Kansei: A Testbed for Sensing at Scale

Emre Ertin†, Anish Arora†, Rajiv Ramnath†, Mikhail Nesterenko‡
Vinayak Naik†, Sandip Bapat†, Vinod Kulathumani†,
Mukundan Sridharan†, Hongwei Zhang†, Hui Cao†

† The Ohio State University ‡ Kent State University
Columbus, Ohio Kent, Ohio

ertine@ece.osu.edu, anish,ramnath@cse.ohio-state.edu, mikhail@cs.kent.edu
naik, bapat, vinodkri, sridhara, zhangho, caohu@cse.ohio-state.edu

ABSTRACT

The *Kansei* testbed at The Ohio State University is designed to facilitate research on networked sensing applications at scale. *Kansei* embodies a unique combination of characteristics as a result of its design focus on sensing and scaling: (i) Heterogeneous hardware infrastructure with dedicated node resources for local computation, storage, data exfiltration and back-channel communication, to support complex experimentation. (ii) Time accurate hybrid simulation engine for simulating substantially larger arrays using testbed hardware resources. (iii) High fidelity sensor data generation and real-time data and event injection. (iv) Software components and associated job control language to support complex multi-tier experiments utilizing real hardware resources and data generation and simulation engines. In this paper, we present the elements of *Kansei* testbed architecture, including its hardware and software platforms as well as its hybrid simulation and sensor data generation engines.

Categories and Subject Descriptors: I.6.0 [Simulation and Modeling]: General

General Terms: Measurement, Experimentation.

Keywords: Sensor Network Testbed, Sensor Modeling, Hybrid Simulation.

1. INTRODUCTION

Research on embedded wireless sensor networks has gravitated towards experimentation with current hardware and software platforms, with testbeds becoming the preferred basis for experimentation. Experience from early field experiments has reinforced the message of tight interdependence between the sensing application and the network and middleware layers; this is primarily due to resource limitations, hardware variability, and sensitivity to environment characteristics. As a result there is growing recognition that end-to-end debugging, validation, and integration of WSN applications requires joint consideration of sensing/environmental domain characteristics *and* networking-hardware-software platform properties.

The *Kansei* testbed at The Ohio State University is designed to facilitate research on networked sensing applications at scale. The basic idea is to couple one or more generic platform arrays that support a broad set of users, with multiple domain-specific sensing platform arrays. One implication of this idea is that *Kansei* needs to be extensible to readily add new platforms—especially domain-specific ones. Towards addressing the scaling challenge, the basic idea is to use arrays that are large enough so as to mirror deployment scale and, if they are not large enough, to capture sensing/radio phenomena at a resolution that enables their scaling via software in a high fidelity manner. We have been developing the *Kansei* facility since Spring 2004, partly through equipment support obtained from DARPA for the *ExScal* project [1] as well as Intel Corporation and The Ohio State University. While a basic purpose for developing *Kansei* was to shorten the long cycle time of *ExScal* field-testing in multiple outdoor settings, we soon found *Kansei* to be supporting a significant number of diverse use cases and users.

Since *Kansei* has been made openly available, it is being increasingly used for research projects at Ohio State and elsewhere. Datasets resulting from experiments on the testbed are being used at several academic institutions. We have used the testbed in project-based graduate and undergraduate courses, including our undergraduate capstone courses, as well as in short classes for training XSM and Stargate users. *Kansei* has also assisted in transitioning software to industry partners, in part by getting them to execute validation tests on components being transitioned. The basic design considerations and the description of *Kansei* use cases including its role in the *Exscal* project is presented in an accompanying, invited summary paper [2]. In this paper, we present a detailed description of the *Kansei* design and its software/hardware components.

In summary, *Kansei* embodies a unique combination of characteristics as a result of its design focus on sensing and scaling: (i) Heterogeneous hardware infrastructure with dedicated node resources for local computation, storage, data exfiltration and back-channel communication to support complex experimentation. (ii) Time accurate hybrid simulation engine for simulating substantially larger arrays using testbed hardware resources. (iii) High fidelity sensor data generation and real-time data and event injection. (iv) Software components and associated job control language to support complex multi-tier experiments utilizing real hard-

IPSN'06, April 19–21, 2006, Nashville, Tennessee, USA.

ware resources and data generation and simulation engines. In the rest of the paper, we first review relevant work on testing of wireless sensor network applications. Next, we present the elements of *Kansei* testbed architecture, including its hardware and software platforms as well as its hybrid simulation and sensor data generation engines. We conclude with an overview of a recent demonstration based on *Kansei* and with future work.

2. RELATED WORK

Domain Testbeds. Domain-specific sensing testbeds include the NIMS [3] deployments at James Reserve, Merced Basin, and Wind River. NIMS testbeds are being used to observe natural environments using a hierarchy of nodes: a few untethered, some tethered but static, and notably others that move in constrained manners along a network of suspended ropes and, in the case of imaging sensors, via pan, tilt and zoom. Sensing fidelity, via autonomous self-awareness of sensing uncertainty, coordinated mobility, and adaptive sampling, are research priorities in this effort. By way of contrast, the testbeds deployed in Huntington Botanical Gardens, Avra Valley [4], Sonoma Redwood Forest [5] and Great Duck Island [6] use only static nodes for studying microclimates and fauna behavior; they have yielded experience in supporting long lived installations. The WISDEN testbed [7] deployed in an earthquake-affected building has enabled seismic structural health monitoring.

Platform Testbeds. Numerous 10-50 sensor node testbeds exist for supporting network and middleware research efforts; these include testbeds that supplement static nodes with mobile ones. A few of these support generic testability goals for a broad set of users: MoteLab [8] is an early example, consisting of 30 MicaZ nodes embedded in a office building, that supports development and testing of sensor network programming environments, communication protocols, system design, and applications. MistLab at MIT has about similarly embedded 60 Mica2/Cricket nodes, each with an programming board, which also provides power and an Ethernet connection. GNOMES [9], a low-cost hardware and software testbed, explores the properties of heterogeneous wireless sensor networks, to test theory in sensor networks architecture, and be deployed in practical application environments.

By way of comparison with the *Kansei* testbed, these testbeds typically have a stationary/mobile array that is relatively smaller than ours and they do not integrate with portable arrays. Coupling and logical/physical configuration of the arrays is key to allowing experimenters to access the environmental condition of their actual deployment space, as opposed to that of their stationary/mobile testbed space. Another distinguishing feature of our testbed is that it can be used in conjunction with model composition and hybrid simulation tools to perform large-scale sensing experiments with high fidelity.

In terms of experimentation on motes, the *Kansei* testbed software architecture is similar to Motelab in the underlying Open-Source Linux-Apache-MySQL-PHP PERL implementation technology, and in certain aspects of the software architecture (web-based scheduling interface, scheduling daemon) of the stationary array. *Kansei* differs in the design of its various software components (user management, database schema, daemon thread-management, interface to the hardware, reservation and data retrieval mechanisms and so on), and it also currently supports experimental setup across all three levels of the *Kansei* hardware (PC clusters, Stargate mesh network and motes).

Related to these testbeds are those that the mesh networking [10] and mobile ad hoc networking communities have recently been developing. ORBIT [11] is a laboratory-based wireless network emulator that uses two-dimensional grid of 400 802.11x radio nodes which can be dynamically interconnected into specified topologies with reproducible wireless channel models. Roofnet [12] is a 20-node 802.11b/g mesh network testbed for conducting 802.11 measurement experiments and studies of link characteristics and protocols. *Kansei*'s static and mobile Stargate nodes will continue to support experimentation that would be relevant to these communities, as a by product of its capabilities.

Simulators and hybrid simulation. Researchers find the convenience and scalability of high level simulation tools such as ns-2 [13], GloMoSim [14], and Prowler [15] during initial application development, even though their representation of reality is circumscribed. None of these simulators allow the designers to run the code of the target platform.

There are a number of simulators that compile and run TinyOS program. TOSSIM [16] and TOSSF [17] and SENS [18] allow the designer to compile the TinyOS program for Intel architecture and run the program on a PC. Avrora [19] emulates the instruction set of the mote's processor on a PC which provides an interesting Debugging option. All four tools are capable of simulating thousands of motes on a single PC. However, the mote hardware is simulated, hence simulation fidelity is an issue.

Few simulators allow running of the code on physical sensor nodes. SensorSim [20] is a modification of ns-2 which allows simulated sensor nodes to communicate with real sensor nodes through gateway machines. This capability extends the usefulness of a high-level simulator as a debugging tool. Yet the limitations of high-level simulators outlined above apply to SensorSim as well. EmStar [21] has a range of tools that can be used for simulation and deployment in sensor networks. EmStar simulates sensor nodes as separate processes running on a host PC. EmStar provides communication between simulated sensor nodes through Unix device files. This communication can be purely simulated or hybrid. In the latter case physical nodes are used to transmit the messages. To maintain coordinated execution of multiple simulated sensor nodes, EmStar relies on periodic resynchronization and the estimates of process execution speed. This task becomes rather difficult in the hybrid mode as coordination with physical nodes have to be factored in. Thus, there are inherent scalability and fidelity limitations in the simulation approach used in EmStar. *Kansei* hybrid simulation engine is designed to simulate large-scale experiments with emphasis on high fidelity on timing, application, radio and sensing environments and complements the prior work on simulation.

Sensor data generation and Injection. Modeling of the environmental processes that gives rise to sensor readings is the core problem in physical sciences. Using model generated sensor readings to drive embedded software/hardware systems is a well established test method in automotive industry and in general control system design, known as hardware-in-the-loop testing [22]. Bruce *et al.* [23] developed a middleware for the implementation of hardware-in-the-loop testing in wireless sensor network ap-

plications. They demonstrated the middleware using pre-generated sensor streams using empirically derived models of acoustic, infrared and magnetic sensors and white normal distributed noise. The traces are first broadcast to each mote and stored in their memory, then the motes execute the application using the stored sensor traces .

Kansei sensor data generation facility combines accurate physics based parametric models of the foreground signal with probabilistic models of background processes to capture spatial and temporal correlations in the background signal. The two modeling techniques are augmented by sample based models using a database of sensor traces collected through the various portable arrays to provide a richer set of high fidelity sensor signals.

In addition, for many sensor-actuator network applications the control actions directly influence the sensor streams experienced by the sensor nodes. For example, the position of a robot in the sensor network will determine the particular sensors that will detect its motion. For these applications pre-generated sensor streams are not appropriate. The sensor data generation and injection is accomplished on the fly in the *Kansei*, which enables dynamic experiments with mobile nodes where the user or the network actions can change the input processes in real time.

3. ELEMENTS OF KANSEI

Kansei consists of a set of hardware platforms, access to which is managed by a remotely-accessible Director framework. The environment supports several tools, for high-fidelity sensor data generation and "hybrid" simulation.

3.1 Hardware Infrastructure

Kansei's hardware infrastructure consists of three components: Stationary Array, Portable Array, and Mobile Array.

Stationary Array. The stationary array consists of 210 sensor nodes placed on a 15×14 rectangular grid benchwork with 3ft spacing. Each node in the stationary array consists of two hardware platforms: Extreme Scale Motes (XSMs) and Stargates (Figures 1 and 2).

XSM is a derivative of Berkeley prototype sensor nodes, that was developed by Crossbow and the Ohio State University DARPA NEST team for use in the *ExScal* Project. Each XSM is equipped with a 7.3 MHz 8 Bit CPU, 128 KB instruction memory and 4KBRAM. For communication, the mote uses a 433MHz low-power radio. The radio's reliable communication range is between 15 and 30 meters when placed on ground level. Each mote accommodates a variety of sensors. Sensors include a photocell, a passive infra red (PIR), a temperature, and a magnetometer sensor, as well as a microphone. The motes run TinyOS [24], which is a lightweight event-based operating system that implements the networking stack and communication with the sensors, and provides the programming environment for this platform.

Stargate is an expandable single-board computer with Intel's 400MHz PXA255 CPU running the Linux operating system. It also has a daughter-card which contains an interface to a mote and a number of other interfaces including RS-232 serial, 10/100 Ethernet, and USB. Its in-band communication is via an 802.11(b) wireless NIC card. We have measured the radio characteristics of outdoor environments for these specific 802.11(b) radios in extensive measurements in May 2004 at OSU Don Scott Airport and at the *Kansei* warehouse. The measured link characteristics were roughly consistent with log normal large scale fading with a slightly larger path loss exponent for indoors (Figure 3). Then we used the combination of an 20 dB RF attenuator, an omnidirectional indoor antenna and power control to simulate the link characteristics observed at 135 meters outdoor antenna separation with an indoor spacing of 1 meter. The resulting packet delivery rate in the testbed follow the general characteristics of the outdoor environment (Figure 4). Through power control we can configure experiments across the stationary array to have connectivity as high as 6 hops.

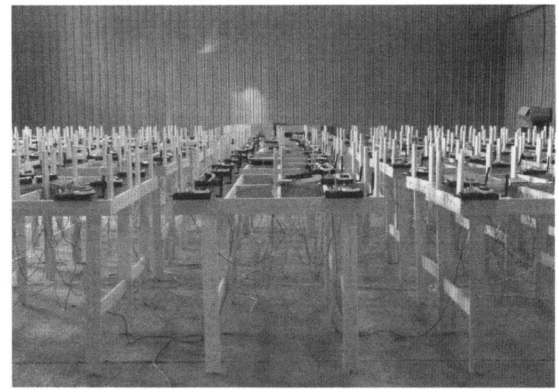


Figure 1: Stationary Array

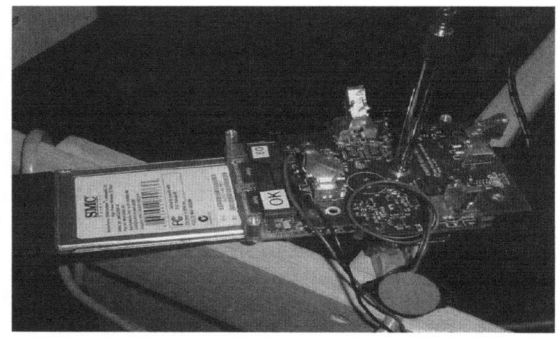
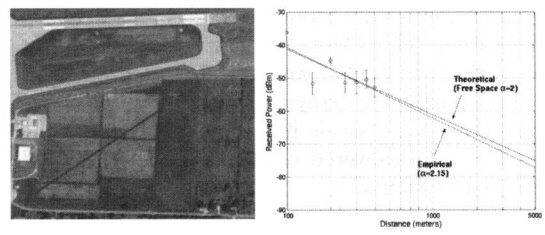


Figure 2: Stationary Array Node



Experiment Area Link Measurements

Figure 3: Outdoor Wireless Link Measurements

The XSM is connected to the Stargate through a dedicated 51 pin connector. The Stargate devices serve as integration points for the mote-level devices, providing channels to them for (a) data-collection, (b) data-analysis, and even (c) local sensor data generation and injection capability. The Stargate devices are connected through high speed net-

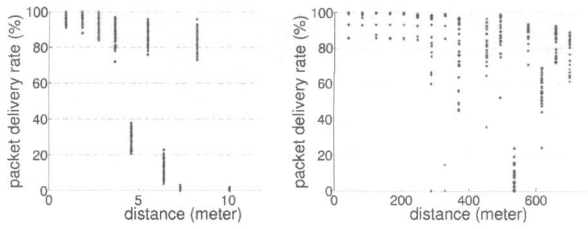


Figure 4: Packet Delivery Rate for testbed and outdoor environments

work switches to an Ethernet back-channel network which provides high-bandwidth connectivity to and from the nodes for management commands, data injection and extraction.

The Ethernet back-channel of the stationary array connects to a cluster of PCs. One PC serves as the primary server node for *Kansei* Director platform as well as for the remote access to *Kansei*. Other PCs are used for running visualizations, compute-intensive analysis, high fidelity sensor data generation, hybrid simulation and diagnostic analysis. We recently upgraded 150 nodes to contain a third hardware platform of TMote Sky nodes, which feature an IEEE 802.15.4 radio operating at 2.4 GHz and an integrated onboard antenna.

The nodes are placed on customized benchwork, with a Plexiglas plane layered on top to support the mobile nodes. Four high-resolution cameras (SONY SNC-RZ30N) with pan-tilt-zoom, and wireless as well as networked programmability, provides slew-to-cue capability for configurable image feeds of indoor testbed operation. These image feeds will serve sensing, visualization and, in some experiments, ground truth purposes.

Portable Array. The stationary array infrastructure is designed to be coupled with one or more portable arrays for in-situ recording of sensor data and field-testing of sensor network applications. Each portable array consists of domain specific sensors and generic software services for data storage, compression, exfiltration, time-synchronization and management.

Kansei currently includes a portable array of 50 Trio motes. The UC-Berkeley designed Trio integrates the XSM sensor board (acoustic, passive infrared, 2-axis magnetometer, and temperature) with TMote Sky nodes and a solar power charging system. The TMote Sky features an IEEE 802.15.4 radio operating at 2.4 GHz and an integrated on-board antenna. This particular array duplicates the sensors in the stationary array for at-scale high-fidelity sensing validation studies. The solar-powered charging makes it suitable for long-term deployments.

Mobile Array. This platform consists of five robotic mobile nodes that operate on the transparent Plexiglas mobility plane. The transparency of the plane allows light sources mounted under the robots to activate the photo-sensors of the nodes in the stationary array. Currently we use robots from Acroname, Inc. with built in motor-boards and a Stargate interface. A Stargate on each robot features an 802.11b radio with the optional attenuated antenna, as in the stationary array. In addition, each robot contains an XSM and TMote Sky node to communicate with the stationary array as well as to run native code for the XSM and TMote platforms.

Figure 5 depicts our current prototype. A robot localization service provides feedback to the experiment interface and enables us to inject sensor data to the stationary nodes at the robots current location. Alternatively, the experimentation tools enable experimenters to inject into each robot a separate sensor data stream depending on its location. This is to facilitate *cooperative* articulated sensing experiments where mobile agents sample any random sensor field adaptively.

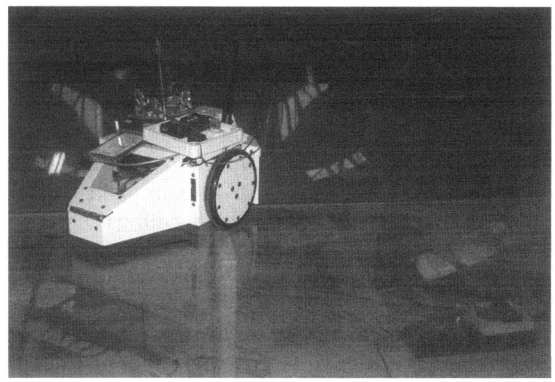


Figure 5: *Kansei* Mobile Node on the Stationary Array

3.2 Director: A uniform remotely-accessible framework for multi-tier WSN applications

The *Kansei* Director is an extensible software platform that enables integrated experimentation on the stationary array, portable arrays and mobile array.

To begin with, the Director provides the basic services of (i) experiment scheduling, deployment, monitoring and management for all array platforms and (ii) creation and management of testbed configurations in support of multiple-user and multiple-use scenarios, such as for allocation to experiments. Thus, "jobs" — potentially consisting of multiple WSN executables, scripts and data files — can be programmed to run on a specific configuration of the testbed for a specified length of time. The status of these jobs may be monitored during their execution. On their successful completion, output data can be retrieved. If their execution fails (such as when a node is unavailable at the time the job is started), the failure status is reported, in addition to which, the portion of the Director log relevant to the job may be retrieved.

Next, the complexity of network embedded applications is growing rapidly, yielding for instance applications that are multi-phase and that are reconfigured from time to time. WSN resources however are not growing at a rate that significantly exceeds application needs. Hence, unlike traditional network based systems, network embedded computing continues to involve operating networks "on the edge", as opposed to well within network capacity. We thus expect application-dependent optimization of resource utilization to be an important integration challenge. This implies awareness of network/embedded resources and rapid configuration of applications in accordance to available resources. The Director therefore provides a set of services for gathering the state of the *Kansei* testbed.

Given the availability of state information, reconfiguration at run-time is the next requirement, thus a core integration challenge is to support application management conveniently, both for human users as well as for mechanization. Thus, the Director also supports the orchestration of an experiment consisting of multiple phases. In *ExScal*, for example, a "localization" phase calculated and disseminated to each mote its (x, y) grid position, which was then stored in flash memory. The mote was then rebooted to a "sensing" phase, that initialized itself by reading this localization information. Complex multi-phase experiments especially occur when iteratively tuning the application to the environment in conjunction with tuning the middleware to the application.

The Director implements a core set of system-level utilities and run-time components. In addition, new *system* utilities and components can be configured, deployed and managed by the *Kansei* Director. Examples include tools for data injection (such as the Injector when the experiment requires the injection of the output data from a previous phase as input for the next one), health monitoring, and logging, and for all array platforms. Components could simply be specific middleware (such as for routing) or run-time components for implementing "reflective" applications - that, for example, monitor resource utilization on the node and reconfigure themselves appropriately. Note that the *development* of applications and system utilities and components is done outside the Director.

System administration services are also part of the Director. These services include user management (creation and deletion of users, and the assignment of access rights) and platform administration (such as the restarting or setting the network configuration of a node).

The Director remotely exposes these services (for experiment scheduling, configuration, deployment and management, system utility deployment and configuration and system administration). End-users access these services through a Web interface and programs through web-services.

Finally, the Director is designed to extensibly support both the current and the future variety of hardware and operating systems platforms across all tiers; we term this "platform plug-and-play". Each platform exposes a uniform set of services (with the *implementation* of each service being different for each platform). This uniformity allows the end-user of *Kansei* to essentially be platform agnostic in the specification of the experiment, including in its orchestration.

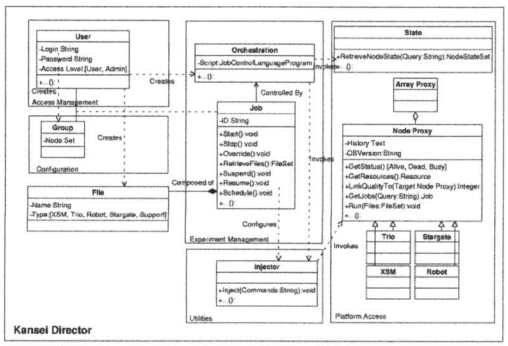


Figure 6: *Kansei* Director Architecture

Director Architecture: Figure 6 shows selected components of the architecture of the main *Kansei* Director. In the main Director, the Configuration subsystem manages testbed configurations (such as a topology and its nodes on the stationary array and portable arrays used in an experiment). The Access Management subsystem manages the levels of users and their access rights. The Platform Management subsystem abstracts the services of the arrays to enable platform plug-and-play through platform manifest files which are installed on *Kansei* when a new platform is incorporated into *Kansei*. The Director uses Orchestration services for the sequencing of steps within a multi-phase job. As the steps of an orchestration execute utilities and applications will be invoked.

Director Implementation: Director services are implemented not just by components that run on the top-level of the arrays, but also on nodes within arrays. For example, deploying an application on a mote involves invoking a director component on the Director server, which in turn invokes a component on the Stargate that serves as a gateway to the mote. Just as the Director is hierarchical, so are several of the *Kansei* utilities. For example, the Injector has a distributed implementation on the *Kansei* node hierarchy. The Injector has an Injector Manager at a PC for orchestrating event injections across the experiment topology, each Stargate node has an Injector for injecting events to applications running on it, as well as to serve as a bridge that injects events into the application on the mote. Each mote application has a Receptor component that receives injections and takes appropriate actions. Note that the component at each tier is not built in the same manner: e.g., the Director and the Injector could each be a standalone Linux process separate from the WSN application under test, but the Receptor has to be a NesC component, compiled and linked at build time into the application.

The main Director runs on a Linux server (which is also runs the web-server for the web interface). Test-bed scheduling, administration, management, and experimentation are implemented in a multi-threaded daemon (written in PERL) that uses scripts and utilities written as PERL modules and which encapsulate testbed services (such as *uisp* for XSM programming via the serial port of the Stargate). PHP modules implement the web-accessible testbed services, such as job-creation, storage of experiment data, and a testbed health-monitoring page. A MySQL database provides persistence for storing job configurations and user reservations. Data generated by jobs are stored on the server filesystem and may be retrieved by links on the web interface.

3.3 High fidelity sensor data generation tools

Sensor data generation is a key component of high fidelity design and testing of applications at scale. In addition to its utility in validation of applications and network services, it provides a theoretical basis for the design of algorithms for efficient sampling, compression and exfiltration of the sensor readings. Currently, *Kansei* users can generate sensor data fields of arbitrary size at high fidelity using two alternative methods: (i) scaling-up the data traces collected on the portable array to larger sensor arrays through sample based modeling tools, and (ii) synthetic data generation from parametric and probabilistic sensor models:

Sample Based Modeling Tools. For many sensor modalities the pyhysical phenomena of signal generation and prop-

agation is too complex for accurate parametric modeling and computationally feasible simulation. In these instances a generic sample based model can be used to simulate sensor readings at large scale. The model maintains a database of sensor snippets indexed by ground truth parameters collected for the source phenomena. To capture spatial correlations in sensor readings, the recordings are made simultaneously on an appropriately sized patch of sensors. Examples are: a) passive infrared energy recordings on a small mesh of sensors as a personnel intruder passes through a tile of sensors. Figure 7 shows snippet as indexed by the intruder movement (speed and direction); b) Acoustic energy recorded on a small mesh of sensors for a windy day indexed by the windspeed at that time and location, c) Signal energy, time and direction of arrival recordings for all neighboring sensor locations for a buzzer node indexed by their relative location to the source. Generation of the sensor data at the desired scale is accomplished by replaying the snippets with appropriate time and spatial shifts. We note that the replay is not in simple sequential order, because the underlying process can cause sensor readings in multiple tiles in overlapping time intervals. For example a person walking over a sensor tile will generate sensor readings before entering and as well as after leaving the tile. The required index parameters and time overlaps for playback can be exogenously determined as in the case of the intruder path and buzzer sequence or it can be generated from a probabilistic spatial model, such as an autoregressive Gaussian random field model for the windspeed.

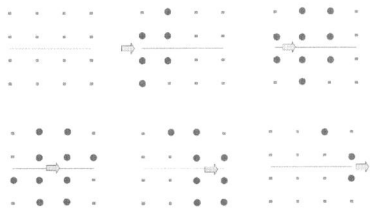


Figure 7: PIR data snippets indexed by the intruder position

Synthetic data generation using parametric models: For many sensor modalities the physical relationship between the sensor reading and the underlying natural phenomena is well understood and the sensor readings are dominated by the foreground signal. Consequently, sensor readings can be generated from a parametric model of the underlying phenomena. As part of the *Kansei* effort we have been developing physics based parametric sensor models for a variety of sensing modalities including models of passive acoustic, seismic, infrared and magnetic sensors. The physics based models are validated and tuned to include nonlinearity and saturation effects introduced by the actual hardware using portable array studies in various deployment environments. Here we review the example of magnetic sensors to highlight the features of the parametric modeling techniques.

Passive magnetic field sensor (or magnetometer) can detect perturbations in the earth's magnetic field caused by ferrous objects (such as vehicles and tools carried by personnel). Fields produced by ferrous objects are well understood and readily predicted. The earth provides a magnetic field which induces a magnetic moment in a ferrous object. The presence of this induced dipole moment creates a distortion in the earth's magnetic field. The magnitude of the variation in the magnetic field is inversely proportional to the cubed distance between the ferrous object and the magnetic sensor. This simple induced dipole model can be used to accurately predict sensor readings for a large ferrous object consisting of spatially distributed dipoles parametrized by their relative location with respect to the sensor. Figure 8 shows the readings on a two-axis magnetometer for a large truck at 5 meter distance as measured on two orthogonal axis, and the parametric model fit to the data using only two induced dipoles.

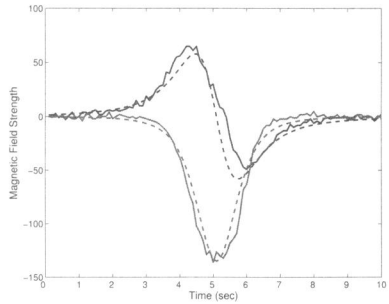


Figure 8: Parametric Modeling: Measured 2-axis magnetometer signal (solid) and its parametric model (dashed)

Probabilistic Modeling Tools. An alternative modeling strategy relies on accurate estimation of the spatial and temporal correlation of the sensor readings. Many sensor modalities can be modeled as time varying random Markov fields. Examples are temperature, gas, humidity and turbulent wind energy distribution. A simple instantiation of such a model is a Gaussian Markov Field with a time varying covariance matrix. In general estimating the correlation structure is not feasible for an N node array because it requires estimation of N^2 parameters. One can constrain the estimation problem by assuming a correlation distance and use a graphical model to restrain the correlations to a given set of neighbors. The difficulty of the covariance estimation problem is compounded by the fact that the covariance structure varies with time varying parameters of the underlying phenomena, such as local wind-speed, sun-light, target position. When appropriate we model the variation of these parameters exogenously, in other cases we use a prior distribution (e.g. auto-regressive process) leading to a hierarchical Bayesian model. To generate samples from the random model we use a generic method for simulation such as a Gibbs Sampler [25]. Figure 9 shows randomly generated sensor readings for wind energy as measured by a microphone. The parameters for the random process was estimated from a 200m x 1,000m data collection experiment over an hour in December 2004.

It is possible that a given sensor modality may require a combination of the previously discussed techniques. For example, a parametric model of the foreground process can be combined with a sample based model of the background process.

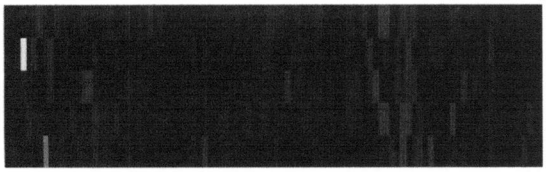


Figure 9: Simulated random field for wind energy

3.4 Hybrid Simulation

For simulation to be an effective tool in evaluating sensor network algorithms it has to correctly model the physical environment for radio signal propagation as well as adequately represent the application being run by the sensor network. *Kansei* features a high fidelity hybrid simulation capability where a PC simulation server is connected to the stationary or the portable array. In hybrid simulation, hundreds of virtual sensor nodes can be simulated using real radio hardware to communicate messages. This ensures fidelity of the simulator with respect to the radio propagation in realistic deployment environments.

Currently, the hybrid simulator is applicable to the Berkeley motes running TinyOS [24] applications. A part of the simulator is — TOSSIM [16]: a TinyOS simulator. The main simulator component is running on the PC. For hybrid modeling the simulator utilizes the out-of-band access to the physical sensor nodes on the stationary array. The simulator allows TOSSIM to run the application on the host PC but relays the communication and sensing requests to the physical motes connected to the PC. This is done by replacing the components that simulate communication and sensing in TOSSIM with components that handle the interaction with the motes.

Reconciling the timing of real and simulated events is one of the most challenging tasks of any simulator. Even in a pure simulator, the problem of simulating the execution of multiple concurrently running motes is far from straightforward. A pure simulator has to overlay the execution of the code of multiple motes on the processor of the host PC. Thus, the duration of the simulated events is distorted. To preserve the fidelity, advanced simulators, such as TOSSIM, divorce the simulated execution from the real time. TOSSIM maintains *simulated time* to measure the duration of the events. TOSSIM advances the simulated time as the motes make progress in their execution.

A hybrid simulator has to coordinate both real and simulated events. Thus, the problem of reconciling the real and simulated time arises. One approach is to allow the real events to occur at their own speed and periodically resynchronize the simulated part with real events. However, this approach has potential scalability and fidelity problems.

Instead, we use the simulated time as a scale of reference and dynamically translate the timing of real events to and from this scale as needed. When the simulation requires a real event, the simulator executes the following sequence. First, the simulator records the simulated timing parameters of this event. Then the simulator freezes the simulation, translates the parameters into real time, and executes the event in real time gathering the timing parameters. After the real event ends, the simulator translates the timing information back into simulated time and resumes the simulation.

As described above, it appears that the simulator needs one physical mote for each simulated one. Observe, however, that the mapping between the physical and simulated motes does not have to be static. This mapping can dynamically change throughout the experiment allowing the simulator to model an arbitrarily large network on a sensor array of a fixed size. The concept we use to enable dynamic mapping is tiling of the virtual sensor network. *Tile* is a repeated region of the physical network composing the virtual network. When a message is sent over the radio its effect is twofold: it can be received and it can interfere with the reception of other concurrently transmitted messages. Thus, if the physical tile is large enough to contain the sender and the all recipients of radio energy, then the other nodes of a large network are immaterial with respect to this transmission.

Observe that the effect of the message transmission can only be adequately represented if the sender is close to the center of the tile. However, the nodes on the fringes of the tile still need to keep sending concurrent messages to properly simulate the message interference. Hence, the tile is divided into the center *active zone* where the motes are sending and receiving messages and the border *interference zone* where the motes are sending messages to simulate the interference and recording the messages received from the motes in the active zone. When the the simulated mote requests a message transmission, the simulator continues to advance the simulated time for an approximate time it takes to send a message. During this time it collects the concurrent messages. The simulator then maps the sender of the original message to the center of the tile and the rest of the senders in the appropriate places in the tile, translates simulated time to real time, carries out the transmission, collects the results and resumes the simulation.

We have used hybrid simulation to simulate several representative applications and tested it at the scale of several hundred simulated sensor nodes. Figure 10 shows the results of a routing application execution. The physical tile ranges from 5- to 13-sensor-node tile. Notice that as the tile size increases, longer links become available.

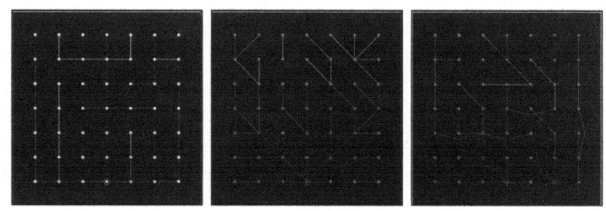


Figure 10: Routing trees for the Surge Application: Hybrid simulation with 5, 9 and 13 hardware motes

4. KANSEI IN ACTION: INTRUDER-INTERCEPTOR GAME

The portable, stationary, and mobile arrays can all be used simultaneously for large scale environment-in-the-loop experiments. In a recent demonstration for the DARPA NEST program, we used all three arrays for an intruder-interceptor game. The concept of operation includes two teams of players: Intruders and Interceptors. There is a long linear asset and a buffer zone around it where a dense WSN is deployed. Intruders try to reach or come as close as

possible to the long linear "goal". Interceptors, a team of autonomous mobile robots wait in the buffer zone and try to intercept the intruders before they reach the long linear goal. The sensor network provides tracking information and intruder assignments to the interceptor team.

For the demonstration, a portable array of 60 Trio Motes was deployed on a 200m x 30m area at Richmond Field Station (Figure 11). We collected PIR sensor traces from human subjects crossing the portable array and fed them to the sensor data generation engine (Section 3.3) to scale them up to simulate intruders on a larger 105 node subset of the stationary array. The scaled traces were injected into the stationary array, where a tracking algorithm was running in real time. The stationary array nodes communicated assignments and the calculated tracks of intruders to the robots placed on the mobility plane on top of the stationary array. Using the environment-in-the-loop experiments, we were able to evaluate different tracking services using energy, catch-distance, network load metrics under realistic deployment conditions.

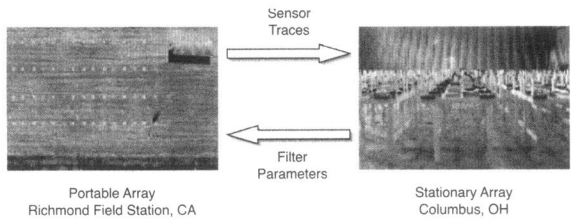


Figure 11: Hybrid Experimentation with the Portable and the Stationary Array

5. FUTURE WORK

Improvements are being continuously incorporated into *Kansei*. We are also presently evaluating new hardware, software platforms to add to the testbed, so *Kansei* can stay apace with significant evolution in sensor node hardware/operating systems. Repeated, remote, experimentation will be extended to *Kansei* mobile platforms through self-charging docking of the robots with a power source between experiments. The hybrid simulator functionality will be extended to other hardware architectures present in *Kansei*. Finally, we are conducting experimental studies to validate the effectiveness of simulated sensor traces in predicting performance at the application level. The testbed is now open for community use. Readers can find more information about *Kansei* on the web at http://www.cse.ohio-state.edu/kansei.

6. ACKNOWLEDGEMENT

This work was sponsored by DARPA contract OSU-RF #F33615-01-C-1901 and the National Science Foundation under Grant No. 0341703. Mikhail Nesterenko's work was sponsored in part by the NSF CAREER Award 0347485. We would like to acknowledge the contributions of Bill Leal, Taewoo Kwon, Brian Dupaix, Prabal Dutta, Chris Anderson, Nathan Stohs, John Wiesemann and Gavin Fox to the development of the *Kansei* testbed.

7. REFERENCES

[1] A. Arora *et al.*, "Exscal: Elements of an extreme scale wireless sensor network," in *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2005)*, Hong Kong, Aug 2005.

[2] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko, and W. Leal, "Kansei: A high fidelity sensing testbed," to appear in *Internet Computing*, April 2006.

[3] W. J. Kaiser em et al., "Networked infomechanical systems (NIMS) for ambient intelligence," *Book, Editor(s): J. Rabaey Collection: Ambient Intelligence*, 2004.

[4] K. A. Delin *et al.*, "Environmental studies with the sensor web: Principles and practice," *Sensors*, vol. 5, no. 1-2, pp. 103–117, 2005.

[5] D. E. Culler, "Toward the sensor network macroscope," in *ACM MobiHoc (Keynote Speech)*, 2005.

[6] A. Mainwaring *et al.*, "Wireless sensor networks for habitat monitoring," in *WSNA*, 2002.

[7] J. Paek *et al.*, "A wireless sensor network for structural health monitoring: Performance and experience," in *Proceedings of the Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, May 2005.

[8] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," in *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN-SPOTS)*,, April 2005.

[9] E. Welsh, W. Fish, and P. Frantz, "GNOMES: A Testbed for Low-Power Heterogeneous Wireless Sensor Networks," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Bangkok, Thailand, May 2003.

[10] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *IEEE Communications Magazine*, vol. 43, no. 9, Sep 2005.

[11] D. Raychaudhuri *et al.*, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *WCNC*, 2005.

[12] B. A. Chambers, "The grid roofnet: A rooftop ad hoc wireless network," MIT Master's Thesis, Tech. Rep., June 2002.

[13] J. Broch *et al.*, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Mobile Computing and Networking*, 1998, pp. 85–97.

[14] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A library for parallel simulation of large-scale wireless networks," in *Workshop on Parallel and Distributed Simulation*, 1998, pp. 154–161.

[15] G. Simon, P. Völgyesi, M. Maróti, and Á. Lédeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," in *IEEE Aerospace Conference*, 2003.

[16] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire tinyos applications," in *Proceedings of the first international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 126–137.

[17] L. F. Perrone and D. M. Nicol, "A scalable simulator for TinyOS applications," in *WSC*, 2002.

[18] S. Sundresh, W. Kim, and G. Agha, "SENS: A sensor, environment, network simulator," in *The 37th Annual Simulation Symposium (ANSS37)*, April 2004.

[19] B. Titzer, D. Lee, and J. Palsberg, "Avrora: Scalable sensor network simulation with precise timing," in *Proceedings of the Furth International Conference on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, April 2005.

[20] S. Park, A. Savvides, and M. B. Srivastava, "SensorSim: A simulation framework for sensor networks," in *Proceedings of MSWiM*, August 2000.

[21] L. Girod *et al.*, "EmStar: a software environment for developing and deploying wireless sensor networks," in *Proceedings of USENIX 04*, 2004.

[22] H. Hanselmann, "Hardware in the loop simulation as a standard approach for development, customization, production test," SAE paper 930207, 1993.

[23] D. Jia, B. H. Krogh, and C. Wong, "TOSHILT: middleware for hardware-in-the-loop testing of wireless sensor networks," http://www.ece.cmu.edu/~webk/sensor_networks/toshilt/.

[24] J. Hill *et al.*, "System architecture directions for networked sensors," *ACM SIGPLAN Notices*, vol. 35, no. 11, pp. 93–104, Nov. 2000.

[25] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.