wireless Sensor Networks course will serve as a tutorial to get students up to speed on using the testbed. Our small-scale MoteLab testbed already been used in a graduate course on Modern Distributed Systems, where it was used by several student research projects, several of which have been submitted for publication.

4.2 Undergraduate teaching and research

CitySense provides a valuable environment for teaching distributed systems concepts to undergraduates. Rather than dealing with simulations, CitySense allows students to easily program a large sensor network, providing "hands on" experience of a distributed system in a realistic setting. Students can explore concepts such as message routing, time synchronization, consensus, and distributed sensor data processing. Although sensor networks are not a traditional medium for teaching distributed systems, we believe that taking this approach will foster a great deal of creativity among students, leading to a range of interesting programming assignments. We plan to extend the current undergraduate Operating Systems curriculum to involve more distributed systems and networking concepts, and expect CitySense to play a significant role.

Harvard has a strong undergraduate research program and many undergraduates get involved in research projects through a senior thesis, summer work, or independent study. Each of the Co-PIs on this proposal has advised multiple senior theses and summer undergraduate research assistants on projects directly related to the sensor network research in this proposal. One of our undergraduates has developed functionality to space-partition the MoteLab testbed across different experiments. We intend to continue integrating undergraduates into the research program supported by CitySense.

4.3 K-12 Educational Outreach

We believe that CitySense can greatly benefit K-12 students as well by providing a hands-on engineering tool to explore the physical world. Together with Chris Rogers and Marina Bers of the Tufts CEEO, we will build a K-12 curriculum in sensor networks. The CEEO has been active in engineering educational outreach for over 15 years. Its mission is to attract and enable young people to pursue further studies in engineering and science, with an emphasis on attracting women. All of the CEEO work is focused on bringing engineering into preK-12 learning environments by providing professional development opportunities for teachers, curricular and other resource material for the classroom, and dedicated engineering programs for girls.[5]

CitySense would provide an extension to ongoing outreach work at the CEEO on robotics and sensors as a way to teach engineering. By leveraging its mesh of 100 meteorological sensors, concepts such as sampling, average values, ranges and trends can be concretely demonstrated. The wireless mesh also affords a unique instructional tool for exploring more advanced engineering concepts such as radio frequency propagation.

Through its web-based interface to the sensors, CitySense would allow access from multiple remote school sites. We envision generating excitement with both teachers and students through active interaction with real-world sensor data that will lead to a large community of users, similar to that of the WeatherBug Software [6]

4.4 Research community support

We envision CitySense not just as a singular testbed to support research at Harvard, but as a resource for the entire sensor network research community. Our ability to dynamically allocate sensor nodes to individual experiments, as well as provide remote (Web-based) programming, debugging, and instrumentation, makes it possible to open up the CitySense testbed to users at other academic institutions and industrial labs. Our MoteLab testbed has external users at UCLA, MIT, UC Berkeley, Boston University, and Ohio State.

In addition, all of the source code for CitySense will be released under an Open Source license, and will be made available on the Internet[7]. The software will also be available for CVS checkin access, allowing developers at other institutions to contribute new features or bugfixes back to the code. We are encouraging

[5]See `http://www.ceeo.tufts.edu/index.asp`.

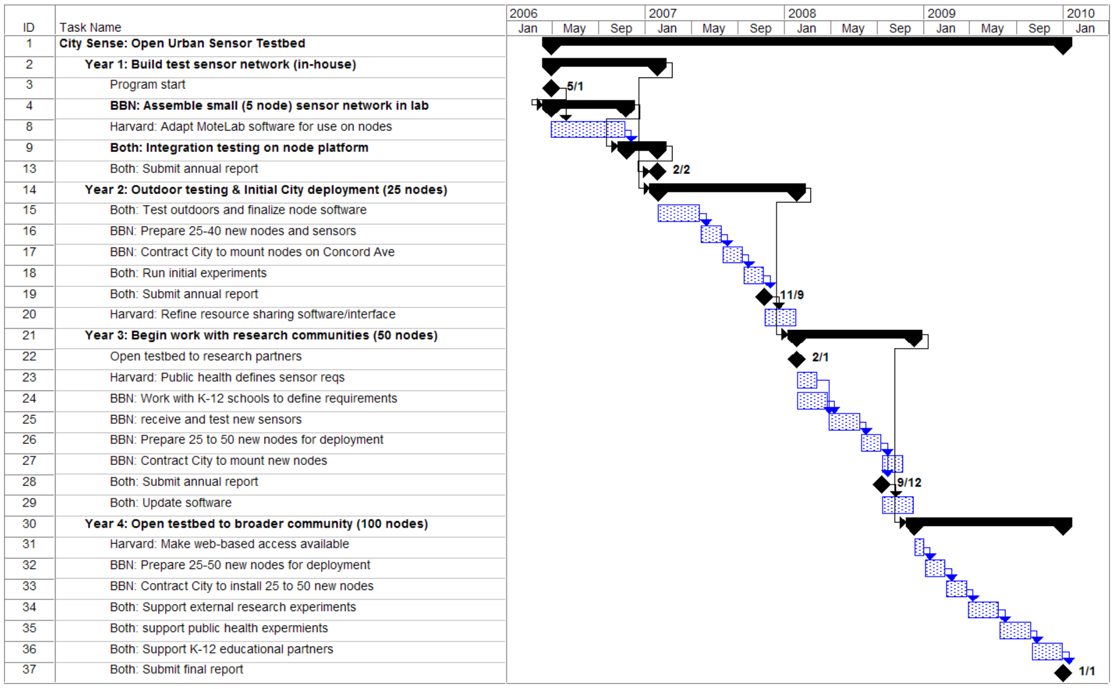
[6]See `http://www.weatherbugachieve.com/assets/pdf/White_Paper_1104.pdf`.

[7]See `http://sourceforge.net/projects/syrah`.

users at other institutions to adopt the MoteLab software for managing their own sensor network testbeds. So far, UC Berkeley, UCLA, and MIT have deployed variants of MoteLab to support their own research efforts.

Our longer-term vision involves providing a common Web-based front-end to many sensor network deployments at various universities and industrial research labs. This effort would link together multiple sensor network testbeds and allow users to experiment with distributed algorithms in a range of physical settings. We have already started discussions with Rutgers [8], UCLA, MIT, and UC Berkeley about this possibility. Before this can be accomplished, however, more experience is needed with deploying and managing a large-scale testbed in terms of addressing the myriad challenges of scaling, resource sharing, debugging, and data storage.

5 Deployment and Management Plan



ID	Task Name
1	**City Sense: Open Urban Sensor Testbed**
2	**Year 1: Build test sensor network (in-house)**
3	Program start
4	**BBN: Assemble small (5 node) sensor network in lab**
8	Harvard: Adapt MoteLab software for use on nodes
9	**Both: Integration testing on node platform**
13	Both: Submit annual report
14	**Year 2: Outdoor testing & Initial City deployment (25 nodes)**
15	Both: Test outdoors and finalize node software
16	BBN: Prepare 25-40 new nodes and sensors
17	BBN: Contract City to mount nodes on Concord Ave
18	Both: Run initial experiments
19	Both: Submit annual report
20	Harvard: Refine resource sharing software/interface
21	**Year 3: Begin work with research communities (50 nodes)**
22	Open testbed to research partners
23	Harvard: Public health defines sensor reqs
24	BBN: Work with K-12 schools to define requirements
25	BBN: receive and test new sensors
26	BBN: Prepare 25 to 50 new nodes for deployment
27	BBN: Contract City to mount new nodes
28	Both: Submit annual report
29	Both: Update software
30	**Year 4: Open testbed to broader community (100 nodes)**
31	Harvard: Make web-based access available
32	BBN: Prepare 25-50 new nodes for deployment
33	BBN: Contract City to install 25 to 50 new nodes
34	Both: Support external research experiments
35	Both: support public health expermients
36	Both: Support K-12 educational partners
37	Both: Submit final report

Figure 7: **CitySense Project Schedule.**

In our deployment of CitySense, we exploit the fact that ad hoc sensor networks can grow little by little. Our plans (see Figure 7) call for developing a small scale in-house sensor network in year 1. BBN will assemble the sensor node hardware and install the baseline ad hoc networking software while Harvard will adapt their MoteLab resource sharing software to the new nodes. We will integrate in both laboratories and begin testing of sensor data collection across the wireless network. In year 2 we will test the sensor nodes in outdoor environments to vett the packaging and ensure reliability prior to city streetlight mounting. By the end of year two we plan to contract the city of Cambridge install a small set of nodes (≈ 25) on streetlights between our two facilities. This initial network will be accessible from both laboratories. In year three we will do a larger city deployment (≈ 50 total nodes). On the newer nodes, we will attach sensors selected and provided by our collaborators from the school of public health to conduct their pollution monitoring studies. Finally in year 4 we will expand the testbed to ≈ 100 nodes. All nodes will be inter-operable, but hardware varies by year to take advantage of Moore's Law and better, cheaper 802.11 radios and sensors over time.

[8] `http://www.orbit-lab.org`

6 Results from prior NSF support

PI Welsh has recently been awarded a Small Grant for Exploratory Research (SGER) entitled "A Wireless Sensor Network for Monitoring Volcanic Eruptions" (CNS-0531631, $55,843, 7/1/05–6/30/06). This is seed funding for a project to deploy a wireless sensor network on Tungurahua Volcano, Ecuador, to measure seismic and infrasonic signals from the erupting volcano [45]. We are collaborating with seismologists at UNH and UNC on this project. We are currently in the process of designing the hardware and software for this system and anticipate undertaking the deployment later this year.

Welsh is also listed as Senior Personnel on NSF grant #ACI-0330244, "Hourglass: An Infrastructure for Sensor Network Applications" (9/15/2003–8/31/2007) but is not receiving any funding from this project. This project is developing an Internet-based infrastructure for collecting, filtering, and delivering data from multiple geographically-diverse sensor networks [18, 40].

Co-PI Ezzati has designed and led field research projects on household energy technology, indoor air pollution, and health in sub-Saharan Africa and China, with novel approaches to integrating data on environmental and behavioral variables. He was the lead scientist for the Comparative Risk Assessment (CRA) project at the World Health Organization which involved more than 100 scientists from multiple disciplines worldwide.

Ezzati has led multiple innovative research efforts that combine quantitative and qualitative methods for assessing exposure to air pollution, including developing new measurement instruments and techniques. For example, Ezzati has designed and conducted a field study in rural Kenya that quantified exposure to indoor air pollution, and its technological and behavioral determinants. This study examined the role of spatial and temporal microenvironments, using data from 210, 14- to 24-hour days of continuous real-time monitoring of pollution, spatial dispersion of smoke inside the house, and the location and activities of household members, supplemented with surveys on energy use and behavior. Together with two years of health records for the same group, this led to the first-ever quantification of the exposure-response relationship for indoor smoke from solid fuels along a continuum of exposure levels.

References

[1] Los angeles area plans to use sensors, cameras to record cars' pollution emissions. Associated Press `http://www.enn.com/today.html?id=8513`, August 2005.

[2] A. AuYoung, B. N. Chun, A. C. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *Proc. 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure*, October 2004.

[3] S. Bhola, R. Strom, S. Bagchi, Y. Zhao, and J. Auerbach. Exactly-once Delivery in a Content-based Publish-Subscribe System. In *2002 International Conference on Dependable Systems and Networks (DSN 2002)*, Bethesda, MD, June 2002.

[4] A. Carzaniga, D. Rosenblum, and A. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions of Computer Systems*, August 2001.

[5] Center for Information Technology Research in the Interest of Society. Smart buildings admit their faults. `http://www.citris.berkeley.edu/applications/disaster_response/smartbuil%dings.html`, 2002.

[6] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proc. the Workshop on Data Communications in Latin America and the Caribbean*, Apr. 2001.

[7] R. Chand and P. Felber. A scalable protocol for content-based routing in overlay networks. Technical Report RR-03-074, Institut EURECOM, February 2003.

[8] S. Chandrasekaran, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.

[9] J. Chen, D. DeWitt, F. Tian, and Y. Wang. NiagraCQ: A Scalable Continuous Query System for Internet Databases. In *SIGMOD / PODS 2000*, Dallas, TX, May 2000.

[10] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik. Scalable Distributed Stream Processing. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.

[11] C. Elliott and P. Hudak. Functional reactive animation. In *Proc. ACM SIGPLAN International Conference on Functional Programming (ICFP '97)*, volume 32(8), pages 263–273, 1997.

[12] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. In *ACM Computing Surveys (CSUR)*, 2003.

[13] M. Hata. Empirical formula for propagation loss in land mobile radio services. *IEEE Trans. on Vehicular Tech.*, pages 317–325, August 1980.

[14] G. Hoek, B. Brunekreef, S. Goldbohm, P. Fischer, and P. van den Brandt. Association between mortality and indicators of traffic-related air pollution in the Netherlands: a cohort study. *Lancet*, 360:1203–1209, 2002.

[15] M. Hope and N. Linge. Determining the propagation range of ieee 802.11 radio lan's for outdoor applications. In *LCN*, pages 49–, 1999.

[16] R. Huebsch, J. Hellerstein, N. Lanham, B. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *VLDB'03*, Berlin, September 2003.

[17] A. Laouiti, P. Muhlethaler, A. Najid, and E. Plakoo. Simulation results of the olsr routing protocol for wireless network. In *1st Mediterranean Ad-Hoc Networks workshop (Med-Hoc-Net)*, 2002.

[18] J. Ledlie, J. Shneidman, M. Welsh, M. Roussopoulos, and M. Seltzer. Open problems in data collection networks. In *Proc. 11th ACM SIGOPS European Workshop*, September 2004.

[19] J. Levy, E. Houseman, L. Ryan, D. Richardson, J. Spengler, et al. Particle concentrations in urban microenvironments. *Environmental Health Perspectives*, 108:1051–1057, 2000.

[20] J. Levy, E. Houseman, J. Spengler, P. Loh, and L. Ryan. Fine particulate matter and polycyclic aromatic hydrocarbon concentration patterns in Roxbury, Massachusetts: a community-based GIS analysis. *Environmental Health Perspectives*, 109:341–347, 2001.

[21] J. I. Levy, D. H. Bennett, S. J. Melly, and J. D. Spengler. Influence of traffic patterns on particulate matter and polycyclic aromatic hydrocarbon concentrations in roxbury, massachusets. *Journal of Exposure Analysis and Environmental Epidemiology*, 13:364–371, 2003.

[22] C. Lochert, M. Mauve, H. Fusler, and H. Hartenstein. Geographic routing in city scenarios. *Mobile Computing and Communications Review*, 9(1), January 2005.

[23] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proc. the 5th OSDI*, December 2002.

[24] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, USA, Sept. 2002.

[25] MIT. Mit roofnet wiki. `http://pdos.csail.mit.edu/roofnet/doku.php?id=radios`.

[26] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query Processing, Resource Management, and Approximation in a Data Stream Management System. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.

[27] R. Newton, Arvind, and M. Welsh. Building up to macroprogramming: An intermediate language for sensor networks. In *Proc. Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, April 2005.

[28] R. Newton and M. Welsh. Region streams: Functional macroprogramming for sensor networks. In *Proc. First International Workshop on Data Management for Sensor Networks (DMSN)*, August 2004.

[29] C. Ng, P. Buonadonna, B. N. Chun, A. C. Snoeren, and A. Vahdat. Addressing strategic behavior in a deployed microeconomic resource allocator. In *Proc. 3rd Workshop on Economics of Peer-to-Peer Systems*, August 2005.

[30] H. Nilsson, A. Courtney, and J. Peterson. Functional reactive programming, continued. In *Proceedings of the 2002 ACM SIGPLAN Haskell Workshop (Haskell'02)*, pages 51–64, Pittsburgh, Pennsylvania, USA, Oct. 2002. ACM Press.

[31] D. of Homeland Security Technical Support Working Group (DHS-TSWG). Multi-sensor environmental monitor. `http://www.hsarpabaa.com/main/technical_support_working_group.htm`.

[32] H. Ozkaynak and J. Spengler. The role of outdoor particulate matter in assessing total human exposure. In *Particles in Our Air: Concentrations and Health Effects*. Harvard University Press, Cambridge, MA, 1996.

[33] P. Pietzuch and S. Bhola. Congestion Control in a Reliable Scalable Message-Oriented Middleware. In *Middleware 2003*, Rio de Janeiro, Brazil, June 2003.

[34] PlanetLab Consortium. Planetlab: An open platform for developing, deploying, and accessing planetary-scale services. `http://www.planet-lab.org/`.

[35] J. Polastre, R. Szewczyk, C. Sharp, and D. Culler. The mote revolution: Low power wireless sensor network devices. `http://webs.cs.berkeley.edu/papers/hotchips-2004-motes.ppt`, August 2004.

[36] R. Ramanathan and J. Redi. A brief overview of ad hoc networks: challenges and directions. *IEEE Communications Magazine*, pages 20–22, May 2002.

[37] R. Ramanathan and R. Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. *IEEE INFOCOM*, 2:404–413, March 2000.

[38] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, Heidelberg, Germany, November 2001.

[39] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *NGC 2001*, UCL, London, November 2001.

[40] J. Shneidman, P. Pietzuch, M. Welsh, M. Seltzer, and M. Roussopoulos. A cost-space approach to distributed query optimization in stream based overlays. In *Proc. 1st IEEE International Workshop on Networking Meets Databases (NetDB)*, April 2005.

[41] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, August 2001.

[42] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[43] A. WeatherBug. Real-world instruction and the world wide web: How weatherbug achieve improves student performance. `http://www.weatherbugachieve.com/assets/pdf/White_Paper_1104.pdf`.

[44] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *Proc. the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, March 2004.

[45] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proc. Second European Workshop on Wireless Sensor Networks (EWSN'05)*, January 2005.

[46] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: A wireless sensor network testbed. In *Proc. Fourth International Conference on Information Processing in Sensor Networks (IPSN'05), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, April 2005.

[47] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesa, R. Govindan, and D. Estrin. A wireless sensor network for structureal monitoring. In *Proc. of the "ACM" Conference on Embedded Networked Sensor Systems (Sensys04)*, Nov. 2004.