Copyright Notice:

Reuse Rights and Reprint Permissions: Educational or personal use of this material is permitted without fee, provided such use: 1) is not made for profit; 2) includes this notice and a full citation to the original work on the first page of the copy; and 3) does not imply IEEE endorsement of any third-party products or services. Authors and their companies are permitted to post their IEEE-copyrighted material on their own Web servers without permission, provided that the IEEE copyright notice and a full citation to the original work appear on the first screen of the posted copy.
Permission to reprint/republish this material for commercial, advertising, or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to the IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway, NJ 08854-4141 or pubs-permissions@ieee.org. Copyright © 2009 IEEE. All rights reserved. Abstracting and Library Use: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee indicated in the code at the bottom of the first page is paid through the Copyright Clearance Center, 22 Rosewood Drive, Danvers, MA 01923.

Full Citation:

Middleware and Metrology for the Pervasive Future. Antoine Fillinger, Imad Hamchi, Stéphane Degré, Lukas Diduch, Travis Rose, Jonathan Fiscus and Vincent Stanford. IEEE Pervasive Computing Mobile and Ubiquitous Systems. Vol. 8, num. 3, page 74-83, July-September **2009**.

Middleware and Metrology for the Pervasive Future

The National Institute of Standards and Technology has developed data and metrology tools for the research community, including common middleware for distributed sensor data acquisition and processing.

Antoine Fillinger, Imad Hamchi, Stéphane Degré, Lukas L. Diduch, Travis Rose, Jonathan Fiscus, and Vincent Stanford
National Institute of Standards and Technology

Mark Weiser of the Xerox Palo Alto Research Center (PARC) envisioned a future that would transform computers from desktop workstations to unobtrusive, ubiquitous data appliances of everyday life. *Smart spaces* constitute a second stage of this transformation, in which user and context-sensitive interfaces sense, recognize, respond to, and assist individual users as they live and work. The processing and fusion of distributed multimodal sensors—recognizing who people are, what they say, and even their gestures and activities as well as context-sensitive responses—is the subject of widespread research. These interfaces and services are of unprecedented complexity and are being developed across a community of advanced-technology laboratories.

To deal with this complexity, the US National Institute of Standards and Technology (NIST) has provided data and metrology tools for the past 20 years to aid the research community. These include the NIST Data Flow System (NDFS)—common middleware for distributed sensor data acquisition and processing. This middleware supports interoperability for algorithms and hardware, collaborative development, and performance measurement.

International Collaboration

NIST's mission is to "promote US innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life" (www.nist.gov/public_affairs/nist_mission.htm). In the global economy, this includes mutually beneficial international collaboration.

Since the mid 1980s, NIST programs in spoken-language and multimodal systems have focused on creating standard research corpora, "ground truth" metadata, interoperability, and metrology tools for measuring recognition performance against standard data sets. NIST has provided the community with

- data sets to train and test classifiers that recognize words, speakers, events, and objects;
- metrics such as dynamic alignment for word error rates, speaker identification, and biometrics; and
- sensor-net middleware for data collection and synchronization of research corpora.

NIST programs support research at leading laboratories in the US and worldwide under programs such as Video Analysis and Content Extraction (VACE), Computers in the Human Interaction Loop (CHIL), and Augmented Multiparty Interaction (AMI). For example, our Rich Transcription 2007 Meeting Recognition Evaluation participants used conference-meet-

Published by the IEEE CS ■ 1536-1268/09/$25.00 © 2009 IEEE

ing, lecture-meeting, and lecture-coffee-break corpora. (The work of many of our collaborators is available elsewhere.[1,2]) These programs used data collected with NDFS middleware and also used the NIST Mark-III microphone arrays. Laboratories collaborating with these programs included:

- Athens Information Technology,
- Computers in the Human Interaction Loop
- Carnegie Mellon University,
- Edinburgh University,
- Evaluations and Language Resources Distribution Agency,
- IBM Research Division,
- International Computer Science Institute,
- Institute for Infocomm Research,
- Nanyang Technological University,
- SRI International,
- Center for Scientific and Technological Research,
- Karlsruhe University,
- Linguistic Data Consortium,
- Laboratoire Informatique d'Avignon,
- Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur,
- University of Sheffield,
- Netherlands Organization for Applied Scientific Research TNO,
- Universitat Politècnica de Catalunya, and
- Virginia Tech.

Tasks included speech to text, close-talk microphone transcription, meeting diarization of who spoke when, and speaker-attributed speech to text. Participants also investigated speech activity detection.

This research required a new generation of data collection facilities, so several CHIL participants (Athens Information Technology, IBM Research Division, Center for Scientific and Technical Research, Karlsruhe University, and Universitat Politècnica de Catalunya) constructed laboratories containing 64-channel NIST Mark-III microphone arrays, T-shaped four-channel arrays, table-top microphones, wireless CTMs, and video cameras. They collected interactive seminars with interruptions and coffee breaks, and evaluated various recognition system tasks.

The Classification of Events, Activities, and Relationships (Clear) program began in 2005 in collaboration between the US-funded VACE and the European-Commission-funded CHIL programs. Fifteen organizations participated in the 2006 Clear evaluation. In 2007 the AMI program joined the consortium. Clear provides a framework for evaluating algorithm research and creating metrics, tasks, and annotated multimodal corpora for perceptual technology. In 2007, there were 24 evaluation tasks, supported by 19 organizations. The tasks included 3D single- and multiple-person tracking; 2D multiple-person tracking; 2D face tracking; person identification; head pose estimation; and detection of acoustic events such as door knocks, footsteps, chair motions, paper work, phone rings, and other nonspeech sounds.

NIST Middleware Development

NIST held the first DARPA/NIST Smart Spaces Workshop in 1998 to explore, with the research community, issues about creating smart spaces using distributed pervasive devices,[3] sensor-rich audiovisual user-sensitive interfaces,[4,5] complex interoperability requirements,[6] and multimodal recognition technologies.[7]

The increasing complexity of interfaces using speech recognition and context-sensitive multimodal responses, such as those proposed by Vincent Stanford and his colleagues,[8] required distributed data transport. In response, the NIST Information Access Division began the NDFS project in 1998, which became operational in 1999. This system drew concepts from an earlier large-grained data-flow system called the Graph Analysis and Design Technique (GADT), for naval array signal processing.[9] GADT provided runtime semantics for graphs in a form similar to the Structured Analysis and Design Technique (SADT) for data-flow-based requirements modeling that Doug Ross and his colleagues developed in the late 1960s and early 1970s.[10] The early work on data-flow systems by Ross and others has inspired several projects in recent years (see the "Evolution in Data-Flow Systems" sidebar).

The NIST Information Access Division began the NIST Data Flow System (NDFS) project in 1998, which became operational in 1999.

We designed NDFS for localized high-bandwidth, multisensor data streams, which consisted of hundreds of microphones and multiple video channels acquired on networked sensors and PCs. It used broadcast networking for application server discovery, so it remained limited to local-area networks, typically with point-to-point gigabit connections between multicore PCs, and wireless connections to pervasive devices. Because NDFS was designed as an open middleware layer for distributed sensors and computations, and served a collaborative research community, we developed multiple-language bindings for it. These included C++, GNU Octave, and Java. GNU Octave offers extensive high-level libraries for matrix and sensor signal-processing-algorithm development. NDFS was also easily adaptable for running with Matlab.

Another goal was access to the widest variety of sensors on all major operating systems. NDFS ran transparently across networks and, more efficiently, through shared memory on single

Evolution in Data-Flow Systems

In recent years, researchers have developed numerous distributed-processing systems, many using data-flow architectures, with various specialized architectural features. Many of these systems significantly overlap with the US National Institute of Standards and Technology's NIST Data Flow System (NDFS) architecture.

Grid computing created a virtual batch of scientific supercomputers from geographically distributed centers.

Several projects, such as TelegraphCQ, Aurora and Borealis, Pipes (Public Infrastructure for Processing and Exploring Streams), and WaveScope, focused on continuous-query (CQ) of SQL databases over data streams, with events corresponding to SQL table rows. (WaveScope also supports parallel signal processing.) This suggests applicability to a different domain than the tightly coupled gigabit bandwidth of NDFS-based smart spaces. For example, a CQ system can process a stream from a NIST Mark-III array by inserting 44,100 64-channel rows per second into an SQL relational database management system (RDBMS), but the overhead would place prohibitive demands on low-cost hardware. Many CQ systems use Java, rather than the portable C++ libraries used in NDFS, to achieve portability. Unfortunately, although Java performance is improving, Java still reduces performance for any given hardware platform.

Hourglass and Global Sensor Networks are both middleware for data collection networks of heterogeneous Internet sensors.

Grid Computing

The Grid Computing Initiative, which began in 1997,[1] provides resource location and allocation, communications, a unified resource information service, an authentication interface, process creation, and data access. It also implements a unified access mechanism called Metacomputing Directory Services (MDS) using the Lightweight Directory Access Protocol (LDAP). Thus, it offers large-scale and geographic distribution and is often used to implement batch parallelism. But it also supports distributed-processing systems, including Parallel Virtual Machine (PVM) and the Message Passing Interface (MPI).

As is often the case with highly successful projects, the grid concept is spreading in many directions, with additions such as service-oriented architecture (SOA), Web services, and even virtual organizations. This wide array of extensions precludes us from providing an exhaustive survey of grid technologies in this article, but it clearly indicates the successful implementation and wide dissemination of grid-computing technology.

TelegraphCQ

The Telegraph Dataflow System was an early project written in Java beginning in 2000, and grew out of earlier adaptive relational-query work. TelegraphCQ, first described in 2002, supports CQ processing over streams using an adaptation of the PostgreSQL RDBMS.[2] It processes streams in unpredictable environments by using query operators and provides load balancing, fault tolerance, and adaptability, enabling better performance compared with static-query plans.

Aurora and Borealis

Aurora began in 2001 and led to the development of the standards-based StreamSQL, which supports continuous and time-windowed queries of transaction streams. Borealis—a second-generation distributed stream-processing engine that can handle high-volume, event-based transaction streams such as stock exchanges—superceded Aurora in 2005.[3] It includes a distributed catalog, nodes, client applications, and data sources. Stream flows connect the nodes, which collaboratively compute the queries and thus embody this system's data-flow aspect. Borealis provides automatic coordination of queries, fault tolerance, load monitoring and balancing, and revision processing for erroneous input. It also has a GUI for creating queries, organizing data streams, and visualizing Borealis networks. Borealis has been built and tested on various Linux distributions.

Pipes

Pipes was developed in 2003. It provides a publish-subscribe architecture for processing and exploring streams, and it offers

machines. It duplicated data flows when necessary and used shared memory when possible. The data transport protocols were lightweight, and it acquired about 4 Gbytes of reference data per minute on low-cost commodity hardware in our NIST Meeting Room Data Collection Laboratory.

The limited processors and bandwidth of the late 1990s forced us to develop NDFS-I, based on C, with efficiency as a primary requirement. The result was platform-dependent and even brittle, but we were able to collect more than 20 Tbytes of multimodal meeting data for the research community.

NDFS-II successfully addresses these limitations. It still specializes in sensor-rich, localized smart spaces, but it's more flexible and fault tolerant. It supports all major PC operating systems, including many versions of Windows, OS X, and Linux, so sensors and hardware interfaces from most vendors are accessible. We chose C++ for the NDFS-II source code because it supports high data rates on modest hardware, yet functionalities to express, implement, and run CQs over data streams.[4] It was implemented in the XXL Java library for query-processing algorithms. Pipes allows the composition of query graphs with runtime resource sharing. The Pipes group has investigated scheduling approaches such as operator threading and graph threading, and it has developed a hybrid approach that allows concurrency with reduced overhead.

WaveScope

WaveScope appeared in 2006 as a data management and continuous sensor data system that integrates RDBMS CQ and signal-processing operations into a single system. The major components include a new language called WaveScript for signal processing, an execution engine for multicore PCs, and a distributed-execution engine. Lewis Girod and his colleagues make a case for this combination of capabilities.[5] According to the developers' Web site, the project is still at an early stage. Developers can compile the signal-processing operators in WaveScript programs into data-flow graphs at a finer-grained level than the large-grained process-level nodes supported by NDFS. Hence, with many processor cores now emerging, WaveScope could be especially well-suited for highly parallel systems.

Hourglass

This system, developed in 2004, is a data collection network for naming, discovering, routing, and aggregating data from geographically diverse sensor networks using a publish-subscribe paradigm.[6] Hourglass is similar to the NDFS architecture in several ways and the systems use a similar discovery mechanism. Both maintain a buffered-flow architecture to permit intermittent connectivity, separate control and data components, discovery of resources such as sensors, and quality of service (QoS). However, Hourglass supports additional Internet services for wider distribution, albeit with higher overhead.

Global Sensor Networks

GSN, developed in 2006, is a Java-based platform for deploying sensor network technologies that uses a set of abstraction layers comprising virtual sensors in a container-based architecture accessed via standard Internet or Web services.[7] According to this project, the lack of standardization and frequent arrival of novel sensors make portable-application development difficult. GSN provides distributed querying, filtering, and combining sensor data. Each GSN virtual sensor corresponds to a database table, and each sensor reading corresponds to a tuple.

REFERENCES

1. I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *Int'l J. Supercomputing Applications*, vol. 11, no. 2, 1997, pp. 115–128.

2. S. Chandrasekaran et al., "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," *Proc. 1st Biennial Conf. Innovative Data Systems Research* (CIDR 03), VLDB Foundation, 2003; www-db.cs.wisc.edu/cidr/cidr2003/program/p24.pdf.

3. D.J. Abadi et al., "The Design of the Borealis Stream Processing Engine," *Proc. 2nd Biennial Conf. Innovative Data Systems Research* (CIDR 05), VLDB Foundation, 2005; www-db.cs.wisc.edu/cidr/cidr2005/papers/P23.pdf.

4. M. Cammert et al., *PIPES: A Multi-threaded Publish-Subscribe Architecture for Continuous Queries over Streaming Data Sources*, tech. report 32, Dept. of Mathematics and Computer Science, Univ. of Marburg, 2003.

5. L. Girod et al., "The Case for a Signal-Oriented Data Stream Management System," *Proc. 3rd Biennial Conf. Innovative Data Systems Research* (CIDR 07), VLDB Foundation, 2007, pp. 397–406; www-db.cs.wisc.edu/cidr/cidr2007/papers/cidr07p45.pdf.

6. J. Shneidman et al., *Hourglass: An Infrastructure for Connecting Sensor Networks and Applications*, tech. report TR-21-04, School of Eng. and Applied Sciences, Harvard Univ., 2004.

7. K. Aberer, M. Hauswirth, and A. Salehi, *The Global Sensor Networks Middleware for Efficient and Flexible Deployment and Interconnection of Sensor Networks*, tech. report LSIR-2006-006, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL); http://lsirpeople.epfl.ch/salehi/papers/LSIR-REPORT-2006-006.pdf.

presents an object-oriented programming interface for all major operating systems. NDFS-II offers interoperability for algorithms and hardware, collaborative development, and performance measurement. (The "Infrastructure for Collaborative Multimodal-Sensor Research" and "Event Recognition in Sensor-Based Smart Environments" sidebars discuss two scenarios in which this middleware has been used.)

NDFS-II lets us capture about 250 Gbytes per hour from sensors in our NIST Meeting Room Data Collection Laboratory for our multimodal corpora (see Figure 1). We synchronize the audio and video channels to within a video frame by estimating time tag trend lines, which we use to adjust offsets, smooth jitter, and compensate for data collection clock rates.

The NIST Data Flow System II

The NDFS-II architecture has the following components:

- *Data-flow servers* discover the peer

Infrastructure for Collaborative Multimodal-Sensor Research

Kevin Donohue and Jens Hannemann
University of Kentucky Center for Visualization and Virtual Environments

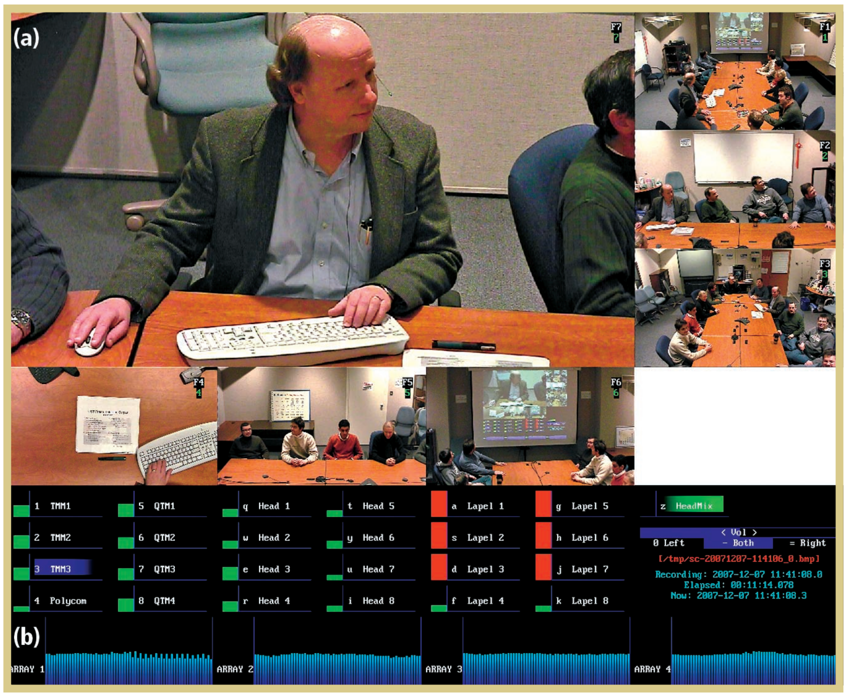
The University of Kentucky Center for Visualization and Virtual Environments is developing a multimodal distributed-sensor-system testbed for research in scene understanding and smart spaces. The US National Science Foundation funds this ambient-virtual-assistant (AVA) project. The testbed's main goal is to establish an efficient infrastructure for researchers to implement and test algorithms for processing the complex, massive data sets that the multimodal sensor system generates.

Our system currently consists of 23 cameras connected to host computers, 40 microphones, and 24 speakers connected to a single host computer running the Jack audio connection kit, several projectors, displays, and RFID readers. These sensors are distributed over three offices and a hallway in our laboratory. We selected the US National Institute of Standards and Technology's NIST Data Flow System II (NDFS-II) to transport streaming data to an 80-node Linux-based cluster for general processing. To explore the possibilities of the NDFS-II API, we developed a Jack front end and back end, which enable the connection of NDFS-II data flows for input and output to Linux-supported sound cards.

Current cluster-scheduling and load-balancing systems, such as Sun's Grid Engine, aren't suited for processing real-time streaming data. To overcome this, the AVA project has designed and implemented an API that lets programmers route streaming data to one or more nodes in a cluster. The multicasting used by NDFS-II, running on the eight-way symmetric multiprocessor (SMP) cluster nodes, improves this routing's efficiency. For each data stream, the testbed creates a tree of processors. These processor trees can run in separate threads, if necessary, fully exploiting the underlying parallelism potential. Data streams from different modes and sensor sets with timing dependencies achieve synchronization via the network using the Precision Time Protocol (PTP).

To enhance the ability to propose and implement experiments on the testbed, we developed an interactive system that enables direct access to a library of basic processing operations and data stream control via the Python scripting language. Researchers can therefore control the routing and scheduling from a remote machine using XML-RPC.

NDFS-II has provided us with a robust, efficient infrastructure for reliably transporting real-time data over local, high-bandwidth networks, thereby enabling flexible processing of data from the multimodal and distributed systems in our testbed.



```
[/tmp/sc-20071207-114106_0.bmp]
Recording: 2007-12-07 11:41:08.0
Elapsed: 00:11:14.078
Now: 2007-12-07 11:41:08.3
```

Figure 1. The US National Institute of Standards and Technology (NIST) Meeting Room Data Collection Laboratory in operation, using the NIST Data Flow System (NDFS) for data transport, synchronization, and storage, as seen at the review station: (a) one pan-tilt high-definition camera (upper left) can follow a presenter, while the other six high-definition cameras show whiteboards, written materials, projector displays, and the other meeting participants; (b) the system displays input levels for 24 individual microphones and four high-resolution NIST Mark-III 64-channel microphone arrays.

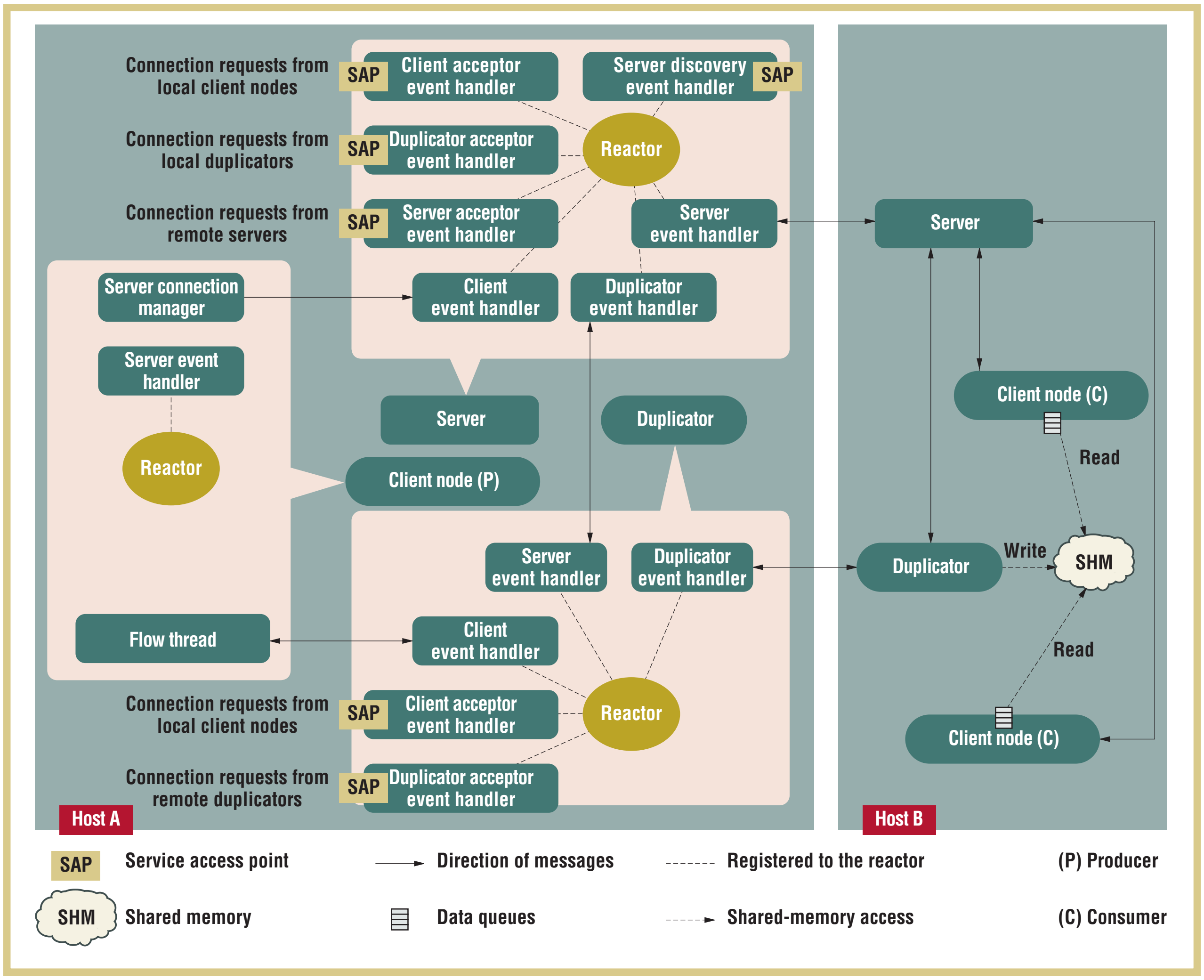

Connection requests from local client nodes
SAP
Client acceptor event handler
Server discovery event handler
SAP
Connection requests from local duplicators
SAP
Duplicator acceptor event handler
Reactor
Connection requests from remote servers
SAP
Server acceptor event handler
Server event handler
Server
Server connection manager
Client event handler
Duplicator event handler
Server event handler
Reactor
Server
Duplicator
Client node (P)
Client node (C)
Read
Server event handler
Duplicator event handler
Duplicator
Write
SHM
Flow thread
Client event handler
Read
Connection requests from local client nodes
SAP
Client acceptor event handler
Reactor
Client node (C)
Connection requests from remote duplicators
SAP
Duplicator acceptor event handler
Host A
Host B
SAP Service access point
Direction of messages
Registered to the reactor
(P) Producer
SHM Shared memory
Data queues
Shared-memory access
(C) Consumer

Figure 2. The major NIST Data Flow System II (NDFS-II) components and their interprocess communication. The data-flow server (top center) manages control messages between clients, duplicators, and hosts. The duplicators (bottom center) each transport one data flow and duplicate the data stream as necessary for multiple local clients. The client node (left) uses a shared library, which has one thread to process messages from the server and one thread per flow for data transport.

application servers and maintain lists of active servers, clients, and flows by using XML-based control messages.

- *Clients* read and write flows. They're connected to the local server for control, and to duplicators for binary data flows.
- *Duplicators* transport data flows, which are controlled by local servers and connected to remote duplicators and local clients. There is one duplicator per flow where that flow is produced or consumed.
- The *control center* displays, controls, and monitors application graphs.

Figure 2 shows the control and data transport connections. NDFS-II relies on an open source, cross-platform communication library, the Adaptive Communication Environment (ACE; www.cse.wustl.edu/~schmidt/ACE.html), for multiplatform concurrent communication and uses the open source Qt toolkit for portable GUIs. ACE provides C++ wrappers and framework components for communication design patterns on major operating systems. NDFS-II uses the ACE reactive model for connection requests, message processing, and data transport. Servers, duplicators, and clients use the reactor framework to automate detection and demultiplexing of events and dispatch them to handlers. The acceptor-connector framework establishes connections and initializes services.

Implementing a server using the reactive model is less error-prone than using multiple threads and semaphores, because NDFS-II a priori avoids interlocking

Event Recognition in Sensor-Based Smart Environments

Albert Ali Salah
Centrum, Wiskunde & Informatica, Amsterdam

The eNTERFACE Workshop (www.cmpe.boun.edu.tr/enterface07) is a one-month gathering where researchers can collaborate on projects involving human-computer interaction. In the 2007 workshop, organized at Bogaziçi University in Istanbul, we implemented a system to identify and track people in a smart room using multimodal information.[1]

We employed low-cost cameras and microphones, which had limited individual accuracy but could produce accurate descriptions working together. We also wanted opportunistic sensing, which uses information sources such as the color of clothing, not ordinarily considered as biometric identifiers but useful in the application context.

The system used facial images, captured as people walked into the room, to recognize them in images from the other cameras. Four ceiling cameras received each person's ID, and feature-based identification clients constructed color-based feature models of each person on the fly because face recognition from the low-quality ceiling cameras was difficult. A locally constructed, 14-element microphone array aided localization and identification based on acoustic information. We wrote a recognition module for simple gestures in visual input, and other modules for motion detection and foreground-background extraction. A visualization client displayed a map of the room, with icons depicting people and identification tags.

We used the US National Institute of Standards and Technology's (NIST) Data Flow System II (NDFS-II) as the middleware to connect the many components needed for the project, and its interface was intuitive and easy to use. Its cross-platform capability was essential because several hosts running both Linux and Windows drove our sensors. On the other hand, we discovered that communication between clients could become a major bottleneck. Also, different clients' frame rates needed adjustment, depending on the resources at hand. In general, the modules written before the workshop by different groups operated under different assumptions, and integrating them quickly was a significant challenge. Nevertheless, using the NDFS-II middleware at this workshop was a stimulating, thoroughly satisfying experience for all the involved parties.

REFERENCE

1. R. Morros et al., "Event Recognition for Meaningful Human-Computer Interaction in a Smart Environment," *Proc. 3rd eNTERFACE Workshop*, Bogaziçi Univ., 2007, pp. 71–86.

between threads and shared variables. Also, the duplicators support a hybrid push/pull data transport model and can synchronize flows by block count for general parallel distributed processing.

The NDFS-II Network

One data-flow server runs on each participating host. Collectively, the sibling servers maintain a crossbar connection to the other servers in the application. The discovery process lets hosts join the NDFS-II network at any time. When launched, a server opens its access points for incoming connections and broadcasts a message containing its application name. In response, each sibling server sends its current application description. Thus, each server builds a full description of the running application as it joins the network.

This peer-to-peer synchronization protocol avoids a single point of failure because all servers maintain a complete and current application description. So, each server manages the application components running on its host by receiving local client requests directly and remote ones indirectly via remote servers. Servers then process requests, forward them as necessary, and update other servers with any changes.

Privileged-client APIs, used in the control center but open to other clients as well, let developers create and control distributed NDFS-II application graphs. These privileged clients use the API's control methods to send requests to their server—for example, launching or stopping a specific client, requesting a description of the running NDFS application, or making changes directly in the application rather than by subscribing to data flows.

Optimized Data Transport

NDFS-II transports data between clients via flows that locally use shared memory or to remote hosts across the network. Clients access flows on the basis of their properties rather than locations. To achieve this network transparency, we use duplicators to transport data (see Figure 2), with flows being duplicated for multiple subscriber clients on the various hosts.

Shared memory allows concurrent reading and avoids the need for multiple copies of data. A duplicator sends data only once per remote host, not once per consumer, thus reducing network bandwidth. Also, a consumer node can suspend data transfer of a specific flow, or even crash, and the duplicator will still transport flows for the other clients.

Data transport might be irregular, depending on the network conditions or client node consumption rates. A hybrid push/pull system regularizes data transport. Each client node has a queue per flow running in a separate thread. When a producer pushes a data block, this data isn't sent immediately but instead is enqueued. The separate thread then dequeues the data to shared memory when notified by the duplicator, which sends the data to remote peer duplicators if necessary. If no consumer subscribes to the flow, the data block is dropped. If a consumer does subscribe, the flow thread symmetrically retrieves a data block from the shared memory and enqueues it in the consumer flow queue. The consumer can then pull the data from the queue.

This queue mechanism allows data transport with quality of service (QoS) and is efficient: when a consumer client node requests a buffer, it will often already be in that node's queue and thus available immediately. Moreover, queues can be customized as either blocking or nonblocking. In the latter case, queues can drop the oldest or newest buffer to make room. The API also provides dedicated methods for handling files when a loss of blocks can't be tolerated.

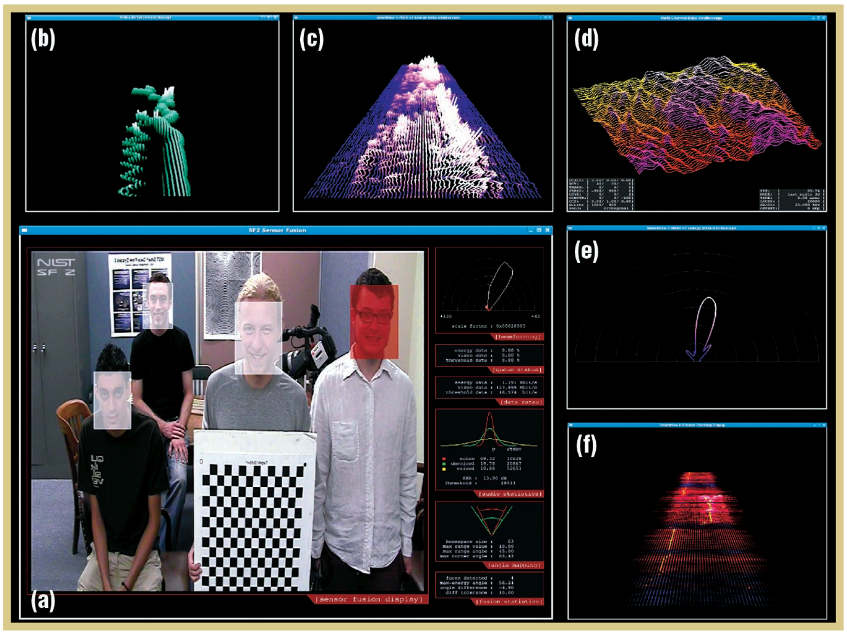


Figure 3. Using NIST Data Flow System (NDFS) to perform real-time sensor fusion of a high-definition MPEG-2 video flow from a camera and a multichannel audio flow from a 64-channel NIST Mark-III microphone array. NDFS correlates the faces' bounding boxes with the sound source's estimated bearing and then denotes the speaker with a red bounding box. (a) The right side of this panel shows audio signal processing for beamforming and speech activity detection. The surrounding panels show (b) the video tracks, (c) beamspace waterfall, (d) array raw data, (e) beam, and (f) audio tracks.

Sensor Fusion to Illustrate Operational Concepts

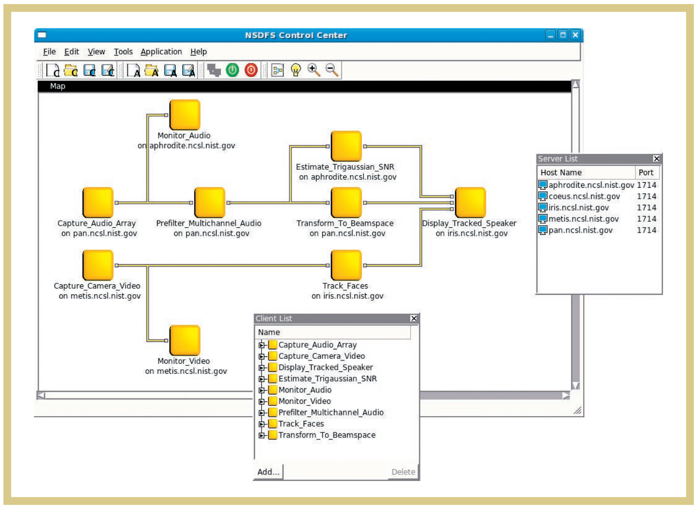
An example of audiovisual sensor fusion illustrated NDFS-II's operational concepts. Using parallel signal-processing pipelines, we processed data captured live from a 64-channel NIST Mark-III microphone array and a high-definition video camera. As the large box in Figure 3 shows, an NDFS-II client estimates the speaker's bearing from the array beamformer (top right), evaluates a speech activity model (middle right), and identifies facial regions and marks them with a white bounding box. In the final fusion step, upon detecting speech activity and matching the speaker's bearing with the face position angles relative to the array broadside is marked the active speaker's face with a red bounding box. This example uses well-known algorithms such as steered-response beamforming and Viola-Jones face localization.

Figure 4 shows the system element flow graph, as rendered by the control center. Capture_Audio_Array reads the 64-channel audio pipeline. Monitor_Audio displays the array audio. Prefilter_Multichannel_Audio applies a band-pass filter to attenuate wavelengths exceeding the array diameter and high frequencies that would be spatially aliased. Transform_to_Beamspace and Estimate_Trigaussian_SNR operate on filtered audio and offer flows to Display_Tracked_Speaker, which uses trigaussian SNR (signal-to-noise ratio) parameters for speech activity detection to decide when to use the room sound field's beamspace representation to estimate speaker direction.

The video pipeline begins with Capture_Camera_Video, which offers an MPEG-2 video flow for subscription. Monitor_Video displays the video input at the user interface. Track_Faces then produces bounding-box coordinates for the faces and offers them for subscription to Display_Tracked_Speaker.

Next, Display_Tracked_Speaker fuses the audio and video features. Estimate_Trigaussian_SNR indicates probable speech in the audio signal. Transform_to_Beamspace computes a fan of beams at increments of approximately 3 degrees, along with each beam's average power, over successive windows. This steered-response beamformer computes power as a function of angle. The speaker bearing is imputed to the angle of the dominant energy, corresponding to a face in video. Then, if the trigaussian-SNR threshold is consistent with speech energy in the dominant beam, a speaker's presence and location are suggested.

Thus, we used NDFS-II to coordinate the distributed clients for data


NSDFS Control Center
File Edit View Tools Application Help
Map
Monitor_Audio on aphrodite.ncsl.nist.gov
Estimate_Trigaussian_SNR on aphrodite.ncsl.nist.gov
Capture_Audio_Array on pan.ncsl.nist.gov
Prefilter_Multichannel_Audio on pan.ncsl.nist.gov
Transform_To_Beamspace on pan.ncsl.nist.gov
Display_Tracked_Speaker on iris.ncsl.nist.gov
Capture_Camera_Video on metis.ncsl.nist.gov
Track_Faces on iris.ncsl.nist.gov
Monitor_Video on metis.ncsl.nist.gov
Add...
Delete

Server List

Host Name	Port
aphrodite.ncsl.nist.gov	1714
coeus.ncsl.nist.gov	1714
iris.ncsl.nist.gov	1714
metis.ncsl.nist.gov	1714
pan.ncsl.nist.gov	1714

Client List

Name
- Capture_Audio_Array
- Capture_Camera_Video
- Display_Tracked_Speaker
- Estimate_Trigaussian_SNR
- Monitor_Audio
- Monitor_Video
- Prefilter_Multichannel_Audio
- Track_Faces
- Transform_To_Beamspace

Figure 4. The NDFS control center, showing the application graph and the allocation of the client list (lower center) to distributed hosts (right). NDFS establishes and transports buffered data flows to allow the sensor fusion and display results. Audio array capture, beamformed speech acquisition, bearing estimation, video capture, and sensor fusion demonstrate the function of distributed clients connected by flows.

acquisition, computing, sensor fusion, and display functions. This example application flow graph (which is available with our open source project at www.nist.gov/smartspace) consists of modular entities that can be reused—for example, to create a testbed hosting various research algorithms. Additional functional elements such as monitoring or recording nodes can be added dynamically at runtime to the flow graph. Built-in data transport mechanisms such as blocking and nonblocking queuing, along with the advanced-synchronization API, help model a wide spectrum of distributed applications. In this case, we used nonblocking data transport functionality because of the application's real-time processing character.

Lessons Learned

The earlier NDFS used a pull mechanism for data transport and had a single server for publish-subscribe flow management. The former required flow block pools at producing clients with reference-count garbage collection. So, a consumer client failure would eventually block all clients subscribing to a given flow because the reference counts couldn't reach 0 for garbage collection. The latter introduced a single point of failure, which could stop all of a running application's clients.

We addressed these issues with a hybrid push/pull mechanism using duplicators, and a peer-to-peer server protocol. Thus, we avoided the single server and made the flows nonblocking in the event of client failures. The control center is now simply another client node with display and communication services, and it doesn't maintain a central flow publish-subscribe directory, as with NDFS-I. We based flows on C++ classes, dynamic libraries, and shared memory, which allows concurrent reading and avoids redundant copy operations. It also minimizes network bandwidth by transmitting one copy per remote host even if there are multiple consuming-client nodes. Implementing core functionalities with multiplatform libraries allows cross-platform development, which insulates NDFS-II from changes in the various target operating systems.

Laboratories worldwide are developing the recognition and classification technologies needed for future pervasive interfaces. NDFS-II can host community-based research that facilitates plugging algorithms into standard and open data-flow graphs. This will enable complex multimodal interface evaluations of collaboratively developed systems. Our data-flow middleware, already in use in our collaborative research programs, can further aid researchers as they collaboratively specify, build, and evaluate advanced multimodal systems using standard corpora, performance measurement tools, and system designs.

ACKNOWLEDGMENTS

Specific commercial products or open source software projects mentioned by name are offered for information to the reader. Such references don't imply any endorsement by the US National Institute of Standards and Technology, and no product mentioned is implied to be superior to others mentioned in the article.

REFERENCES

1. J.G. Fiscus et al., "The Rich Transcription 2006 Spring Meeting Evaluation," *Proc. 3rd Int'l Workshop Machine Learning for Multimodal Interaction*, LNCS 4299, Springer, 2006, pp. 309–322.
2. R. Stiefelhagen, R. Bowers, and J. Fiscus, eds., *Multimodal Technologies for Perception of Humans*, LNCS 4625, Springer, 2008.
3. R. Want, M. Weiser, and E. Mynatt, "Activating Everyday Objects," *Proc. DARPA/NIST Smart Space Workshop*, US Nat'l Inst. Standards and Technology, 1998, pp. 7-140–7-143.
4. J.D. Flanagan et al., "Multimodal Human/Machine Communication," *Proc. DARPA/NIST Smart Space Workshop*, US Nat'l Inst. Standards and Technology, 1998, pp. 7-30–7-37.

5. J.D. Flanagan and V. Stanford, "Situation Awareness in Smart Spaces," *Proc. DARPA/NIST Smart Space Workshop*, US Nat'l Inst. Standards and Technology, 1998, pp. 3-1–3-13.

6. J. Heidemann, R. Govindan, and D. Estrin, "Configuration Challenges for Smart Spaces," *Proc. DARPA/NIST Smart Space Workshop*, US Nat'l Inst. Standards and Technology, 1998, pp. 7-30–7-37.

7. S. Basu et al., "Beyond Audio-Based Speech Recognition for Natural Human Computer Interaction," *Proc. DARPA/NIST Smart Space Workshop*, US Nat'l Inst. Standards and Technology, 1998, pp. 7-8–7-13.

8. V. Stanford et al., *Continuous Speech Recognition and Voice Response System and Method to Enable Conversational Dialogues with Microprocessors*, US patent 5615296, 1997.

9. J. McBride and V. Stanford, "The Graph Analysis and Design Technique: A Visually Oriented Systems Development Environment," *Proc. 5th Computers in Aerospace Conf.*, Am. Inst. Aeronautics and Astronautics, 1985, pp. 509–514.

10. D.T. Ross, "Structured Analysis (SA): A Language for Communicating Ideas," *IEEE Trans. Software Eng.*, vol. 3, no. 1, 1977, pp. 16–34.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Visit

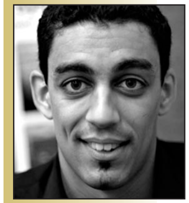
IEEE pervasive COMPUTING MOBILE AND UBIQUITOUS SYSTEMS

on the Web

www.computer.org/pervasive

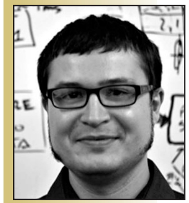
the AUTHORS

Antoine Fillinger is a system engineer with Dakota Consulting at the US National Institute of Standards and Technology (NIST) Smart Space Laboratory and the software team leader for NDFS II. His research interests include distributed and parallel computing, sensor-based networks, and system and software engineering. Fillinger has a master's in computer science from Ecole Supérieure d'Informatique et Applications de Lorraine (ESIAL). Contact him at antoine.fillinger@nist.gov.



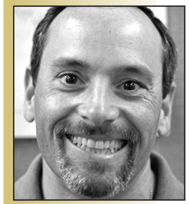
Imad Hamchi is a researcher at the NIST Smart Space Laboratory. His research interests include multimodal data-transport infrastructure and distributed complex systems, especially parallel distributed complex systems in agent-based optimization problems. Hamchi has a master's in computer, network, and telecommunication sciences from ESIAL. Contact him at imad.hamchi@nist.gov.

Stéphane Degré was a member of the NIST Smart Space Laboratory, and he has participated in the design and development of the NIST Data Flow System. His research interests included high-performance computing and networking as applied to distributed agent-based optimization. Degré has a master's in computer science from ESIAL. Contact him at stephane.degre@blue-erp.com.

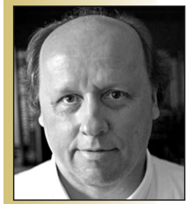


Lukas L. Diduch is a research engineer at the NIST Smart Space Laboratory. His research interests include array signal processing, speech recognition, computer vision, statistical pattern recognition, multisensor data fusion, distributed sensor networks, and ambient intelligence. Diduch has a Dipl-Ing in electrical engineering and information technology from the Technical University of Munich. Contact him at lukas.diduch@nist.gov.

Travis Rose is a member of the NIST Multimodal Information Group. His research interests include multimodal interaction and interfaces, grid computing, and computer vision technology evaluation, as applied to gesture, speech, and gaze recognition and processing. Rose has a master's in computer science from Virginia Tech. Contact him at travis.rose@nist.gov.



Jonathan Fiscus is a computer scientist in the NIST Multimodal Information Group. His research interests include performance assessment methods for speech and video content extraction technologies. Fiscus has an MS in computer science from the Johns Hopkins University. Contact him at jonathan.fiscus@nist.gov.



Vincent Stanford manages research on multimodal interfaces and distributed computing at the NIST Smart Space Laboratory. His research interests include statistical signal processing, pattern recognition, parallel processing, and systems engineering. He has applied these to speech recognition, speaker identification, electrocardiograms, phonocardiograms, acoustics, radar, sonar, and seismic signature analysis in distributed sensor-based systems. Stanford has a bachelor's degree in mathematics from Indiana University. Contact him at vincent.stanford@nist.gov.