CitySense: An Urban-Scale Wireless Sensor Network and Testbed

Rohan Narayana Murty*, Geoffrey Mainland*, Ian Rose*, Atanu Roy Chowdhury*
Abhimanyu Gosain†, Josh Bers†, and Matt Welsh*
* School of Engineering and Applied Sciences, Harvard University
† BBN Technologies, Inc.

Abstract

In this paper, we present the vision for an open, urban-scale wireless networking testbed, called CitySense, with the goal of supporting the development and evaluation of novel wireless systems that span an entire city. CitySense is currently under development and will consist of about 100 Linux-based embedded PCs outfitted with dual 802.11a/b/g radios and various sensors, mounted on buildings and streetlights across the city of Cambridge. CitySense takes its cue from citywide urban mesh networking projects, but will differ substantially in that nodes will be directly *programmable* by end users. The goal of CitySense is explicitly not to provide public Internet access, but rather to serve as a new kind of experimental apparatus for urban-scale distributed systems and networking research efforts. In this paper we motivate the need for CitySense and its potential to support a host of new research and application developments. We also outline the various engineering challenges of deploying such a testbed as well as the research challenges that we face when building and supporting such a system.

1 Introduction

Research progress in wireless and sensor networking in the last decade has been astounding. Recent developments include campus-wide and community-wide wireless mesh networks [12, 9, 14], in-depth studies of the dynamics of wireless networks in mobile and static settings [13, 18, 10], and real-world sensor network deployments in environments as diverse as forests [27, 4], active volcanoes [28], and the Golden Gate Bridge [19]. At the same time, to design, deploy, and evaluate novel wireless systems at scale requires substantial effort. Most research groups must be content with simulations or small-scale, homegrown test deployments to evaluate their ideas. Of the larger-scale wireless network testbeds [26, 30, 3, 29, 23], most are deployed in research labs or office buildings, representing a fairly narrow range of target environments. Moreover, one could argue that a lab-based testbed, regardless of scale or fidelity, is "uninteresting" from an application point of view, since the environment limits the range of potential use cases.

At the same time, recent advances in wireless mesh routing, such as Roofnet [12] and CuWIN [17], among others, have demonstrated the feasibility of achieving multihop wireless connectivity across large areas. Given the availability of robust wireless mesh networking, we want to ask the question, *how can we support the development and evaluation of novel wireless systems that span an entire city?* In the same way that development of TCP/IP enabled a vast range of new applications and business models, we wonder how urban-scale wireless mesh networking can support as-yet-unforeseen models of communication, computation, sensing, and interaction between users and their environment.

To this end, we envision developing an *open, urban-scale wireless networking testbed*, called CitySense, that will provide an invaluable community resource for driving research in wireless and sensor networking across an entire city. CitySense is currently under development and will consist of about 100 Linux-based embedded PCs outfitted with dual 802.11a/b/g radios and various sensors, mounted on buildings and streetlights across a city.[1] CitySense takes its cue from citywide urban mesh networking projects, but will differ substantially in that nodes will be directly *programmable* by end users. The goal of CitySense is explicitly not to provide public Internet access, but rather serve as a new kind of experimental apparatus, much in the same way that PlanetLab [24] has supported research in Internet-based distributed computing.

CitySense is designed to support research into new wireless mesh routing protocols, sensor networking at urban scales, new distributed algorithms for in-network data processing and aggregation, novel programming abstractions, and complete applications leveraging the emplacement of the CitySense nodes and attached sensors throughout a city. By providing an open infrastructure, CitySense can be readily customized to support a wide range of applications. Users can reprogram and monitor CitySense nodes via the Internet, allowing diverse research groups to leverage the infrastructure remotely.

Of course, there are a range of research problems we must address in order for CitySense to be a success. Supporting reprogrammability and multiple concurrent experiments comes at the potential cost of decreased reliability and increased resource contention. Moreover, since most CitySense nodes will only be able to communicate via a wireless mesh, we must ensure that experiments do not disrupt the backchannel (hence the need for two radios). Remote monitoring and administratation of CitySense nodes will require great care, as we cannot ensure physical access to nodes for routine maintenance or repairs. Frequent software updates (both for administrative tasks and end-user experiments) demands an efficient approach to over-the-air reprogramming. Finally, in the con-

[1] Our current deployment target is Cambridge, MA.

text of CitySense, due to bandwidth and computational restrictions, merely providing users with bare-bones *ssh* access to each node is unlikely to be either robust or efficient. Hence, we must consider appropriate programming models and runtime services to better support application deployment.

In this paper, we outline our vision for the CitySense platform and how we believe it will transform wireless and sensor networking research. We begin with motivation and a discussion of related work in Section 2. In Section 3 we sketch the design of the CitySense architecture and describe our current prototype implementation. In Section 4 we discuss the major research challenges that we face in designing and deploying CitySense. Finally, Section 5 concludes our discussion.

2 The Case for CitySense

Much like how PlanetLab has enabled large scale internet research, we believe CitySense as a testbed, can enable large scale sensor and wireless networking research in a real-world urban setting.

Two key features of CitySense are its city-wide deployment and its ability to monitor the physical world via sensors. These features, coupled with significant computational resources on each node, open up a host of new application and research domains. We outline some of them below:

Citywide publish/subscribe: Each CitySense node can receive, maintain, and compute new state which can then be pushed out to other nodes in the surrounding environment. These capabilities can be used for developing publish/subscribe applications, for example, supporting mobile users or vehicular network applications throughout the city. The AdTorrent [22] and CarTel [2] projects are studying the effect and feasibility of prepositioning content for mobile users in an urban setting. An infrastructure such as CitySense would serve as an ideal back end in such a setting. CitySense nodes could track the trajectory of a moving vehicle (or 802.11 device), estimate a probable future trajectory, and then pre-position content which is finally delivered to the moving vehicle. Likewise CitySense nodes could be used as collection and aggregation points for data collected by a vehicular network; providing not just network connectivity but also localized computational resources throughout the city.

Location integrated applications: A host of new collaborative applications can leverage user and wireless device locations. For example, consider a "Where am I?" applet for a social networking site, such as Facebook, that allows a user sign up and display their locations during the day, or a system that would permit users in nearby physical locations to post and read "digital graffiti" [9, 7].

Real-time network monitoring: CitySense has the potential to open up large scale realtime network monitoring of the various city-wide WiFi networks, for studying performance and deployment characteristics, analysing security risks, and identifying malicious behaviors. Much like Jigsaw [15] and DAIR [10], which are intended to monitor and diagnose indoor enterprise networks, CitySense could be used to monitor chaotically deployed WiFi networks in and around a city.

Sensornet applications: We expect various CitySense nodes to feature a diverse set of sensors, including weather conditions, air and water pollutants, biochemical agent concentrations, and more. In contrast to most work in wireless sensor networks, which is concerned with severe power constraints and hence limited CPU, memory, and bandwidth, CitySense represents a regime with much more substantial per-node resources. As such we envision CitySense acting as a testbed for future sensor platforms and applications that can leverage the increased capacity.

Evaluation: Apart from the applications described above, CitySense intends to provide a realistic evaluation testbed for wireless and sensor networking researchers themselves. For example, we hope that CitySense provides a valuable test harness for studying real systems on an urban scale, supporting research in wireless mesh routing, distributed algorithms, sensor fusion and data aggregation, and programming models.

While it would be possible to leverage CitySense for providing public Internet access, this is not our focus. A number of successful projects, including Roofnet [12], CuWin [17], and others [1, 8], have demonstrated the efficacy of wireless mesh for providing urban-scale network connectivity. Our goal is to build upon these efforts and provide a programmable, customizable research facility. Shifting the focus away from public Internet access frees us from the requirement to provide a stable, high-performance mesh focused on serving Web pages and VOIP calls. Indeed, the requirements of a production-quality wireless mesh run counter to our goal of offering an open, programmable environment, which we expect some users to break.

There exist other open research testbeds in the systems and networking community that have each come into their own in the recent past. Apart from the well-known PlanetLab [24] and Emulab [30] systems, significant wireless and sensor testbeds include ORBIT [26], Kansei [5], MoteLab [29], Mirage [23], and Mint [6]. These systems differ in terms of scale, node capabilities, programming model, and so forth; though all of them are indoor, laboratory-based testbeds, typically using wired backchannels to each node for remote management.

3 The CitySense Architecture and Prototype

The impetus for CitySense arose from recent work on city-wide wireless mesh networking, including the RoofNet [12], CUWin [17], and TFA [14] projects to provide connectivity to communities using inexpensive wireless equipment. While those projects were focused on the networking layer for providing general-purpose Internet connectivity, the key realization is that mesh router nodes (often based on embedded Linux PCs) could be augmented with a range of sensors as well as opened up to remote programming by end users.

Realistic sensor network applications are now demanding much larger networks, distributed over wider areas, with higher data rates and more sophisticated processing requirements. Large-scale monitoring of pollutants in an urban environment, is such an example. 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

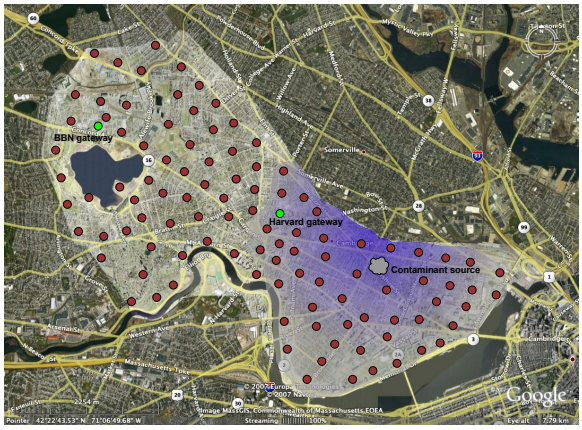

Figure 1: **Conceptual Deployment of Sensor Nodes in Cambridge, MA.**

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, to cover a large urban area the distance between any two sensors must be generally greater than the (ideal) 100m obtained by low-power WPAN (802.15.4) zig-bee 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.

Over the next few years, we plan to deploy approximately 100 sensor nodes distributed throughout an urban area. Our current deployment target is Cambridge, MA (see Figure 1) and is pending approval by the municipal government. Each node is linked to its neighboring nodes via 802.11-based wireless mesh. We have designed an enclosure that enables a sensor node to be mounted on a city streetlight, which also provides power to the node.

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, multiple environmental sensors (including temperature, humidity, wind speed and direction), and high-bandwidth 802.11 radios outfitted with high-gain omnidirectional antennas.

A unique feature of the CitySense design is the availability of power from the streetlight mounting. While past sensor networks have by necessity mandated very lightweight processing and communications power [25], we are not constrained by battery longevity. This freedom enables us to opt for the powerful Soekris net4826 motherboard (Figure 2). Matched with a Linux based OS, the Soekris platform enables rapid development using standard software tools. Coupled with high transmit power 802.11a/b/g miniPCI radio cards, our sensor nodes combine ample local processing power with high-bandwidth wireless connectivity.

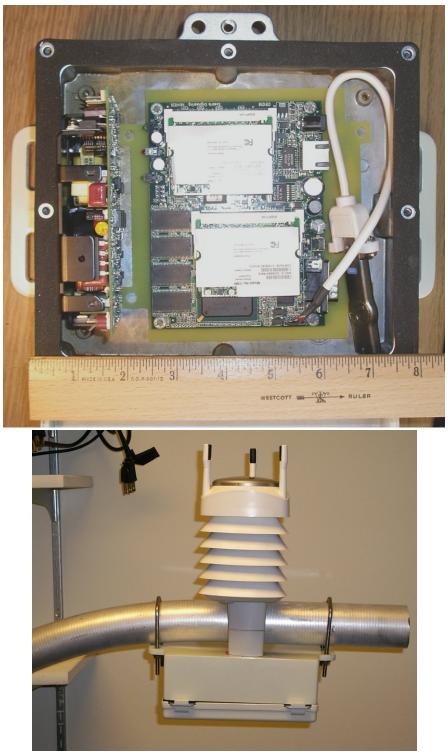

Figure 2: **Sensor Node Hardware. Top: The Soekris net4826 mounted in enclosure with power supply at left. Bottom: NEMA-6 weatherproof enclosure mounted on streetlight armature with Vaisala WXT510 Weather Sensor attached (WiFi antennas not shown).**

Apart from the sensor nodes themselves, the CitySense testbed includes wireline gateway nodes linking the wireless mesh to the Internet, as well as back-end servers providing services for reprogramming and monitoring the testbed, storing data generated by user jobs, and a Web-based interface to end users.

3.1 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.

3.1.1 Physical environment

Extreme weather, theft and vandalism, and malfunctioning hardware are realities that we expect to face when deploying our network in an outdoor urban environment. We hope to install CitySense nodes atop of streetlight fixtures, as shown in Figure 2. 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 continuous electrical power (AC) for the CitySense node.

For outdoor deploymet, a weatherproof housing is needed for both CPU and sensors. We have selected a NEMA-6 enclosure from PacWireless (see Figure 2).[2] During the winter

[2] `http://www.pacwireless.com/products/DCE.shtml`

of 2006-2007, we measured the temperature differential between the environment and the inside of a node powered via power over Ethernet (PoE). During cold days, we recorded a +10°C average differential. The presence of the power supply within the enclosure should provide a large enough differential that operation will be possible on very cold days. We plan to perform tests this summer on how well the unit performs in high-temperatures to determine if some form of venting is needed.

Our meteorological sensor, the Vaisala WXT510, ensures low maintenance by having no moving parts and by providing a heater that allows operation in air temperatures ranging from -52°C to +60°C.

3.1.2 Network Coverage

The streetlight mounting benefits network connectivity between CitySense nodes in two ways. First, it leverages the natural line-of-sight paths provided by roadways. 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 as well as the probability of interference from non-CitySense ground-based 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 roughly less than half the range for the radios. This ensures RF overlapping such that two nodes will not lose connectivity should a single intermediate node fail. Our initial spacing will be based upon estimates, however, we will experimentally validate our estimates prior to deployment.

3.1.3 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 WPA 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 distribute new keys to the nodes using a secure transport protocol.

3.2 Software environment

Each node runs an embedded Linux distribution, enabling us to use many available tools for network monitoring and management. For mesh connectivity, we are currently making use of the Optimized Link State Routing protocol (OLSR), since it has been shown (via simulation) to scale well in networks with more than 100 nodes [20]. We plan to experiment with alternative mesh routing protocols as well.

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 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 control network is lost for more than a predefined period of time.

3.3 Sensor Hardware

We plan to select sensor hardware based on criteria such as form factor, maintenance overhead, cost, and relevance to research activities of prospective users. For the current testbed, we have selected the Vaisala WXT510 weather transmitter, shown in Figure 2, 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 connects to the Soekris motherboard via its serial port.

On some nodes we will install carbon dioxide sensors, the Vaisala GMP343. In the future, carbon monoxide and polycyclic aromatic hydrocarbons (PAH) detectors may be added to the baseline depending on the needs of supported research projects. An option we are exploring is to incorporate and support new sensor technologies as they emerge.

3.4 Monitoring and management

The current administrative interface to our testbed is fairly low level and still relies heavily on standard UNIX tools (e.g., using *ssh* into each node and running commands as root). This is clearly not a scalable or robust solution to running a large testbed and we are planning development of further tools to simplify this process.

We have implemented a preliminary monitoring daemon that runs on each node and periodically runs a set of scripts to collect information on node state, such as uptime, routing tables, and statistics on network performance (e.g., an all-to-all ping and bulk transfer measurement performed hourly). The output of each script is routed to a central server node where it is logged to a database. A simple web interface provides real-time access to the network statistics. We have also implemented a simple approach to updating software on each node based on an *rsync* tree, described further in Section 4.

4 Research Challenges

Our initial prototyping work on CitySense has revealed a range of research challenges that we must address in order to deploy the full system. Broadly, the following factors give rise to most administrative research challenges in CitySense.

Management via wireless: Unlike most existing network testbeds, most CitySense nodes are expected to be managed via multihop wireless links. Each CitySense node will support two 802.11 a/b/g radios, where one radio can be devoted to the "management mesh" and the other radio used for experimental purposes. Few nodes will have wired backchannels, owing primarily to the high cost of running wired connections to each of the nodes. We believe this aspect of CitySense represents the evolution of large-scale wireless systems. In essence, the research challenge here is being able to manage a large wireless deployment in the wild via a multihop wireless backhaul. This raises questions of the reliability and bandwidth that can be expected of the mesh, which has broad implications for all management aspects of the system, including installing software updates, syncing the remote node's file system, and collecting data generated by experiments. This

challenge is exacerbated by the fact that we will not have physical access to CitySense nodes once they have been deployed: one cannot simply hit the reboot switch on a node mounted on a streetlight owned by the city. Further, a multi-hop testbed such as this will face the parking lot problem [14].

Deployment in the wild: CitySense must coexist with a wide range of 802.11 networks already deployed throughout the city, and must continue operating despite wide fluctuations in radio channel utilization, interference, and environmental variations, including the legendary Boston winter. To extend range between CitySense nodes, we are making use of high-gain omnidirectional antennas, although this also increases their sensitivity to (and impact on) ambient traffic. In our initial measurements, we have seen significant variations in achievable throughput, ranging from as much as 20 Mbps to as little as 1 Mbps over a short (40 meters) link between two nodes on the same rooftop! As the testbed scales up to cover various parts of the city of Cambridge, the system must adapt to highly variable conditions.

Resource constraints: While not severely power constrained, CitySense nodes do exhibit resource constraints in terms of the CPU, memory and bandwidth availability. Although the CitySense nodes run Linux, it would be a mistake to treat them as conventional desktop or server systems: with only 128 MB of RAM, a few poorly-designed applications could rapidly saturate available memory. The overhead imposed by the management software must be accordingly low to avoid interfering with experiments.

We believe a synthesis of these factors gives rise to a host of system challenges that are unique to CitySense, and that we plan to tackle as part of our work on this project.

Application programming model: It is tempting to treat CitySense as simply a batch of conventional Linux PCs, giving users login accounts and allowing them to *ssh* into each node to upload and run applications. Such an approach ignores resource and bandwidth limitations, not to mention node and link failures. Experience with application design and deployment in PlanetLab suggests that we can do better, by providing an appropriate set of tools and programming APIs to support CitySense application development; for example, a simple set of APIs to expose the physical location of each CitySense node, the network topology and link characteristics between various nodes.

Applications could make use of such a framework to decide where to cache data or perform aggregation. Such operations, in an urban wireless network could be location sensitive. For example, a particular application may choose to cache data on nodes that are located in a particular region. However, we do not intend to limit application developers to adhere to such frameworks and we wish to merely explore the option of providing such frameworks. On the other hand, restricting the programming model also yields a measure of control over the application's behavior, e.g., for resource containment.

Another aspect to explore is the possibility of providing varying "service levels" on CitySense with differing programming interfaces, depending on application needs.

Resource management and sharing: Closely related to the programming model is how resources will be shared across multiple users and applications. With such limited memory, CPU, and bandwidth resources on nodes, this problem is even more pressing than in less resource-constrained systems, like Planetlab. PlanetLab is adopting a containment approach based on Xen [11], although it is unlikely this will be effective on the embedded PCs used in CitySense. We are also concerned with how resource sharing might affect experimental results, for example.

One approach is to eschew timesharing in favor of a batch-scheduling policy; space-sharing can be accomplished by permitting different jobs to run on different partitions of the testbed. We could potentially allow lightweight jobs (with strictly enforced resource limits) to be intermingled with batch jobs. Experience with real applications will give us a better feel for which policies are most appropriate.

Reliability and failure recovery: Unlike other large-scale systems testbeds, such as PlanetLab [24] and Emulab [30], CitySense must rely chiefly on wireless communications to the individual nodes, and we cannot assume that we will have physical access to nodes to reboot or repair them in case of software failure. This suggests that we need to devote a significant effort to ensure that the base CitySense platform is robust, and that individual nodes can be restored to a "known good" state automatically following a failure. We plan to employ both a software watchdog timer as well as a hardware *grenade timer* that physically reboots the node at a preset time each day, regardless of its state. Further, such timers must be staggered to prevent a total network outage. This approach trades off slightly reduced uptimes for a high assurance that a node can be returned to a known state each day.

Wireless mesh routing: We wish to stress that the focus of the CitySense project is *not* to innovate on wireless mesh routing (although we hope that external users may wish to use CitySense for this purpose). Rather, we intend to leverage previous work on robust urban mesh routing [18, 16, 12, 17] as much as possible to build the testbed. As part of this, we also need to think about the impact of mutual interference between CitySense and pre-existing networks on channel and route selection. Further, due to various constraints around the city (for example, the streetlights at Harvard square are gas powered and decorative - hence we are not allowed to mount PCs on them) we could end up with partitions in the network. We expect to connect such partitions with 900Mhz long range radios and this may impact various routing decisions.

Software updates: Maintaining software consistency across 100+ CitySense nodes raises challenges in terms of dealing with failed nodes and network links; potentially frequent updates from many users; and the impact of software updates on node reliability. To avoid high network load for propagating updates, we have been experimenting with the use of a *spanning tree*, wherein a central server node pushes updates to a selected set of "seed" nodes, and those nodes push updates to their children, and so forth, making use of the *rsync* tool for efficient propagation of updates at each hop. Each update to the node filesystem will be represented by a monotonically increasing *version ID*, allowing a node to easily determine whether it has the latest software installed. Each user will have access rights to an isolated subtree of the node's filesystem; critical system files can only be updated by the administrator. Rather than push updates on demand,

we expect to batch all updates each day into a single bulk operation. Finally, we plan to maintain a *golden image*, on a separate bootable partition, which nodes will fall back to in the event of reboot or failure, which effectively disables all user applications and only permits the node to contact a central server for administrative control.

5 Conclusions

Mobile and wireless networking are emerging as a commodity in homes and workplaces, and increasingly found in other public spaces. As a result, usage models of computing are branching far beyond the proverbial user at a desk, supporting nomadic users with laptops, PDAs, and smartphones. At the same time, the ability to instrument our environment at high density with a broad range of sensors opens up new directions for merging observation and interaction with the physical world. To support continued innovation in these areas, we believe that it is absolutely critical to develop large-scale testbeds and development environments.

We envision CitySense as a unique experimental facility that will at once spur development into new application domains for wireless and sensor networking, as well as provide an evaluation platform with significant scale and realism. In the future, we anticipate that CitySense will play a role in the larger GENI effort proposed by the NSF, acting as one of many open wireless testbeds available to the GENI community. Still, substantial research challenges remain for deploying and maintaining such a system. Apart from the inherent reliability and performance issues with wireless multihop routing, we must be concerned with managing limited node resources; efficient and frequent software updates to the testbed; providing appropriate programming abstractions; and contending with failures and the inherent variations in the wireless environment throughout the city.

References

[1] Berlin roofnet. `http://sarwiki.informatik.hu-berlin.de/BerlinRoofNet`.

[2] Cartel. `http://cartel.csail.mit.edu/`.

[3] Exscal sensor network testbed. `http://ceti.cse.ohio-state.edu/exscal/`.

[4] James reserve data management systems. `http://dms.jamesreserve.edu/`.

[5] Kansei: Sensor testbed for at-scale experiments. `http://ceti.cse.ohio-state.edu/kansei/`.

[6] Mint, an autonomic reconfigurable miniaturized mobile wireless experimentation testbed. `http://www.ecsl.cs.sunysb.edu/mint/`.

[7] Place lab. `http://www.placelab.org/`.

[8] Technology for all - rice wireless mesh deployment. `http://tfa.rice.edu/`.

[9] Ucsd active campus. `http://activecampus.ucsd.edu/`.

[10] P. Bahl, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, , and B. Zill. Dair: A framework for managing enterprise wireless networks using desktop infrastructure. In *Proc. the 4th ACM Workshop on Hot Topics in Networks (HotNets-IV)*, November 2005.

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

[12] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *Proc. Mobicom 2005*, August 2005.

[13] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. Measurement study of vehicular internet access using unplanned 802.11 networks. In *Proc. MOBICOM 2006*, 2006.

[14] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement driven deployment of a two-tier urban mesh access network. In *Proc. ACM MobiSys 2006*, Uppsala, Sweden, June 2006.

[15] Y.-C. Cheng, J. Bellardo, P. Benko, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *Proc. SIGCOMM 2006*, 2006.

[16] T. Clausen and P. Jaquet. RFC 3626 - Optimized Link State Routing Protocol (OLSR). `http://www.faqs.org/rfcs/rfc3626.html`, October 2003.

[17] CUWiN Foundation. `http://www.cuwireless.net/`.

[18] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh network. In *Proc. MOBICOM 2004*, 2004.

[19] S. Kim, S. Pakzad, D. Culler, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proc. Fourth International Conference on Information Processing in Sensor Networks (IPSN'07)*, April 2007.

[20] 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.

[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, massachusetts. *Journal of Exposure Analysis and Environmental Epidemiology*, 13:364–371, 2003.

[22] A. Nandan, S. Tewari, S. Das, M. Gerla, and L. Kleinrock. Adtorrent: Delivering location cognizant advertisements to car networks. In *Proc. Third IEEE/IFIP Annual Conference on Wireless On-demand Network Systems and Services (WONS'06)*, January 2006.

[23] 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.

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

[25] 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.

[26] D. Raychaudhuri et al. Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC 2005)*, 2005.

[27] 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.

[28] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. OSDI 2006*, 2006.

[29] 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.

[30] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.