The Networked Sensor Tapestry (NeST): A Privacy Enhanced Software Architecture for Interactive Analysis of Data in Video-Sensor Networks

Douglas A. Fidaleo
dfidaleo@ucsd.edu

Hoang-Anh Nguyen
anh@geekdom.net

Mohan Trivedi
mtrivedi@ucsd.edu

University of California San Diego
9500 Gilman Dr.
La Jolla, CA 92093-0434
1-(858)-822-0002

ABSTRACT

This paper details the architecture of a test-bed under development for secure sharing, capture, distributed processing, and archiving of surveillance data called the Networked Sensor Tapestry (NeST). The test-bed consists of core software modules including a centralized server, client interface library, a layered XML messaging scheme. Mobile hardware clients are interfaced to the NeST using a Tiny-OS based microcontroller with sensor data collected over a 1-wire data bus.

Maintaining subject privacy in video and other sensor monitoring scenarios can be imperative for the successful deployment of surveillance networks. Subject privacy is integrated into the architecture and can (if desired) operate as a buffer to the server core, denying access to identity specific information to any or all modules or operators. We introduce 3 fundamental privacy concepts: The *privacy buffer*: is a core component of the NeST server and utilizes programmable plug-in *privacy filters* operating on incoming sensor data to prevent access to or transform data to remove personally identifiable information. These privacy filters are developed and specified using a *privacy grammar* that can connect multiple low-level data filters and features to create arbitrary data-dependent privacy definitions. The utility of the architecture is demonstrated with a connection to a variety of hardware/software clients including PDA based client hardware, remote sensor interface devices, software modules for sensor data inferencing, data visualization, sensor control and data archival applications.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *current awareness systems, information networks, distributed systems.*

General Terms

Design, experimentation, and security.

Keywords

Surveillance architecture, video-sensor networks, privacy.

1. INTRODUCTION

Basic surveillance systems require access to multiple sources of hybrid data such as video camera streams, motion and light sensors. However specialized applications such as structural monitoring or first responder assistance may have additional sensor requirements such as temperature, seismic, smoke/chemical/radiation, or GPS data. The focus of this work is to develop hardware and software infrastructure to enable rapid prototyping and deployment of a broad range of distributed surveillance applications.

In the most trivial case, an application would simply retrieve information from the myriad sensors and relay the data to a central repository where it may be viewed or archived. It is more useful, however, if this data can be shared with other applications in real-time to allow these applications to perform further processing. Higher level semantic information can then be extracted and inserted back onto the network to be accessed by yet another application, or routed to its final destination point which may in fact be a visualization or archiving utility. By adding these semantic processors or so-called *virtual sensors* to a sensor network, the application scope of the network dramatically increases. A virtual sensor may be as simple as a compression algorithm operating directly on the sensor output, or as complex as an inference module that requires access to multiple sources of real-time or historical sensor data.

The Networked Sensor Tapestry (NeST) is an architecture for secure sharing, capture, distributed processing, and archiving of surveillance data. A combination of core software modules and hardware components, it is an extendible, modular, real-time test-bed for surveillance and interpretation of surveillance data.

The generality and flexibility of the NeST software architecture is extended to hardware interface modules. Using TinyOS based hardware and the 1-wire sensor interface, arbitrary low-data-rate sensors can be easily integrated with the NeST.

Quick response and remote monitoring of an accident scene, seismic event, or other critical incident is only possible if the surveillance infrastructure is in place before the event takes place. The key to maximizing the utility of surveillance infrastructure is therefore ubiquity. The general public, however, understandably resists mass surveillance due to concerns over privacy and misuse of information. Maintaining subject privacy in video and other sensor monitoring scenarios is thus imperative for the successful deployment of surveillance networks.

This paper introduces 3 fundamental concepts for maintaining privacy in surveillance and programmatically defining properties of private data.

(1) The *privacy buffer* is a shell around the server core that utilizes programmable plug-in filters operating on incoming sensor data to prevent access to private information or transform data to remove personally identifiable information.

(2) Data is labeled as private or non-private by one or more *privacy filters* plugged into the privacy buffer. Filters are specified for different types of data and are matched to incoming data.

(3) Privacy filters are specified with a *privacy grammar*. The privacy grammar allows an end user to construct new definitions of privacy based upon combinations of low level features and high level semantics derived from raw sensor data.

In this paper, the core ideas and components of the architecture will be presented. A preliminary implementation of the architecture is complete, allowing us to demonstrate its flexibility with a variety of hardware/software clients including PDA-based client hardware, remote sensor interface devices, software modules for sensor data inferencing, data visualization, sensor control and data archival applications.

2. PREVIOUS WORK

Size and cost of sensing and processing modules are rapidly decreasing, however power requirements still remain high for modest computational power. To battle this dilemma there has been increased interest in utilizing multiple small, low-power devices for sensor network applications. The EmStar work at UCLA is an extremely flexible tool for simulating and deploying embedded sensor network applications using Linux based micro-servers (Ipaqs) and TinyOS Motes [2].

Large-scale low-power sensor networks have great potential for ecosystem and structural monitoring and as such, significant research effort has been invested in developing architectures and operating systems to manage the low resource capabilities of such devices [2][10][11]. However, many compelling surveillance applications are currently vision based, requiring significantly more resources and more reliable connectivity than can be readily delivered by battery driven motes or even higher power micro-servers such as the Ipaq. Fortunately, such connectivity can be acquired using commercial-off-the-shelf (COTS) technologies, bringing surveillance applications out of specialized experimental environments, into the real-world.

Architectures for surveillance applications have also been widely explored. The MPI-Video infrastructure allows for fusion of multi-sensory data and models the statistical properties of layered events in the environment. Events models are built from video streams and can be queried through a server and archived [9].

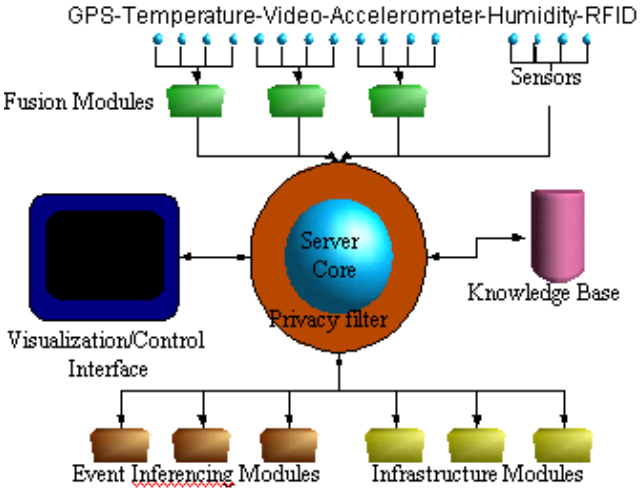

Figure 1: The Networked Sensor Tapestry architecture diagram.

Our work is focused on building an architecture that reduces the complexity of developing, prototyping, and deployment of experimental surveillance systems. Our design choices are predicated on the fact that surveillance infrastructure is commonly resource rich, with AC powered nodes and either wired or reliable-link wireless connectivity. The core of the architecture utilizes concepts familiar in sensor network architectures [2][7] but as computation, power, and bandwidth resources are more abundant in surveillance networks, we can achieve increased flexibility by relaxing these constraints.

3. THE NEST ARCHITECTURE

As illustrated in Figure 1 the complete architecture consists of a centralized server to which multiple clients may connect via standard TCP/IP sockets and share/collect information.

The architecture is comprised of 6 primary components:

Server Core: The server core is designed to provide control of arbitrary clients in a flexible and extendible architecture. The server enables coordinated extraction, dissemination, and processing of data from multiple sources of sensory data. These data sources include rectilinear, thermal, and omni-directional cameras, audio transducers, GPS, RFID, and seismic sensors.

Privacy Filtering and Security: It is imperative that the architecture communication channels be secure to prevent access to intermediate and end results of analysis by unauthorized users. Enforcing a closed network policy can reduce this problem, but this is not always possible, particularly in cases where it is impractical to lay physical connections between sensors and processing modules. The architecture is augmented with secure communication channels via SSL and client authentication.

In addition to data security, the privacy of the monitored subjects may also be of concern. It is often the case that the *behavior* of the subject is relevant, while the *identity* is not, until the behavior

warrants further analysis. Subject privacy is integrated into the architecture and can (if desired) operate as a buffer to the server core, denying access to identity specific information to any or all modules or operators.

Sensors and Interface Hardware: At the most basic level, for sensors to connect to the architecture they must be capable of establishing a TCP/IP connection and constructing XML packets. Most raw sensors therefore will require some sort of interface hardware. More capable hardware interfaces will allow response to sensor control requests to set sampling and other properties of the low level sensors. Section 4 discusses our current sensor interface hardware.

Knowledge Base: The knowledge base contains archived sensor data as well as environmental and expert knowledge. Expert knowledge may come from offline training (ie. training on known vehicle audio/visual signatures for vehicle recognition) and/or modified in real time using unsupervised learning techniques. Raw sensor data, inferenced information, and other sources of data are automatically archived when connected to the NeST subject to user configuration of filter parameters.

Activity Visualization: A key component of the NeST is an interactive 3D interface for control and filtering of surveillance networks consisting of hybrid sensors (Figure 7). This programmable interface allows an operator to specify spatial areas of interest as well as focus on the particular class of information (video, temperature, multiple levels of semantic interpretation, etc).

Virtual Sensors: Virtual sensors are generic software applications operating on real sensor data. These clients can be used for event inferencing, data reduction and correlation, or sensor configuration (sensor calibration and setup).

3.1 Data Flow Illustration

For illustration purposes, before delving into details of individual modules, we follow a sensor client from it's initial connection to the server through the data sharing, privacy filtering, visualization, and archiving process:

Step 1) A temperature sensor board connects to the NeST via an SSL enabled socket. Client authentication is initiated.

Step 2) The client's username and password are checked against a local database of allowable clients. If authorized, the client sends a "message sink" and "message source" packet defining the type of information it can provide and the classes of data it is interested in receiving.

Step 3) The client reads the onboard temperature sensors, packages them in an SENSORDATA.TEMP XML packet, and transmits to the server.

Step 4) The server accepts the packet and routes the data through the privacy buffer. The privacy buffer matches the data to the appropriate privacy filter. (Currently filters are defined for specific types of data, ie temperature, or track data) The data is tagged as either private or non-private with a level of mandate. If the mandate level is high, and the data is marked private, then the packet does not leave the privacy buffer. Otherwise, the data is sent to the packet router.

Step 5) Data that is either not matched to a privacy filter, or is deemed non-private arrives at the packet router. The router checks the client message sink list for clients that have requested to receive temperature data and forwards the message to these clients.

Step 6a) Assuming the visualization application (CoVE) has registered to receive temperature packets, the packet is also forwarded to the CoVE. The CoVE parses the packet and uses embedded location information (supplied by the sensor board) to render a colored sphere in the 3D world reflecting the temperature of the sensor. If no location information is present in the packet, the CoVE can also query the database for a client location entry corresponding to the connected temperature sensor.

Step 6b) Similarly, if the archival client receives the temperature packet, the packet is parsed and the data automatically inserted into a MySQL database with the current time stamp.

The following sections will elaborate on the individual NeST modules.

3.2 Server Core

The primary function of the server core is to authenticate and route data packets to the appropriate clients. The core communication layer handles several fundamental elements of the sensor network including:

1) Sensor and application connectivity via sockets.
2) Client authentication
3) Data packaging via XML
4) Client registration and request handling
5) Communication security
6) Bidirectional communication between the server, sensors, and applications
7) Selective information routing by client request

During the client registration process, each connected client specifies the data it requires from other resources on the network. This is particularly important for application and other software nodes that may require data at different sample rates or resolutions.

3.2.1 Message Classes

Generality and flexibility is maintained between the server and clients by using a layered XML messaging scheme. Each layer defines a class of messages from which nodes in the NeST compose messages. The currently defined message classes are:

SENSOR DATA: Hardware and software (virtual sensor) data packets. Events are also defined as a subclass of virtual sensor data. Examples: Temperature, humidity, acceleration, static images, object information (location, id, bounding box), intruder alerts.

SENSOR CONTROL: A message indicating a change of sensor parameters such as sample rate, spatial and quantization resolution, pan-tilt-zoom, and camera focus.

HEALTH MONITORING: For infrastructure outfitted with health monitoring facilities, these messages provide health measurements such as current power/bandwidth consumption, and remaining battery life.

SYSTEM: System messages consist of critical server events, client heartbeats, synchronization, and authentication data.

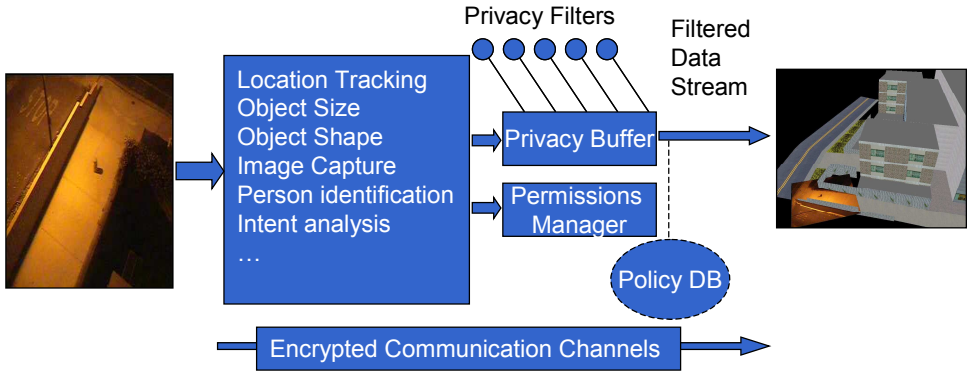

Figure 2: Information extracted from a video stream using a semantic processing client is filtered to remove private content prior to delivery to a visualization application.

APPLICATION: Local application clients may have configurable parameters or may require communication with other connected applications. The application message layer allows these parameters to be set dynamically through the NeST.

New message classes can be easily added to the system as new applications arise.

3.2.2 Message Sinks

Message classes allow a client to easily define the specific type of data it is interested in receiving. Acceptance/rejection filters called *message sinks* can be specified at any level of resolution, from entire classes of data to specific subclasses. A simple message sink such as

```
<MESSAGESINK ALL="SENSORDATA">
```

specifies that the client sending the message sink requests any sensor data from all clients. Filters can also be defined on the data itself, for example, if an application only wants to receive temperature information above a threshold. For example, the slightly more complex filter

```
<MESSAGESINK ALL="SENSORDATA.TEMP
 UNITS==Fahrenheit TEMP>100.0"/>
```

specifies a request for any temperature data where the Fahrenheit temperature is above 100 degrees.

3.3 Virtual Sensors and Configuration Modules

The rectangular boxes in figure 1 are software based virtual sensors and applications. These clients operate on either raw sensor data or processed information from other clients.

Data Fusion and Correlation Clients: To deal with multi-sensory data, a set of data fusion and correlation modules are designed to eliminate redundancies in data and establish relationships between events occurring at potentially disparate spatial and temporal instances. By virtue of the modular architecture, fusion and correlation can occur as a preprocessing step, interfacing sensors to the NeST, or post-transmission to the NeST. These are currently architectural conveniences, as no data fusion is currently performed in the implemented version.

Event Inferencing Clients: Event inferencing clients pull raw data from sensor clients, data fusion clients, and the knowledge base to make decisions about critical events occurring in the observed environment. These clients may be probabilistic in nature, responding to queries regarding the probability of a particular event. Alarms can be programmatically defined for each event inferencing client to alert the operator. Even inferencing clients are typically applications on stand-alone CPUs.

Infrastructure Modules: Infrastructure modules are designed to provide support to the underlying hardware. Such modules include sensor calibration, network health monitoring, and performance tuning. Each infrastructure client may have its own interface to allow operator interaction for calibration tasks that cannot be reliably automated. Like all clients, calibration, health, and other such data are logged in the knowledge base.

Database Interface Client: The DB interface client handles query requests from connected clients as well as parsing of client data packets and subsequent insertion into the database. By default the DB client receives and archives all data flowing through the NeST server core. A table is generated for each client describing the current client state, including its location, sensor properties, and login time. As client data packets arrive to the DB client, they are inserted into a separate table. Each data record references a particular client state record.

3.4 PRIVACY BUFFER

Clients connected to the NeST are unrestricted in the types of information they can extract and share with other clients. In surveillance applications however, much of this information may be private to the monitored individual(s). The identity of the individual can be present in the raw data (such as in an image of a person's face) or in transformed representations of the data (such as the output of a face recognition algorithm). To address this problem, as data enters the server core it must first pass through a filtering mechanism that detects and removes private or identifiable information from the stream. This filtering mechanism shown in figure 2 is called a privacy buffer (PB).

Identity and privacy decisions are made via a set of programmable privacy filter plugins. When registering with the server a filter describes the type of information it is able to process allowing the PB to automatically match incoming data to the appropriate filter.

Filters may not be able to make instantaneous decisions about the privacy content of a data stream. For example, the dynamics of facial expression or the gait of a subject may have embedded identity information that is only detectable after analyzing several

frames of video. Buffering of data, however, is not always possible in time critical applications.

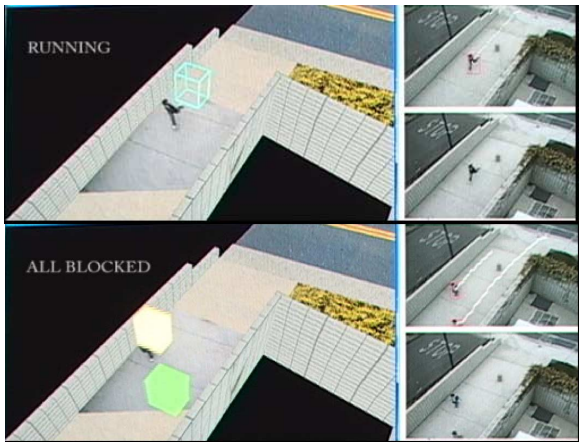

Figure 3: (top) The identity of subjects in video footage embedded in the CoVE is blocked using tracking information. (bottom) Velocity of running person is used to trigger unblocking of the privacy filter.

To allow an application to manage the tradeoff between privacy and real-time performance, the PB has two primary modes of operation: *paranoid* and *apathetic*. In paranoid mode, it is assumed that all data is private, and unless deemed "identity free" is blocked from entering the server core. This mode requires buffering of data until a decision is made. Buffer time is filter and data specific and can be set programmatically.

In apathetic mode, the PB maintains an internal buffer of incoming data, but passes all data through to the server until a filter determines that private information is present. This is necessary for time critical applications where performance is critical and privacy is a secondary concern. By default there is no delay in the information pass through in this mode. However, an application can adjust the transmission delay from the buffer to tune the tradeoff between performance and privacy maintenance.

We have currently implemented a primitive velocity based privacy filter for tracking data acquired from video images. Instead of completely blocking the data, it augments the track packet with a private/non-private attribute to allow a visualization application to render the data appropriately. This feature is shown in Figure 3 and demonstrates the ability to perform behavioral filtering of data, for example, only allowing the identity of "suspicious characters" to be scrutinized.

3.4.1 Privacy Grammar

The current filter specification is relatively ad-hoc, requiring low-level manual specification of filter properties and matching of data. The definition of privacy policies, however, is very complex and arguably an unsolved problem. Furthermore, there is no quantitative definition for private data, as the concept of privacy is largely dependent on the perception of the individual whose privacy is violated. Greater flexibility is therefore necessary for experimental specification of what kinds of data or parts of data are private. We introduce the notion of a *privacy grammar* for this purpose. With a privacy grammar we propose to combine low level feature extraction operators with higher level semantic interpretations of sensor modes to enable flexible construction of privacy filters.

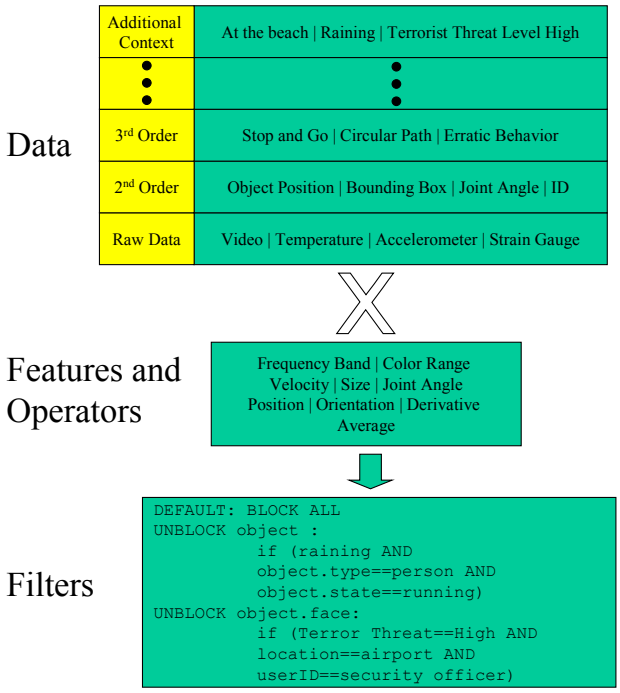

Figure 4: Privacy filter generation using data hierarchy and generic data operators/feature extractors.

As shown in Figure 4, data is classified into multiple levels of abstraction. Raw sensor data is at the lowest level with derived data layered above. An additional "context" layer is added to describe data that may be human-operator supplied, such as the current terrorist threat level or semantic interpretation of the location under surveillance. These layers all provide components that may be used with a privacy grammar to specify individual filters.

Using the privacy grammar, new filters can be defined and matched to different goals and focuses of end users. For example randomization filters can be defined where 1 out of 100 persons are randomly unblocked by the system or logged into the database. Or in the case of anonymizing filters, objects can be shown but key identifiable information may be blocked such as a person's face, a vehicle license plate, or a characteristic dent on a car. The underlying privacy filtering mechanism can also serve as a generic data-filtering framework for focusing the delivery of data to the end user.

Access control is a fundamental issue for the privacy control system. Dependent on a user's privileges, he/she should be given access to different components of the data. Standard methods can be used such as applying policies by user ID's and groups. By breaking the system down into low level components that can be assembled into higher-level filters, we can easily construct policies based on user and group ID's. For example, a policy definition such as

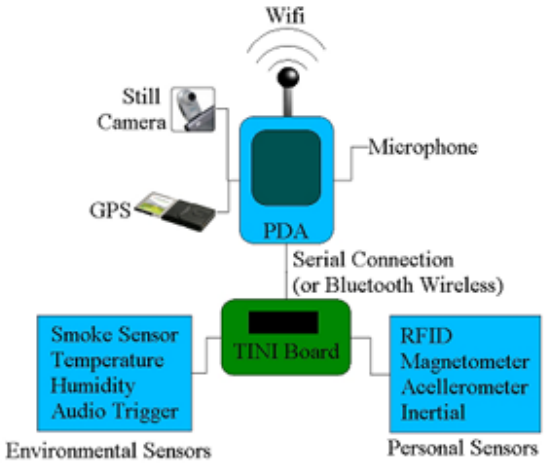

Figure 5: Mobile sensor interface device.

```
UNBLOCK object IF (groupID==security
personnel AND object.type==human AND
(Profile(object)==suspicious))
```

specifies a filter that allows a particular object to be seen if the user requesting the ID is an authenticated member of a security personnel team, the object is a human, and the human is deemed by the system to belong to a suspicious profile class (such as having an expired passport).

4. SENSOR INTERFACE HW/SW

We are in development of a general purpose, extendable, modular interface device for collection and transmission of environmental sensor data using the NeST (Figures 5 and 6). New sensors can be seamlessly added to the device and their capabilities recognized and data shared with the NeST. The interface hardware is responsible for data acquisition from one or more attached sensors and relaying this data to the NeST via a TCP/IP connection. The hardware must also respond to requests from other applications or the server itself for changes in sensor parameters.

The current implementation uses a TinyOS-based microcontroller by Dallas Semiconductor (DSTINIm400) to interface between a 1-wire sensor network and a (currently) wired Ethernet connection to the NeST or may connect wirelessly through an 802.11b connection on an HP Ipaq PDA . The 1-wire sensor network facilitates the addition of new sensors to the NeST. A variety of 1-wire devices already exist including temperature sensors, data loggers, and timers. Analog sensors that are not 1-wire compatible may be interfaced using a 1-wire A/D converter as we have done with the Memsic accelerometer.

The NeST connectivity library has been ported to the microcontroller's scaled-down Java implementation and C++ for the Ipaq. A preliminary sensor control API has also been developed to respond to SENSOR CONTROL messages on the PDA and interface hardware. We have interfaced a variety of sensors including temperature, battery monitors, GPS devices and accelerometers and are currently working to include RFID, smoke sensors, humidity sensors and inertial trackers.

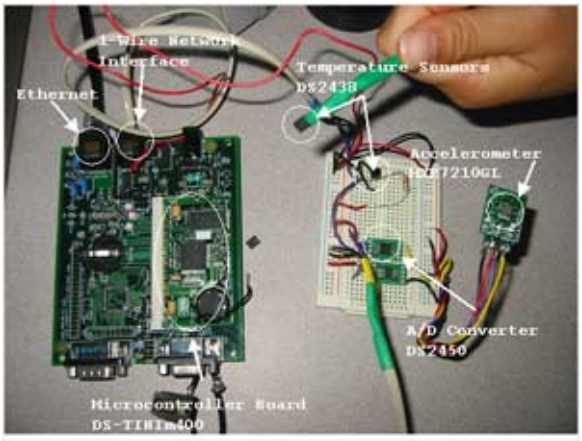

Figure 6: Sensor interface hardware with attached temperature sensors and accelerometer.

Though the PDA currently supplies wireless access to the NeST, the primary intended function is to provide an interface to the user. This interface can be critical to first responders or other users who may make use of context information collected by the device. In many cases purely passive acquisition of data can be useful (no interface needed), for example by adding a data collection device to a vehicle, or as an additional piece of equipment to a first-responder's tool chest. Near-term future work will therefore involve equipping the core sensor module with wireless connectivity directly. This will also allow significant reduction in the size of the device.

5. IMPLEMENTATION AND RESULTS

The NeST server is implemented in Java and currently runs on a dual Intel Xeon 2.8 GHz Linux machine with a MySQL 4.0 database.

Due to the centralized nature of the NeST it is currently not possible to pass several high-resolution video streams through the server core. Streaming video data is therefore accessed on an individual application basis through either locally connected cameras or remote Axis video servers.

Several surveillance related applications have already been built on or connected to the NeST including (1) a vehicular tracking and classification application utilizing strain gauge sensor data for triggering of video acquisition and analysis, (2) a sentry application detecting unauthorized entry to an arbitrary space using computer vision techniques, (3) an intelligent room (MICASA) [1], (4) an indoor temperature sensor array, (5) a custom designed PDA-based mobile sensor device with GPS, temperature, acceleration, and interactive communication abilities.

5.1 Connection to the Context Visualization Environment (CoVE)

Video images from spatially disjoint cameras are difficult to interpret, especially as the camera parameters such as direction and field of view are altered. By registering and merging sensor data into a fully 3D environment, much greater insight can be gained and the operator can focus on critical information, rather

than wasting precious resources resolving image disparity [3][4]. The Context Visualization Environment (CoVE) is designed to give the operator programmable control over the displayed data while "critical events" inferenced in backend NeST processes can pre-empt the operator's focus and generate alerts and warnings of impending dangers. The operator is also given interactive control over sensors such as the pan-tilt-zoom parameters of surveillance cameras, rate of sampling, and sampling resolution if such overrides are deemed relevant for a given task.

Figure 7 shows the application connected to the NeST along with the database client, an indoor and outdoor tracker [1], and an interface board with 3 temperature sensors. The outdoor tracker processes a live video stream and detects the time varying location of humans in the scene. The position and bounding box information is shared with the NeST. Temperature is visualized as a colored sphere at the sensed location. Though not required, all client applications are currently running on separate machines.

6. CONCLUSION AND FUTURE WORK

The NeST has proven to be an extremely flexible architecture for building surveillance applications. The simple connection and communication model has enabled several prototype applications to be developed in a very small amount of time. The layered XML messaging scheme allows new applications with unforeseen requirements to be quickly developed without affecting existing communication models. The novel privacy buffering mechanism addresses an increasingly important problem in ubiquitous surveillance networks. The modularity of the NeST is reflected in the modular sensor interface hardware to allow seamless insertion of new sensors to the infrastructure.

A fundamental component missing from the NeST is robust client and sensor synchronization. Currently a rudimentary latency calculation is performed at client authentication to generate an offset between the server clock and local client clock, however there are several compelling applications that rely on synchronized data. An example is a distributed camera handover application where tracked objects leave one camera and enter another. If the track data is not synchronized, objects may be easily confused by the handover application.

Provisions for network health monitoring, data fusion, and event extraction exist in the messaging scheme and module abstraction (base classes) but the full details have not been specified or implemented. Particularly, a scheme akin to the proposed privacy grammar can be implemented for specification of notable events that should be flagged and notify a user or other application.

Though the privacy grammar holds great promise for specification of privacy policies and filters, the privacy buffer is currently in its infancy. Currently there are only 2 classes of private data: private and non-private which is clearly too simple for general purpose applications. Privacy definitions contingent upon access rights are a first step, but a generally more rigorous approach to policy definition will be explored in the future.

The core pieces of the NeST have been specified and implemented but the development and testing is ongoing. Analysis of the performance limitations of the NeST will also be conducted. We intend to leverage these initial results to define subsequent revision of the architecture and further specify module abstractions relevant to surveillance applications.

7. ACKNOWLEDGMENTS

This research was supported in part by a UC Discovery Program Grant (CoRE) and by a US DoD research contract.

The authors would like to thank their colleagues at the UCSD Computer Vision and Robotics Laboratory especially for assistance in the development of the experimental test-bed.

8. REFERENCES

[1] Mohan M. Trivedi, Kohsia S. Huang, Ivana Mikic, "Dynamic Context Capture and Distributed Video Arrays for Intelligent Spaces," Accepted (January 2004) for publication in *IEEE Trans. on Systems, Man, and Cybernetics, special issue on Ambient Intelligence*.

[2] Lewis Girod, Jeremy Elson, Alberto Cerpa, Thanos Stathopoulos, Nithya Ramanathan, Deborah Estrin, "EmStar: a Software Environment for Developing and Deploying Wireless Sensor Networks", *Proceeding of the 2004 USENIX Technical Conference*.

[3] B. Hall, M. M. Trivedi, "A Novel Interactivity Environment for Integrated Intelligent Transprotation and Telematic Systems," *5th International IEEE Conference on Intelligent Transportation Systems, Singapore,* pp. 396-401, September 3-6, 2002.

[4] U. Neumann, S. You, J. Hu, B. Jiang, and J. W. Lee, "Augmented Virtual Environments (AVE): Dynamic Fusion of Imagery and 3D Models",
IEEE Virtual Reality 2003, pp. 61-67, Los Angeles California, March 2003.

[5] Kanade, T., R. Collins, A. Lipton, P. Burt, & Wixson, L. (1998). Advances in cooperative multi-sensor video surveillance. Proc. of DARPA Image Understanding Workshop, 1, 3-24.

[6] K. Huang, M. M. Trivedi, "Distributed Video Arrays for Tracking, Human Identification, and Activity Analysis," *Proceedings of the 4th IEEE International Conference on Multimedia and Expo*, Baltimore, MD, pp. 9-12, July 2003.

[7] Henk Muller and Cliff Randell. "An Event-Driven Sensor Architecture for Low Power Wearables". In: *ICSE 2000, Workshop on Software Engineering for Wearable and Pervasive Computing*, pages 39--41. ACM/IEEE, June 2000

[8] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister. "System Architecture Directions for Networked Sensors" *In Proceedings of the 9th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 93-104, Cambridge, MA, November 2000.

[9] J. Boyd, E. Hunter, P. Kelly, L. Tai, C. Phillips and R. Jain, MPI-Video Infrastructure for Dynamic Environments", *IEEE International Conference on Multimedia Systems 98*, Austin TX, June 1998.

[10] W. J. Kaiser, G.J. Pottie, M. Srivastava, G.S. Sukhatme, J. Villasenor, and D. Estrin "Networked Infomechanical Systems (NIMS) for Ambient Intelligence", *in CENS Technical Report #31*, December 5 2003.

[11] V.A. Kottapalli, A.S. Kiremidjian, J.P. Lynch, E. Carryer, T.W. Kenny, K.H. Law, Y. Lei, "Two-tiered wireless sensor network architecture for structural health monitoring", *SPIE's 10th Annual International Symposium on Smart Structures and Materials*, San Diego, CA, USA, March 2-6, 2003.

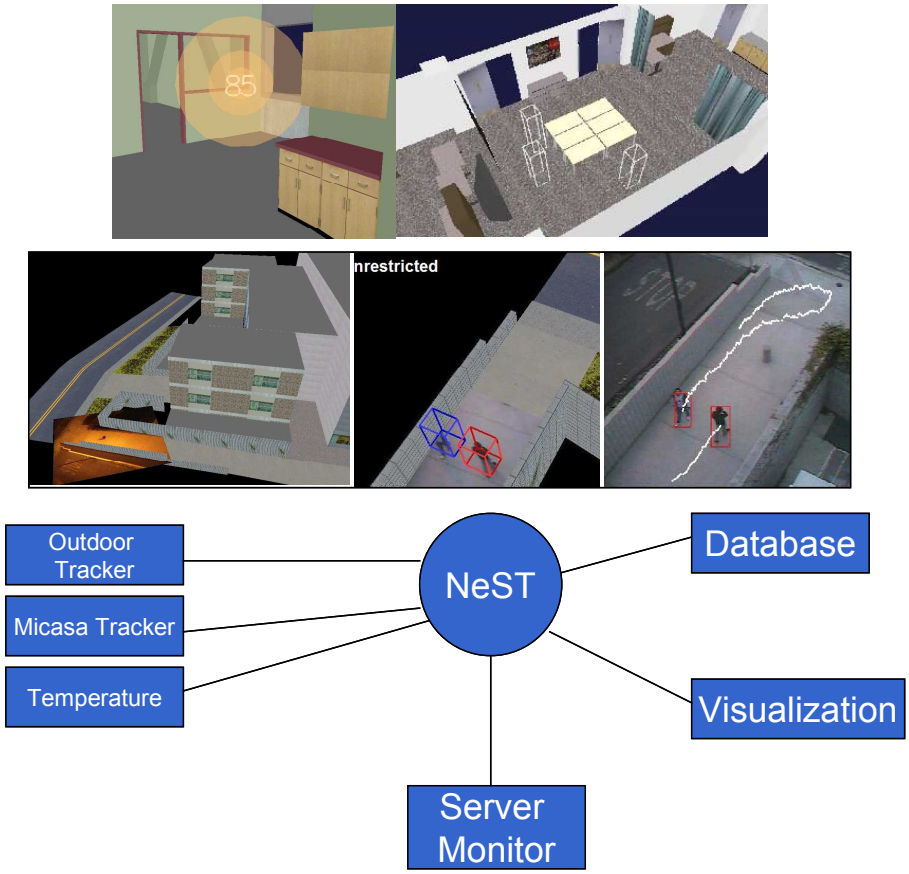

Figure 7: CoVE visualization environment siphoning data from trackers and temperature sensors via the NeST.