A Virtual Sensor System for User-Generated, Real-Time Environmental Data Products

David J. Hill[a,*] Yong Liu[b], Luigi Marini[b], Rob Kooper[b], Alejandro Rodriguez[c], Joe Futrelle[d], Barbara S. Minsker[e], James Myers[f], Terry McLaren[b]

a Department of Civil and Environmental Engineering, Rutgers University
b National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign
c Amazon.com
d Woods Hole Oceanographic Institute
e Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign
f Computational Center for Nanotechnology Innovations, Rensselaer Polytechnic Institute

Abstract

With the advent of new instrumentation and sensors, more diverse types and increasing amounts of data are becoming available to environmental researchers and practitioners. However, accessing and integrating these data into forms usable for environmental analysis and modeling can be highly time-consuming and challenging, particularly in real time. For example, radar-rainfall data are a valuable resource for hydrologic modeling because of their high resolution and pervasive coverage. However, radar-rainfall data from the Next Generation Radar (NEXRAD) system continue to be underutilized outside of the operational environment because of limitations in access and availability of research-quality data products, especially in real time. This paper addresses these issues through the development of a prototype Web-based virtual sensor system at NCSA that creates real-time customized data streams from raw sensor data. These data streams are supported by meta-data, including provenance information. The system uses workflow composition and publishing tools to facilitate creation and publication (as Web services) of user-created virtual sensors. To demonstrate the system, two case studies are presented. In the first case study, a network of point-based virtual precipitation sensors is deployed to analyze the relationship between radar-rainfall measurements, and in the second case study, a network of polygon-based virtual precipitation sensors is deployed to be used as input to urban flooding models. These case studies illustrate how, with the addition of some application-specific information, this general-purpose system can be utilized to provide customized real-time access to significant data resources such as the NEXRAD system. Additionally, the creation of new types of virtual sensors is discussed, using the example of virtual temperature sensors.

Key Words: Cyberinfrastructure; Virtual Sensor; NEXRAD; Real-Time Sensing; Workflow; Environmental Sensors; Collaborative Technology; Data Integration

* Corresponding author:

David J. Hill, Department of Civil and Environmental Engineering, Rutgers the State University of New Jersey, 623 Bowser Rd., rm 110, Piscataway, NJ 08854, ecodavid@rci.rutgers.edu

Software and Data Availability

The virtual sensor system described in this paper is currently operating in prototype mode and can be accessed through the Web site http://sensorweb-demo.ncsa.uiuc.edu, which is hosted on the National Center for Super-computing Application's (NCSA) cloud computing platform. On this Web site, users can interact with the virtual precipitation sensors described in the case study below. To create new types of virtual sensors, users will need to download a desktop version of Cyberintegrator from the Web site: http://isda.ncsa.uiuc.edu/cyberintegrator/. This download is free but requires registration. The virtual sensor system software is being made available through NCSA's open source license[1] to permit interested users to create new instances of the virtual sensor system on their own servers. Interested users are invited to contact Yong Liu at yongliu@ncsa.illinois.edu for information on creating new types of virtual sensors within the prototype operational system as well as on creating new instances of the system.

Introduction

Recent advances in environmental sensing technology provide the ability to observe environmental phenomena at previously impossible time and space scales [NSF, 2004]. Not only do these observations provide valuable quantification of the space-time dynamics of large-scale environmental systems, but they also give insight into the scale relationships between processes that drive the behavior of these systems. Thus, the data provided by environmental sensor networks present potentially profound opportunities for improving our understanding of and ability to sustainably manage large-scale environmental systems [NRC 2009, 2008]. However, accessing and integrating these data into forms usable for environmental analysis and modeling can be highly time-consuming and challenging, particularly in real time [Granell et al. 2009; Horsburgh et al. 2009; Denzer 2005]. At the same time, data products published by agencies, such as the National Weather Service's (NWS) Next Generation Radar (NEXRAD) MPE and Level III products, are traditionally viewed as official final products of a particular processing

[1] http://www.opensource.org/licenses/UoI-NCSA.php

regimen. However, to meet the diverse requirements of the research and operations community, different customized products are needed.

Consider radar-rainfall data, which are a particularly valuable resource for hydrologic modeling because of their high resolution and pervasive coverage. For example, recent studies have shown that the use of direct (not gauge-corrected) radar-rainfall estimates as input to flood forecast models produces more accurate forecasts than the use of data from the existing raingauge networks [Bedient et al. 2000; Sempere-Torres et al. 1999]. This result has been attributed to the high spatial and temporal resolution of radar-rainfall data, which allow the data to more accurately reflect the spatial and temporal variability of rainfall—a feature that is especially important when forecasting in small watersheds such as those present in urban environments. Although the value of these data is well recognized by the research community, radar-rainfall data from the NEXRAD system are underutilized outside of the operational environment of the NWS for forecasting river flows [NRC 1999b]. Two studies by the National Research Council (NRC) have attributed this behavior to limitations in access and availability of research-quality data products [NRC 1999a, b]. Our research begins to address these issues through the development of a prototype Web-based system for transforming raw sensor data and producing real-time customized environmental data products, or *virtual sensors*. This *virtual sensor system* is developed around the concept of an interactive, service-based framework that encourages collaboration and facilitates the democratization of data product creation (Liu et al. 2009b). To achieve this, the virtual sensor system was created by combining a number of software components created at the National Center for Supercomputing Applications (NCSA). The main contribution of this paper is in integrating existing general-purpose components that support streaming data management, triggered workflows, and Web interaction to create an interactive service-based system that is extensible to a wide range of environmental data types. To illustrate the functionality of this virtual system, we explore two case studies, in which the virtual sensor system is employed to create real-time customized radar-rainfall data streams from raw NEXRAD data. The next section of this paper describes the interactive service-oriented architecture of our virtual sensor system. We then discuss the specific software

components that are used to implement this functionality. The NEXRAD case studies are presented next, followed by a discussion of how the virtual sensor system could be applied to other types of environmental data. Finally, conclusions and future work are discussed.

Virtual Sensor System

The virtual sensor system developed in this research not only effectively lowers the barriers for individual researchers to access officially published raw/customized sensor data products, but it also enables members of the research community to create their own customized sensor data products and to republish these products as new virtual sensor data streams in real time for sharing and reuse. It is an important distinction that the virtual sensor system does not simply provide transformation tools that researchers can download to their desktops or static customized products that can be downloaded. The components described in the next section, combined into an integrated system, provide an overall framework for tackling the tedious and often challenging tasks associated with streaming data (fetching real-time raw data streams, storing and indexing data streams, and publishing data streams), along with the analysis tasks associated with transforming the raw data into the desired data products (creating and publishing workflows as real-time services) to produce a real-time stream of user-customized data products that can be republished in one of several common data formats for sharing and reuse.

Previous studies have introduced the terms virtual sensor and software sensor to refer to the application of a model or other one-time transformation of raw sensor data to produce a higher-level data product [Cecil & Kozlowska 2009; Havlik et al. 2009; Douglas et al., 2008; Aberer et al. 2007; Jayasumana et al. 2007; Ciciriello et al. 2006; Kabadayi et al. 2006]. The virtual sensor system developed in this research, however, provides more interactivity and potential for customization and collaboration through the publication of both the derived data products and the *workflow* that created them. A scientific workflow is the description of a process for accomplishing a scientific objective, usually expressed in terms of tasks and their dependencies [Ludäscher 2009; Gil et al. 2007]; a workflow

can be reused, modified, and executed as an ongoing service to process and model incoming or historical data.

The virtual sensor system developed in this research employs a workflow system (discussed shortly) to perform the computations required to spatiotemporally and thematically transform raw measurements to more usable forms (e.g., transforming from reflectivity to precipitation). During these transformations, the virtual sensor system tracks *provenance*, meaning that it automatically records the data sources and sequences of computations as metadata [for more details see Liu et al. 2010; Moreau et al. 2008] that are made available along with the transformed data stream. The virtual sensor data products and metadata are then published by assigning unique identifiers that can be used to immediately and unambiguously access them. The virtual sensor data and metadata are published for user access using uniform resource identifiers (URIs).

Workflows and provenance tracking provide transparency to virtual sensor data, analysis, and decisions, and they can be used for community review and sharing of virtual sensor–derived results. In addition to storing the basic data provenance (i.e., data sources and processing steps) for the derived data products, our virtual sensor system also captures the full configuration of the workflow application and service infrastructure used in its production, which provides a template that can accelerate the development of new virtual sensors: researchers need only develop new transformation and/or aggregation modules and then swap them into existing workflows. The revised workflow can then be published as a new virtual sensor service. Over time, this will allow data users to select from a broad array of virtual sensor streams to support their research. Thus, virtual sensors permit research needs to drive the creation of data products, rather than having a centralized agency decide on what research data products to distribute. Finally, storing workflows, input data, and their linkages as provenance facilitates comparisons between existing and new virtual sensors and existing virtual sensors and new physical sensors. The architecture of the virtual sensor system is designed to provide a framework for *open development* and *sharing* of virtual sensor data, as well as the transformations that create these data. A high-level architecture diagram is shown in Figure 2.

As shown in Figure 2, the architecture is divided into three layers. The bottom layer is the remote sensor store layer, where the heterogeneous sensor networks reside. We consider data loggers and remote sensing repositories that are accessible through HTTP or FTP protocol as examples of such stores. The repositories that compose this layer are distributed across the Internet and are managed directly by the agencies that collect and publish the data. Examples of such repositories include the United States Environmental Protection Agency's Storage and Retrieval (STORET) Warehouse[2] and the United States Geological Survey's Water Information System Web Interface (NWIS-Web).[3] When producing the virtual rainfall sensors, this layer is one of the LDM distribution servers, which is usually an FTP server.

The middle layer is the virtual sensor abstraction layer, which is largely based on the NCSA digital synthesis framework (DSF) middleware components for building virtual observatory cyberinfrastructure. This layer provides middleware and virtual sensor data models and management tools that facilitate the retrieval, curation, and publication of virtual sensor data. Specifically, this layer implements workflow-based processing (Cyberintegrator) over a semantic content repository abstraction (Tupelo), augmented by a temporal stream management layer. The result of system operation is a set of file-like data sets (reference-able via global identifiers) that have additional descriptive information, as well as provenance and temporal relations with other data sets, recorded as queriable resource description framework (RDF) statements in well-defined vocabularies (e.g., the Open Provenance Model [Moreau et al. 2008]). The virtual sensor system uses the recorded information to generate data outputs and can provide a graphical display of the data provenance and relationships, as shown in Figure 3. Provenance and other metadata can also be accessed programmatically, for example through SPARQL protocol and RDF query language queries. Each of these components

[2] http://www.epa.gov/storet/

[3] http://waterdata.usgs.gov/nwis

are described in more detail below, followed by a summary of the typical event and data flow during operation of the system.

The top layer is a Web-based collaboration layer, where a map-based Web interface provides users the capability to deploy and visualize new instances of the virtual sensors that will be computed using the published workflows. Several visualization capabilities are provided as part of the toolkit, including basic visualization of the time-series of the streaming data in a line graph plot [Liu et al. 2008], as well as more advanced spatiotemporal visualization of changes integrated over specified geospatial areas [Liu et al. 2009a].

Software Components Enabling the Virtual Sensor System

Implementation of the virtual sensor system leverages a number of components that have been developed synergistically within the National Center for Supercomputing Applications (NCSA), namely Cyberintegrator, the streaming data toolkit, Tupelo, and a virtual sensor abstraction and management service. These are general purpose tools for workflow development, streaming data management, content management, and virtual sensor deployment, respectively. These components are described in detail below, followed by a description of a typical event and data flow scenario that illustrates how these components interact.

Scientific Workflow System (Cyberintegrator)

Cyberintegrator [Marini et al. 2007] is a graphical user interface (GUI)–based scientific workflow composition and management software that provides workflow editing and composition capability. Cyberintegrator allows a user to build a computational workflow that chains a few tasks together to accomplish a specific scientific research goal. Once the workflow is composed, it can be published on a Web server as a workflow service, which can be triggered either by a time-based execution service, which runs the workflow at a scheduled interval (e.g., every 20 minutes), or by an event-based execution service, which runs the workflow when a specific event occurs (e.g., whenever a piece of new

data arrives or a user clicks a button on the Web interface). Figure 4 shows the Cyberintegrator GUI.

The data-processing steps that derive the virtual sensor data are usually set up as a series of linked workflows. Some of these workflows depend on the results of other workflows or on the arrival of new data from physical sensors, and thus workflow execution is coordinated. The Cyberintegrator workflow system is designed to facilitate the use and coordination of workflows and their steps (hereafter referred to as modules) that are implemented in different programming languages or tools (e.g., one module may be executed in C and another in Matlab). This permits the user to implement new modules in the language of his/her preference. Modules are created externally and then imported into Cyberintegrator through a graphical wizard interface. When the workflow is executed, Cyberintegrator records metadata details about all of the entities (e.g., input and parameters) within the workflow. These metadata (provenance) capture low-level workflow processing (e.g., what transformations were done and when), as well as user activities, such as the request of a user to generate a virtual sensor through the Web interface. Several computational and data-centric science (i.e., eScience) applications have demonstrated the value of such provenance for collaborative verification of results by providing transparency to the data processing [e.g., Moreau et al. 2008; Sahoo et al. 2008]. Thus, we anticipate that this provenance information will allow users to formally verify and validate their own and others' virtual sensors.

Streaming Data Toolkit

The streaming data toolkit contained in the virtual sensor abstraction layer provides utilities to retrieve remote data streams, to trigger workflow execution based on the arrival of new data, to query local data streams using time-based semantics (e.g., latest frame), and to publish newly created virtual sensor data as a stream [Rodriguez et al. 2009]. The streaming data tool also provides a Web service interface for querying the data stream and for publishing the stream in multiple formats, including JavaScript object notation (JSON) and Open Geospatial Consortium (OGC) sensor web enablement

observations and measurements (SWE O&M) forma.[4] Such flexibility creates tremendous value for interoperability with other environmental information systems, such as the Consortium of Universities for the Advancement of Hydrologic Sciences, Inc. (CUAHSI) hydrologic information system (HIS).[5] Thus, the customized real-time data products created by the virtual sensor system can be easily integrated into existing environmental information systems to support delivery of both raw data (as in the HIS system) and processed data products.

Semantic Content Management Middleware (Tupelo)

Tupelo [Futrelle et al. 2009] is a semantic content management middleware that uses the resource description framework (RDF) to represent context as (subject-verb(predicate)-object) triples and typed files/binary objects to store content, with both types of information being managed by one or more underling physical data stores (e.g., a mySQL data base and file system). Because all of the context and content managed by Tupelo is represented generically as RDF triples and associated binary data, Tupelo is capable of managing a wide variety of data types (e.g., point/spatial data). Within the virtual sensor system, Tupelo manages the sensor data, the temporal connection of data into streams, the provenance of how the data was ingested and processed, and the configuration of the virtual sensors and triggered workflows themselves. Within Tupelo, this information is assigned unique identifiers that can be used to immediately access them across all of the interacting components and across all of the machines involved in the processing. Globally unique identifiers eliminate the need for resolving conflicts between local identifiers when data is migrated or aggregated. Access to data and metadata is provided using standard practices such as representational state transfer (REST), allowing for a variety of different access methods to be supported, while retaining the benefits of global identification (Kunze 2003). Tupelo provides applications with a core set of operations for reading, writing, and querying data and metadata as well as a framework and set of implementations for performing these operations using existing storage systems and

[4] http://www.opengeospatial.org/projects/groups/sensorweb

[5] http://his.cuahsi.org/

protocols, including file systems, relational databases, syndication protocols (such as RSS feeds), and object-oriented application data structures. It also implements the emerging open provenance model's (OPM)[6] application programming interfaces (APIs), so that provenance information across different system components can be integrated and queried [Liu et al. 2010]. Queries are expressed in SPARQL or via Tupelo's query API, and Tupelo acts as a broker between clients and a variety of underlying query engines and implementations. Query results include the URIs of the relevant content which are used to access to these data within the virtual sensor system.

Tupelo plays a critical role in this system, managing data and capturing provenance across the three layers, as well as mapping between representations as needed. The virtual sensor transformation processing described in the previous subsection, which is implemented using the Cyberintegrator workflow engine, records provenance in Tupelo using an ontology developed prior to the creation of OPM. To make this information available as OPM records, we implemented a cross-walk between these two ontologies. The resulting OPM output, which captures the end-to-end provenance of the virtual sensor data, is in a form usable in any OPM compliant tool. We were thus able to generate Figure 3 (below) using a generic OPM graphing code to traverse the PointVS (described below) virtual sensor's OPM information and automatically produce simple graphics representing process steps and dependencies. This feature permits our system to share provenance information with other OPM-compatible systems.

Virtual Sensor Abstraction and Management Service

The virtual sensor abstraction and management service provides an ontology (a data model that defines the metadata and their relationship to the virtual sensor) of virtual sensors and virtual sensor–related utility tools. The tools include virtual sensor definition tools, OGC keyhole markup language (KML) toolkits for managing spatial/temporal information, and the OGC semantic Web enablement (SWE) sensor observation service, which allows other components of the system (such as the visualization interface) to

[6] http://twiki.ipaw.info/bin/view/Challenge/OPM

retrieve the virtual sensor data stream. Data, metadata, and the registry of virtual sensor definitions are all managed via Tupelo.

Currently, both point-based virtual sensors and polygon-based virtual sensors are supported. Point-based virtual sensors represent derived measurements analogous to a single physical sensor deployed at a specific latitude/longtitude. Polygon-based virtual sensors represent derived measurements aggregated to a specified polygonal area, such as the average rainfall rate within a city boundary. The virtual sensors provide new time-series that can be at the same or different frequency as the original sensor data streams used to derive the virtual sensor measurement. A virtual sensor can also provide indirect measurements (i.e., derived quantities) that are not directly measured by physical sensors (e.g., radar reflectivity–derived rainfall rates). Furthermore, more complex analyses and optimization or simulation models can be integrated into the transformation workflow to produce virtual sensor streams of higher-level information, of the type often needed by decision makers, such as forecasts and visualization movies.

Summary of Typical Event and Data Flow Scenario during Operation of the System

The virtual sensor system has been in semi-operational mode since 2008 and currently has a live Web site at http://sensorweb-demo.ncsa.uiuc.edu, which is hosted at the NCSA's cloud computing platform. This is an event-driven, online, near-real-time system, and it currently supports both point- and polygon-based virtual rainfall sensors (detailed case studies are described in the following sections). A typical event and data flow scenario is described here to help readers understand how the system works in near-real time.

During operational mode, a continuously running data fetcher program (part of the streaming data toolkit described above) on the host server fetches remote sensor data from sensor data stores (e.g., NEXRAD Level II data in a remote FTP server) at a user-defined frequency. The newly fetched data is deposited into a local Tupelo-managed repository of the system. Each new raw sensor data packet triggers a set of virtual sensor

transformation workflows, which compute derived data products and re-publish the resulting data streams as new live virtual sensor data streams, again using the streaming data service publishing capability.

The Web-user interface front-end (e.g., Figure 5) can be used to view, query, and explore the set of existing virtual sensors and their data. The interface can also be used to add a new point-based virtual sensor. A click of the mouse on the map will initiate recurring data-triggered execution of an associated back-end workflow to produce a new live data stream from a new virtual sensor at that point. Users can also upload a new KML file to trigger a new workflow based on the polygon information contained in the KML file to generate a polygon-based virtual sensor. Minimal fault-tolerance capability is built into the system so that corrupted new raw sensor data packets will not trigger any workflow execution. At this time, data-gap filling methods have not been implemented within the virtual sensor system (this capability could be added to the virtual sensor system, by us or by third parties, via additional workflow modules). Thus, downstream applications currently need to be designed to accommodate data gaps. In this work, we checked for gaps by adding a data integrity checking algorithm (e.g., checking the header of the NEXRAD Archive II file) in the streaming data fetcher prior to the triggering of associated workflows. A purging workflow is run on the server to remove outdated raw sensor data in the local repository to conserve data storage space, a service that can be flexibly scheduled or disabled, depending on whether storage space is a concern or not.

Creating Virtual Rainfall Sensors

To illustrate the use of the virtual sensor system for an environmental application, we implemented virtual rainfall sensors by linking radar-rainfall-specific processing modules to the general purpose virtual sensor system described above. This section begins with a description of the weather radar data used by the virtual rainfall sensors. Radar-rainfall-specific processing modules are then introduced, followed by a description of how they were linked together to create virtual precipitation sensors that produce customized radar-rainfall products in real time.

NEXRAD System and Data Products

The NEXRAD system is composed of over 100 weather surveillance 1988 Doppler (WSR-88D) radar installations located throughout the United States and selected overseas areas. The WSR-88D operates by sending out electromagnetic pulses from an antenna that rotates around a central axis and measuring the reflections of these pulses on airborne objects. Each 360° rotation is referred to as an elevation scan, and several different elevations are measured to create one volume scan. Up until 2009, the WSR-88D had a standard resolution of 1° azimuth by 1 km (hereafter referred to as *legacy resolution*) and a range of 460 km. After this time, the radars began to be upgraded incrementally to *super-resolution* which has a resolution of 0.5° by 0.25 km. The number of elevation scans in each volume scan is selected by the radar on the fly to accelerate the volume scans during rainfall events in order to increase the temporal resolution of the data (at the expense of spatial resolution). As currently designed, the WSR-88D takes approximately 5, 6, or 10 minutes to complete a volume scan, depending on the scanning strategy. Thus, the raw radar data are reflectivity measurements that represent spatial averages over the radar gates (i.e., cells in the radar coverage map defined by the radar resolution) in each elevation scan at (approximately) instantaneous points in time. For each elevation scan, these measurements are referenced on a planar polar grid (defined in terms of azimuth and range) centered at the radar.

Following their measurement, the raw reflectivity data from each radar are processed to create *products* that represent estimates of meteorological process variables [Fulton et al. 1998; Klazura & Imy 1993]. The rainfall products are categorized according to a hierarchy that indicates the increasing amount of preprocessing, calibration, and quality control performed [Klazura & Imy 1993; Fulton et al. 1998; Wang et al. 2008]. This hierarchy is illustrated in Figure 1.

Stage I data are further subdivided into Level I, Level II, and Level III data, referring to the original reflectivity measurements made by the radar, the analog to digital converted raw measurements, and 41 data products, respectively. Within Stage I, Level III, there are five precipitation products: one-hour precipitation total (N1P), three-hour precipitation

total (N3P), storm total precipitation (NTP), digital precipitation array (DPA), and digital storm total precipitation (DSP). These products provide an estimate of the surface rainfall, and thus are represented as two-dimensional maps. The N1P, N3P, and NTP products are represented on the legacy resolution polar grid. Currently, the data from radars producing super-resolution Level II data are recombined to produce legacy resolution Level III products. The DPA and DSP products are represented on a 4-km by 4-km grid derived from the hydrologic rainfall analysis project (HRAP). The N1P and DPA products represent one-hour rainfall averages, the N3P product represents a three-hour rainfall average, and the NTP and DSP products represent variable time averages based on storm durations.

All five of these products are based on a direct conversion of reflectivity to rainfall based on the *Z*-*R* power law,

$$Z = aR^b \tag{1}$$

where Z is the radar reflectivity (mm^6/mm^3), R is the rainfall rate (mm/hr), and a and b are parameters related to the drop size distribution [NWS-ROC 2003]. The default values of a and b used by the NEXRAD system are 300 and 1.4, respectively, although the radar operator has the option of changing these parameters on the fly based on his/her intuition or experience.[7]

Stage II provides estimates of hourly rainfall accumulations that merge the DPA and DSP products with quality-controlled rain gauge measurements [Fulton et al. 1998]. Multi-sensor precipitation estimator (MPE) refers to a mosaic of gauge-adjusted rainfall products from multiple radars that cover an entire forecasting region of a river forecast center (RFC). Because of the quality control necessary to create the Stage II and MPE data products, their latency (on the order of an hour) is such that they cannot be classified as real-time products.

[7] Personal communication, Dennis Miller, National Weather Service, Office of Hydrologic Development.

The data types and transformations used to create the Level III, Stage II, and MPE data are tailored to the needs of the NEXRAD agencies, and to the RFCs in particular. However, to use these data for research, different transformations (e.g., *Z-R* transformations), interpolations, or aggregations (e.g., 15 minute averages on a 1-km by 1-km grid for urban environments with rapid hydrologic response) may be desired. To avoid loss of resolution, these research data products should be derived from the raw (i.e., Level II base reflectivity) data rather than from the higher-level products. This is especially true given the recent introduction of super-resolution Level II data, given that currently, the Level III and higher products are recombined by the NWS to the legacy resolution.

Level II data are distributed in real time from the NWS through Unidata's Internet Data Distribution (IDD) project,[8] which is designed to deliver data to universities as soon as they are available from various observing systems. The delivery vehicle employed by IDD is Unidata's local data manager (LDM) software,[9] which captures and manages the data on a local server. The Level II data from a single volume scan of a radar are distributed through LDM as a single NWS Archive II binary file [NWS-ROC 2008a, 2b].

Recently, Krajewski et al. [2008] presented the Hydro-NEXRAD prototype, a system that provides researchers with historical NEXRAD Level II data at the radar or watershed level. Hydro-NEXRAD facilitates the transformation of historical Level II data using a set of predefined options to achieve a customized output from multiple radars. The output from the Hydro-NEXRAD system is currently limited in scope to a well-defined historical time window that must be prior to implementation of super-resolution data in March 2009, and the derived products are available at a limited number of radars and transmitted only to the requesting individual. A near-real-time version of Hydro-NEXRAD (Hydro-NEXRAD II) is under development (Krajewski, personal

[8] http://www.unidata.ucar.edu/software/idd/

[9] LDM is freely available from http://www.unidata.ucar.edu/software/ldm/

communication) that delivers an ongoing data stream of rainfall estimates from the super-resolution data, using a rectangular grid surrounding a particular watershed.

When applied to NEXRAD data, the virtual sensor system discussed here allows near-real-time custom transformation and aggregation of the Level II data, enabling researchers to implement their own transformations of the reflectivity data and combine them with a library of pre-existing software modules (e.g., format conversion and data aggregation or transformation). The resulting data products (virtual rainfall sensors) can be made available to a larger community (the entire Web community or a smaller group, such as a project team) as soon as they are published. Additionally, using the NEXRAD-specific processing modules, the virtual rainfall sensors developed in this research have the capability to deliver rainfall estimates for any user-specified custom region or point for which NEXRAD coverage exists.

Deploying Virtual Rainfall Sensors

Two types of virtual rainfall sensors are currently deployed in the virtual sensor system. The first virtual rainfall sensor (PointVS) converts raw radar data into a rainfall estimate at a particular point in space at a regular (user-specified) frequency. An illustration of the processing steps performed by this virtual rainfall sensor is given in Figure 6. The second virtual rainfall sensor (PolyVS) converts the raw radar data into a rainfall rate estimate averaged over a spatial polygon at the temporal frequency of the radar. This virtual rainfall sensor is illustrated in Figure 7. These virtual rainfall sensors are created by linking data-processing modules together in a workflow that is triggered by a data fetcher provided by the streaming data toolkit.

Because of the irregular measurement frequency of the radar, the data fetcher module checks a local LDM server for new measurements every minute. Given that the fastest radar volume coverage pattern (VCP) takes 5 minutes, this frequency is sufficient to capture new data in a timely manner, but more frequent data checks can easily be implemented if needed. When a new radar measurement is available, it is archived, and

workflows that depend on new data from the radar, such as the PointVS and PolyVS workflows described in more detail below, are triggered.

These virtual sensors can be deployed by specifying values for the required and optional parameters (discussed below) on a Web form within the Web-based virtual precipitation sensor GUI. Furthermore, at the workflow level, these templates can be modified to create new template virtual sensors by adding processing modules or replacing processing modules with alternative methods. The remainder of this section describes the radar-rainfall-specific processing modules in detail.

Radar-Rainfall Point Estimator

This module creates an estimate of the rainfall rate at a user-specified point at the time of a radar scan using the following steps: interpreting the binary data file containing the radar data, registering the polar grid of the measurements with the United States Department of Defense World Geodetic System 1984 (WGS-1984) coordinate system [NIMA 2000] used by the global positioning system (GPS), interpolating the radar reflectivity to the user-specified point, and (if indicated by the user) transforming the reflectivity to rainfall rate in mm/hr using the *Z-R* relationship (Equation 1). The required user input to create a new virtual sensor using this workflow is the GPS coordinates of the point at which to estimate the rainfall rate, while other optional input parameters include the rain threshold, hail cap, and *Z-R* parameters, as discussed shortly.

Following Smith et al. [2007], the point rainfall estimates are derived from the lowest elevation scan in each volume scan (usually 0.5°). The reflectivity measurements are projected onto a local plane coordinate system centered at the radar. This coordinate system is registered with the WGS-84 coordinate system [Zhu 1994]. Once the point of interest (i.e., the user-selected point location) and the radar data are in the same coordinate system, the reflectivity at the point of interest is interpolated using a distance-weighted average of the reflectivity in the four nearest radar gates [Smith et al. 2007]. The reflectivity is filtered to remove signals that are too weak to be indicative of rainfall (rain threshold) and signals that are too strong to be indicative of liquid rainfall (hail cap).

The default values for the rain threshold and hail cap used by the virtual sensor system are 18 dBZ and 53 dBZ, respectively [Fulton et al. 1998].

Finally, the interpolated reflectivity is converted to rainfall rate using Equation 1. The default values of the parameters a and b are 300 and 1.4, respectively; however, different parameters may be specified in the workflow.

Radar-Rainfall Polygon Estimator

This module creates an estimate of the rainfall rate aggregated over a user-specified geospatial polygon at the time of a radar scan using a process similar to that of the point-based estimator, except that instead of interpolating the radar data to a point, it is averaged over the user-specified geospatial polygon The required user input to create a new virtual sensor using this workflow is the KML file defining the polygons over which to average the rainfall rate, while other optional input parameters include the rain threshold, hail cap, and *Z*-*R* parameters, as discussed in the previous section. The polygons are discretized into a Cartesian grid with default granularity of 0.5-km by 0.5-km, a resolution indicated by Vieux and Farfalla [1996] to sufficiently fill the Cartesian grid from the polar grid. Like the point-based virtual rainfall sensor described above, the polygon-based virtual rainfall sensor uses the reflectivity from the lowest elevation scan, which is projected downward to create a polar grid defined on a flat plane at the land surface. The average reflectivity for the Cartesian grid cells defining each polygon is computed using a distance-weighted average of the four closest radar pixels surrounding the Cartesian cell centroid. This procedure is the same as calculating the point-based rainfall estimate at the cell centroid, and thus accounts for the rain threshold and hail cap as discussed previously.

KML defines polygons as a sequence of adjacent vertices georeferenced with GPS coordinates. The virtual sensor management middleware extracts the list of polygon vertices and passes this list into the module as a parameter. The vertices are then transformed from the WGS-84 coordinate system to the east-north-up (ENU) Cartesian coordinate system of the local grid [Zhu 1994]. An efficient polygon-filling algorithm is

then performed, which parses the rows of the grid in south-to-north order once, identifying the columns in each row through which an arc of the polygon passes. Assuming that the polygon falls completely within the grid, then processing the list of columns in east-to-west order quickly reveals the pixels with centroids that fall inside the polygon. Once the Cartesian grid cells that fall within the polygon are identified, the rainfall rate (calculated by Equation 1) of these cells is calculated and averaged. An illustration of the pixelation of the polygon is shown in Figure 8.

Temporal Aggregator

The point-based and polygon-based rainfall estimators described above operate on individual radar volume scans and thus create radar-derived rainfall estimates at the original temporal frequency of the radar. Because the radar data represent instantaneous measurements, temporal aggregation is needed to produce the regular frequency time-series data in the form of Δt-minute accumulations, used as input by many models. This module performs temporal aggregation from time t to $t+\Delta t$. The module requires the user to specify the granularity of the output time series (Δt).

The temporal aggregation module queries Tupelo to retrieve radar-rainfall estimates (e.g., from the radar-rainfall point estimator). The irregular frequency of the radar measurements produces the need for a specialized query that we call a *now-minus-delta-plus* query, which is provided by the streaming data toolkit. This query retrieves the data corresponding to the time period extending from $t_c-\Delta t$ (where t_c is the current time) to t_c, as well as the most recent datum that is strictly less than $t_c-\Delta t$. For example, if the aggregation period is 20 minutes, it is currently 12:00, and the most recent measurements from the radar came at 11:58, 11:46, and 11:34, then the now-minus-delta-plus query for a 20-minute granularity will retrieve all of these measurements from the Tupelo-managed semantic content system (despite the fact that 11:34 occurred more than 20 minutes ago). The reason for this type of query is that it is highly unlikely that there will be a radar record corresponding to exactly time $t_c-\Delta t$, and thus this value is estimated by linear interpolation. Following this logic, it is also unlikely that a measurement will exist exactly at the current time (t); however, rather than wait for the next measurement to

arrive from the radar (which would increase the latency of the derived estimate), this value is estimated as being equal to the most recent datum. Using these estimates of the boundary points, along with the actual measurements that occur during the aggregation period, the Δt-minute accumulation (in mm) is calculated by integrating the time-series using the trapezoidal rule [Chapra & Canale 2001].

Case Studies Using Virtual Rainfall Sensors

To demonstrate the virtual sensor system, two case studies are explored: (1) radar-raingauge comparison and (2) modeling urban flooding. These case studies are drawn from the authors' ongoing research efforts, for which the virtual sensor system is currently being run as an operational capability to supply rainfall data products in real time. These case studies both employ data from the Romeoville WSR-88D radar (KLOT) in Illinois.

Real-Time Radar Bias Adjustment and Gauge Quality Control

Radar-rainfall estimates provide information of the spatial distribution of rainfall at a resolution unparalleled by most operational raingauge networks. However, because of the nature of radar-based observation, the uncertainty in these measurements can be difficult to quantify [Battan 1976; Austin 1987; Austin 1987; Smith & Krajewski 1993; Steiner & Smith 1998; Tustison et al. 2001]. In the operational environment, this uncertainty has led to the adjustment of radar-rainfall estimates using ground-based raingauge data to improve their accuracy [Smith & Krajewski 1991]. This adjustment usually takes the form of a multiplicative bias term. However, because raingauges deployed in the field often malfunction, this method can introduce significant errors into the gauge-adjusted radar-rainfall estimate if the gauge data are not carefully quality controlled before being used for bias adjustment [Steiner et al. 1999]. Although methods for calculating the bias term in real time have been proposed [e.g., Seo et al. 1999], these methods require that the gauge data be clean, and given that automatic methods for raingauge quality control are still in the experimental stage, real-time bias correction cannot yet be performed operationally. For these reasons, there has been much research into understanding the relationship between radar-rainfall measurements and ground-based measurements,

including scaling of precipitation measurements in space and/or time [Habib et al. 2004; Grassotti et al. 2003; Tustison et al. 2003; Tustison et al. 2001; Ciach & Krajewski 1999], conversion of radar reflectivity to rainfall estimates [Morin et al. 2003; Ciach et al. 1999; Smith et al. 1996; Xiao & Chandrasekar 1997], and real-time bias correction [Henschke et al. 2009; Seo et al. 1999].

This case study demonstrates how point-based virtual sensors can be set up to estimate 10-minute rainfall accumulations at the location of five telemetered tipping bucket raingauges reporting 10-minute rainfall accumulations. The raingauges, the locations of which are illustrated in Figure 9, cover a region of approximately 55 mi^2. The five resulting virtual sensor data streams are currently being used by the authors to develop new real-time bias adjustment procedures, but they could also have many other possible applications as described above.

Each of the virtual sensors is set up through the Web interface using the PointVS virtual sensor by specifying the location of the new virtual sensor using the GPS coordinates of one of the tipping bucket gauges and using the default virtual sensor rain threshold and hail cap and Z-R relationship (a = 300, b = 1.4). For the temporal aggregation, a 10-minute accumulation is selected, because this is the resolution of the gauges used in this study.

Figure 10 compares the time-series of point-based radar-rainfall estimates to the gauge estimates for a 20-hour period beginning at 16:00 UTC on June 4, 2007. This figure shows that, although there is not perfect agreement between the virtual sensors and the raingauges, the two types of sensors produce very similar results. The differences between the radar-derived virtual sensor data and the rain gauge data can be attributed to either uncertainty in the radar-rainfall relationship (discussed above) or to measurement errors by the sensors (primarily the raingauges). Currently, the virtual sensor system does not account for uncertainty in the virtual sensor data products; however, recent work

in representing uncertainty as metadata (e.g., UncertML[10]) could provide an avenue to add this capability to the system. The close correspondence between the virtual sensor data and the raingauge data has proven useful for identifying errors within the raingauge data stream. Figure 11 compares the point-based radar-rainfall estimates produced by the virtual sensor system with the tipping bucket gauge observations over an 8-hour period beginning at 18:00 UTC on August 23, 2007. At locations A, C, D, and E, the virtual sensors and gauges produce very similar measurements, indicating two rainfall events that begin at approximately 20:00 and 23:00, respectively. At location B, however, the gauge and virtual sensor data deviate significantly, with the gauge reporting no rain during the entire 8-hour period and the virtual sensor indicating two rainstorms that correspond with the rainstorms observed at the other four locations. The usual correspondence between the gauge and virtual sensor data, combined with evidence from the other four locations that two rainstorms passed through the study region, suggests that the gauge at location B was malfunctioning. These data are being used in the authors' current research to develop a real-time Bayesian radar-raingauge data fusion algorithm that will be implemented as a virtual sensor in the future.

Providing Rainfall Estimates for Urban Drainage Models

Traditionally, real-time raingauge networks have been employed to provide data for urban drainage models; however, these networks are generally not dense enough to provide accurate flood forecasts [Sharif 2006; Bedient et al. 2003]. Recently, several studies have documented the potential of radar-rainfall estimates for improving forecasts of flooding and drainage in urban watersheds [Einfelt et al. 2004; Smith et al. 2007; Bedient et al. 2000; Sempere-Torres 1999], as well as for predicting critical events associated with urban flooding (e.g., combined sewer overflows) [Thorndahl et al. 2008; Roualt et al. 2008; Vieux & Vieux 2005; Faure & Auchet 1999]. This case study will demonstrate the use of the virtual rainfall sensors for providing real-time spatially and temporally averaged rainfall rates in support of combined sewer overflow (CSO) forecasting for the City of Chicago.

[10] http://www.uncertml.org/

In order to understand the effect of operational decisions on CSOs, a drainage model that discretizes Chicago into *sewersheds* (i.e., the spatial region that drains to a particular CSO outfall) is being employed to forecast the result of implementing particular management strategies. As illustrated in Figure 5, the City of Chicago has approximate dimensions in the north-south and east-west directions of 45 km and 12 km, respectively, and it is divided into approximately 300 sewersheds that range in size from approximately 0.5 km^2 to 20 km^2. The sewershed-scale virtual rainfall sensors for Chicago were set up using the PolyVS virtual sensor described above. Figure 4 shows the Cyberintegrator GUI for this instance of the virtual sensor. A KML file delineating the sewersheds is used as one of the inputs of the PolyVS, and the default rain threshold and hail cap and the standard convective *Z-R* relationship (a = 300, b = 1.4) are used for the radar-rainfall conversion.

Figure5 illustrates the sewershed-average rainfall rate over the Chicago study area at 21:52 UTC on April 23, 2010. This figure illustrates the high degree of spatial variability in the amount of rainfall each sewershed receives during a rain event and suggests the spatial variability of sewer loading during the storm. Visualizations such as this can be used to better understand the response of the sewer system to rain events, and the data underlying them can be incorporated into a model to predict the impacts of current management decisions on future CSOs. This model is designed to run either in reanalysis mode (i.e., using historical data) or in real-time mode. In reanalysis mode, the sewershed-averaged rainfall rates are provided as a flat data file that is created by archiving the real-time data produced by the virtual sensors. In real-time mode, the model takes advantage of the streaming data toolkit, stepping through time by requesting the most recent measurement, processing it, and waiting until a new measurement is produced by the virtual sensors.

Creating Additional Virtual Sensors

The virtual rainfall sensors described above were created by linking processing modules together using Cyberintegrator to form workflows that are triggered by time-based and/or event-based execution services provided by the streaming data toolkit. These processing modules are implemented as executables that are loaded into a Cyberintegrator transformation module repository within the prototype virtual sensor system.

New types of virtual sensors other than the types described in the case study or for regions of the world other than the Chicago area used in the case study can be implemented and deployed in the prototype system by creating and registering new workflows within the prototype. As mentioned above, the virtual sensor system software publishes the derived data sets by executing a predefined set of workflows at regular intervals using time-based and/or event-based execution services. Each workflow contains the code to retrieve the data from external Web services, to ingest and transform such data, and finally to publish the derived data on appropriate data streams in the repository that will then be picked up by the application running the Web front-end that the user interacts with.

New virtual sensors can be developed by creating new workflows using the Cyberintegrator workflow management system, which must be installed on the user's computer. Details on acquiring Cyberintegrator are discussed in the "Software Availability" section of this paper. Connection of the local installation of Cyberintegrator to the remote module repository hosted by a specific deployment of the virtual sensor system software permits the user to access previously used modules and workflows and to deploy new modules and workflows within the virtual sensor system deployment. If the transformation modules needed for the new workflow are already available in the remote Cyberintegrator repository, then the user can compose and configure a new workflow using the visual programming interface provided by Cyberintegrator. This is accomplished by selecting the transformation modules from the module repository, configuring the input parameters, and connecting the outputs of one module to the inputs of another to create the desired processing sequence.

If new transformation modules are desired that are not already available in the Cyberintegrator repository, they can be added using a GUI-driven wizard. Separate wizards are available for importing transformation modules based on how the modules are executed (command line, Java code, Matlab code). This wizard-driven approach facilitates the incorporation of existing software into the virtual sensor system. Furthermore, given that command line executable modules can be imported into Cyberintegrator, new transformation modules can be created in any programming language the user is comfortable with, because code in almost all languages can be compiled and executed as a command line tool. Once the modules are created, they must be saved to the remote module repository hosted by the virtual sensor system deployment. This can be done through drag-and-drop operations within Cyberintegrator. Once all the required modules are stored in the remote repository, they can be assembled into a workflow using Cyberintegrator's visual programming interface. Saving this workflow will store it in the remote repository as well.

Once the new modules and workflows have been created, they can be registered with the execution services provided by the streaming data toolkit and the Web application front-end. At this point in time, these two steps require assistance from the deployment manager. For the virtual sensor system deployment discussed in this paper, the appropriate contact is Yong Liu (yongliu@ncsa.illinois.edu). From then on, users interacting with the virtual sensor system will be able to see and retrieve data specific to the new virtual sensor.

The prototype virtual sensor system deployed to run the case study discussed above contains a set of modules specific for precipitation data derived from NEXRAD data for a small area of the United States. Users can register their own transformation modules for other types of data and/or other regions of the world within this system, thus extending the applicability of this particular instance of the virtual sensor system software. As more users upload modules to the current prototype, the ability to combine existing modules to address new problems will be increased, and the robustness of the underlying virtual

sensor software may be improved through user comments (including bug reports). Thus, we expect that as more users participate in the system, it will become easier for users to create their own custom virtual sensors within the prototype system because there will be more existing modules registered within it. Thus, new users could reuse other users' compiled code and assemble virtual sensors through Cyberintegrator's visual programming interface.

New instances of the virtual sensor system software can be deployed by interested users on their own machines. This process is beyond the scope of the current paper, as the virtual sensor system described in this manuscript is developed within an interactive, service-oriented framework. Service orientation permits remote users to interact with the existing system, creating a centralized forum for users to create virtual sensors. This community development aspect is one of the key features of the system we have created. Because module repositories are local to the particular virtual sensor system deployment, the existence of multiple virtual sensor system instances can reduce the opportunity for users to reuse existing modules, given that these modules may be spread over several non-interacting deployments of the virtual sensor system software. Thus, we encourage users to implement their own virtual sensors within the existing virtual sensor system prototype currently hosted at NCSA. If users want to set up their own instance of the system, however, they can do so, as all the software is being made available under NCSA's open source license. Please see the software availability section for more information.

Creating Virtual Temperature Sensors

To illustrate the process for creating new virtual sensors, consider the case of virtual temperature sensors. Air temperature is one of the key parameters in the land surface energy budget; thus, it is an important input parameter for many types of environmental models, including hydrologic and climate models. The daily maximum and minimum temperature are commonly used to characterize the warming (or cooling) process at the daily time scale (for example to calculate growing degree days). Minimum and

maximum temperatures throughout the United States are measured by ground-based meteorological sensors participating in the NWS Cooperative Observer Network.[11] These data represent point measurements of daily minimum and maximum temperatures across a relatively sparse network of sensors (there are approximately 8,000 stations in the continental United States). These data are transmitted to the NOAA National Climactic Data Center (NCDC) in near-real time and are served to data consumers via Web services.[12] In many cases, it is important to estimate the minimum/maximum temperature at a particular point location that does not have an existing sensor. Such information could be used, for example, to estimate freeze-thaw cycling on infrastructure components (FHWA 2006).

Implementation of such a virtual temperature sensor would require the addition of a data fetcher for the raw minimum/maximum temperature data and an interpolation module to estimate the minimum/maximum temperature at ungauged locations. The temperature data fetcher could be created by modifying the existing NEXRAD data fetcher by directing it to connect to the NCDC database of raw cooperative observer network data, using the available Web services. Again, by checking for new data more frequently than the data are produced (in this case daily), we can ensure that the data latency is minimized. Acquisition of a new day's worth of data by the data fetcher would trigger a minimum/maximum temperature point estimator that would interpolate the raw data to a set of user-generated points of interest. This point temperature estimation could be performed by inverse-distance-weighted spatial interpolation, which has been shown by several researchers to be an appropriate method for minimum/maximum temperature data (Jolly et al. 2005; Jarvis & Stewart 2001a, b). Once these modules had been implemented in a programming language and compiled into an executable, they would be uploaded into the Cyberintegrator module repository using the import wizard for command line executables. They would then be connected to create the desired processing sequence using Cyberintegrator's visual programming interface. Finally, the

[11] http://www.weather.gov/om/coop/

[12] http://www.ncdc.noaa.gov/ws/

new workflow would be registered with the time-based execution service and the Web application front end to create the virtual temperature sensor. Instances of this virtual sensor could be deployed through the Web interface in the same way as the point rainfall virtual sensor.

Conclusions

This paper presents a prototype virtual sensor system for environmental observation that facilitates real-time customization of physical sensor data and publication (through the assignment of URIs) of the processed data products, as well as the workflows and provenance involved in creating the data products. Given appropriate transformation, interpolation, and extrapolation models, the system is capable of providing estimates of environmental variables at any user-specified custom region or point (for which sufficient physical sensor data is available). The system is designed to meet the needs of geographically dispersed researchers with different specializations and will be particularly helpful for sharing these data in centrally managed environmental observatories. By adding modules to the general virtual sensor system that provide access to the NEXRAD data streams and that perform spatial, temporal, and thematic (i.e., reflectivity to rainfall rate) transformations of the NEXRAD data, two types of virtual rainfall sensors were created. These virtual rainfall sensors lower some of the barriers noted by the NRC [1999a, b, c] to accessing and using data collected by the NEXRAD system. The improved access provided by virtual rainfall sensors is particularly important given the recent deployment of super-resolution NEXRAD data, which presents an opportunity for observing rainfall at an unprecedented range of scales but also creates even greater challenges in manipulating larger data files.

Currently, this pilot system can provide point-averaged radar-rainfall products at either the temporal resolution of the radar or as a temporal average over a fixed time period, as well as polygon-averaged radar-rainfall products at the temporal resolution of the radar. As shown in the case studies, these types of virtual rainfall sensors have many scientific and operational uses, including understanding the relationship between radar

measurements and the rain that reaches the land surface, adjusting the radar data using raingauge observations to mitigate sampling errors, and creating and serving real-time data products in support of real-time forecasts. Although this system was demonstrated using only data from the KLOT radar, the system is generic and can be deployed for other WSR-88D radars, for more sophisticated radar-rainfall transformations (e.g., by deploying algorithms from the Hydro-NEXRAD system into workflows), or for other types of physical sensors besides weather radars.

This system is designed as a community tool and has a number of features designed to reduce the effort required to adapt existing sensor data to specific research uses and to support interactions between researchers to accelerate the pace of scientific discovery. Users can leverage deployed virtual sensor types, interactively creating and sharing new customized virtual sensors at required locations and with parameters best suited to the researcher's purpose. Creation of new types of virtual sensors is also possible. Researchers can use published virtual sensor workflows as templates or work on their desktop computer in the programming language(s) of their choice to create workflows instantiating new types of virtual sensor that can be published for community use alongside, or as a replacement for, existing sensor types. Workflows can also be created that ingest virtual sensor data and process it further, enabling reuse of intermediate products developed by other community members. We believe that the ability for community users to create new virtual sensors will facilitate investigations of alternate radar-rainfall data products developed using different algorithms—for example, different conversions from reflectivity to rainfall rates—and/or using new physical sensors (e.g., X-band radars). Additionally, we expect that the provenance tracking feature will facilitate comparisons between existing and new virtual sensors and between virtual and physical sensors.

Acknowledgements

The authors would like to acknowledge the Office of Naval Research, which supports this work as part of the Technology Research, Education, and Commercialization Center (TRECC) (Research Grant N00014-04-1-0437), and the TRECC team working on the Digital Synthesis Framework for Virtual Observatories at NCSA. We also thank the

UIUC/NCSA Adaptive Environmental Sensing and Information Systems initiative for partially supporting this project. The authors would like to thank Dr. Thomas Over and Mr. David Fazio (USGS Illinois Water Science Center) and Ms. Catherine O'Connor (Metropolitan Water Reclamation District of Greater Chicago) for contributing data and support used to develop the case studies. The authors also acknowledge Dr. Dennis Miller (NWS Office of Hydrologic Development) for his suggestions and insight regarding precipitation observation. We also thank Sam Cornwell at the University of Illinois at Urbana-Champaign for his assistance in implementation of the virtual sensor system.

References

Aberer, K., Hauswirth, M., and Salehi, A. (2007). Infrastructure for data processing in large-scale interconnected sensor networks. Proceeding of the 8th International Conference on Mobile Data Management, MDM 2007, pp. 198-205.

Austin P.M. (1987). Relation between measured radar reflectivity and surface rainfall. Monthly Weather Review, 115, 1053-1070.

Battan L.J. (1976). Vertical air motions and the Z-R Relation. Journal of Meteorology, 15, 1120-1121.

Bedient, P.B., Hoblit, B.C.,Gladwell, D.C., and Vieux, B.E. (2000). NEXRAD radar for flood prediction in Houston. Journal of Hydrologic Engineering, 5(3), 269–277.

Bedient, P.B., Holder, A., Benavides, J.A., and Vieux, B.E. (2003). Radar-based flood warning system applied to Tropical Storm Allison. Journal of Hydrologic Engineering, 8, 308-318.

Cecil, D. and Kozlowska, M. (2009). Software sensors are a real alternative to true sensors. Environmental Modelling and Software. doi: 10.1016/j.envsoft.2009.05.004.

Chapra, S.C., and Canale, R.P. (1998). Numerical Methods for Engineers, 3rd Edition. New York: McGraw Hill.

Ciach, G.J. and Krajewski, W.F. (1999). On the estimation of radar rainfall error variance. Advances in Water Resources, 22(6), 585-595.

Ciach, G.J., Krajewski, W.F., Anagnostou, E.N., Baeck, M.L., Smith, J.A., McCollum, J.R., et al. (1997). Radar rainfall estimation for ground validation studies of the tropical Rainfall Measuring Mission. Journal of Applied Meteorology, 36, 735 -- 747.

Ciciriello, P., Mottola, L. and Picco, G.P. (2006). Building virtual sensors and actuators over logical neighborhoods. In International Workshop on Middleware for Sensor Networks, MidSens 2006. Co-Located with Middleware 2006, pp. 19-24.

Douglas, J., Usländer, T., Schimak, G., Esteban, J.F., Denzer, R. (2008). An open distributed architecture for sensor networks for risk management. Sensors 2008, 8, 1755-1773.

Denzer, R. (2005). Generic integration of environment decision support systems – state-of-the-art. Environmental Modelling & Software 20 (10), 1217–1223.

Einfelt, T., Arnbjerg-Nielsen, K., Golz, C., Jense, N.-E., Quirmback, M., Vaes, G., and Vieux, B. (2004). Toward a roadmap for use of radar rainfall data in urban drainage. Journal of Hydrology, 299, 186 – 202.

EPA (United States Environmental Protection Agency) (2004), Report to Congress on Impacts and Control of Combined Sewer Overflows and Sanitary Sewer Overflows (Report No. EPA 833-R-04-001), Environmental Protection Agency.

Faure, D., and Auchet, P. (1999). Real time weather radar data processing for urban hydrology in Nancy. Phys. Chem. Earth, 24(8), 909 – 914.

FHWA (Federal Highway Administration) 2006. Verification of Long-Term Pavement Performance Virtual Weather Stations: Phase I Report—Accuracy and Reliability of Virtual Weather Stations. Publication No. FHWA-RD-03-092. Federal Highway Administration: McLean, VA.

Futrelle, J., Gaynor, J., Plutchak, J., Myers, J., McGrath, R., Bajcsy, P., Kooner, J., Kotwani, K., Lee, J.S., Marini, L., Kooper, R., McLaren, T., and Liu, Y. (2009). Semantic middleware for e-science knowledge spaces. 7th International workshop on Middleware for Grids, Clouds, and e-Science (MGC 2009), Urbana Champaign, IL, Nov 30 – Dec 1 2009.

Fulton, R.A., Breidenbach, J.P., Seo, D.-J., and Miller, D. (1998). The WSR-88D rainfall algorithm. Weather and Forecasting, Vol. 13 (June), pp 377-395.

Gil, Y., E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers (2007), Examining the challenges of scientific workflows, *Computer*, *40*(12), 24-32, doi: 10.1109/MC.2007.421.

Granell, C., Diaz, L., and Gould, M. (2009). Service-oriented applications for environmental models: Reusable geospatial services. Environmental Modelling and Software, 25(2), 182-198.

Grassotti, C., Hoffman, R.N., Vivoni, E.R., and Entekhabi, D. (2003). Multiple-timescale intercomparison of two radar products and raingauge observations over the Arkansas-Red River Basin. Weather and Forecasting, 1207 – 1229.

Habib, E., Ciach, G.J., and Krajewski, W.F. (2004). A method for filtering out raingague representativeness errors from the verification distributions of radar and raingauge rainfall. Advances in Water Resources, 27, 967-980.

Havlik, D., Bleier, T., and Schimak, G. (2009). Sharing Sensor Data with SensorSA and Cascading Sensor Observation Service. Sensors, 2009, 9, 5493-5502. doi: 10.3390/s90705493.

Henschke, A., Habib, E., and Pathak, C. (2009). Evaluation of the radar rain Z-R relationship for real-time use in south Florida. In Proceedings of the 2009 World Environmental & Water Resources Congress, Kansas City, MO, May 17-21, 2009.

Horsburgh, J.S., Tarboton, D.G., Piasecki, M., Maidment, D.R., Zaslavsky, I., Valentine, D., Whitenack, T., (2009). An integrated system for publishing environmental observations data. Environmental Modelling and Software, 24(9), 879-888.

Jarvis, C.H., and Stewart, N. (2001a). A comparision among strategies for interpolating maximum and minimum daily air temperatures. Part I: The selection of "guiding" topographic and land cover variables. Journal of Applied Meteorology, 40, 1060-1074.

Jarvis, C.H., and Stewart, N. (2001a). A comparision among strategies for interpolating maximum and minimum daily air temperatures. Part II: The interaction between number of guiding variables and the type of interpolation method. Journal of Applied Meteorology, 40, 1075-1084.

Jayasumana, A.P., Qi, H., and Illangasekare, T.H. (2007). Virtual sensor networks – A resource efficient approach for concurrent applications. Proceedings of the 4th International Conference on Information Technology: New Generations, ITNG 2007, pp. 111-115.

Jolly, W.M., Graham, J.M., Michaelis, A., Nemani, R., Running, S.W. (2005). A flexible integrated system for generating meteorological surfaces derived from point sources across multiple geographic scales. Environmental Modelling and Software 20, 873-882.

Kabadayi, S., Pridgen, A., and Julien, C. (2006). Virtual sensors: Abstracting data from physical sensors. Proceedings of the 2006 International Symposium on a World of Wireless, Mobile, and Multimedia Networks, WoWMoM, pp. 587-592.

Klazura, G.E. and Imy, D.A. (1993). A description of the initial set of analysis products available from the NEXRAD WSR-88D system. Bulletin of the American Meteorological Society, 1293-1311.

Krajewski ,W. F., Kruger, A., Smith, J. A., Lawrence, R., Goska, R., Domaszczynski, P., Gunyon, C., Seo, B. C., Baeck, M. L., Bradley, A. A., Ramamurthy, M. K., Weber, W. J., Delgreco, S. A., Nelson, B., Ansari, S., Murthy, M., Dhutia, D., Steiner, M., Ntelekos, A. A., Villarini, G. (2008) Hydro-NEXRAD: community resource for use of radar-rainfall data, CUAHSI CyberSeminar April 25, 2008. Available at http://www.cuahsi.org/cyberseminars/Krajewski-20080425.pdf

Kunze, J. (2003). Towards electronic persistence using ARK identifiers. Proceedings of the 3rd ECDL Workshop on Web Archives, Trondheim, Norway, August 2003. Available at https://confluence.ucop.edu/download/attachments/16744455/arkcdl.pdf.

Liu, Y., D. J. Hill, A. Rodriguez, L. Marini, R. Kooper, J. Futrelle, B. Minsker, J. D. Myers (2008), Near-Real-Time Precipitation Virtual Sensor based on NEXRAD Data, ACM GIS 08, November 5-7, 2008, Irvine, CA, USA

Liu, Y., Hill, D., Marini. L., Kooper, R., Rodriguez, A., Myers, J. (2009a). Web 2.0 Geospatial Visual Analytics for Improved Urban Flooding Situational Awareness and Assessment, ACM GIS '09 , November 4-6, 2009. Seattle, WA, USA

Liu, Y., Wu, X., Hill, D., Rodriguez, A., Marini, L., Kooper, R., Myers, J., and Minsker, B. (2009b) A new framework for on-demand virtualization, repurposing, and fusion of heterogeneous sensors. Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems, pp. 54-63. doi 10.1109/CTS.2009.5067462.

Liu, Y., Futrelle, J.; Myers, J.; Rodriguez, A.; Kooper, R. (2010). A provenance-aware virtual sensor system using the Open Provenance Model, 2010 International

Symposium on Collaborative Technologies and Systems (CTS), pp.330-339, 17-21 May 2010, doi: 10.1109/CTS.2010.5478496

Ludäscher, B. (2009), What makes scientific workflows scientific? *Lect. Notes Comput. Sci., 5566 LNCS*, 217, doi: 10.1007/978-3-642-02279-1_16.

Marini, L., Kooper, R., Bajcsy, P., and Myers, J. (2007). Supporting exploration and collaboration in scientific workflow systems. EOS Transactions of the AGU 88(52), Fall Meeting Supplement, Abstract IN31C-07.

Moreau, L., L., Groth, P., Miles, S., Vazquez-Salceda, J., Ibbotson, J., Jiang, S., Munrow, S., Rana, O., Schreiber, A., Tan, V., Varga, L. (2008), The provenance of electronic data, *Commun ACM, 51*(4), 52-58, doi: 10.1145/1330311.1330323.

Morin, E., Krajewski, W.F., Goodrich, D.C., Gao, X., and Sorooshian, S. (2003). Estimating rainfall intensities from weather radar data: The scale-dependency problem. Journal of Hydrometeorology, 4, 783-797.

NIMA (National Imagery and Mapping Agency) (2000). Department of Defense World Geodetic System 1984: Its definition and relationships with local geodetic systems. Technical Report TR8350.2, Third Edition, Amendment 1.

NRC (National Research Council) (1999a) Adequacy of Climate Observing Systems. National Academy Press: Washington, D.C.

NRC (National Research Council) (1999b) Hydrologic Science Priorities for the U.S. Global Change Research Program. National Academy Press: Washington, D.C.

NRC (National Research Council) (1999c) Enhancing Access to NEXRAD Data- A Critical National Resource. National Academy Press: Washington, D.C.

NRC (National Research Council) (2008). Integrating Multiscale Observations of U.S. Waters. National Academy Press: Washington, D.C.

NRC (National Research Council) (2009). Observing Weather and Climate from the Ground Up: A Nationwide Network of Networks. National Academy Press: Washington, D.C.

NSF (National Science Foundation) (2004). Sensors for Environmental Observatories: Report of the NSF-Sponsored Workshop, December 2004. Arlington, VA: NSF.

NWS-ROC (National Weather Service Radar Operating Center) (2003), Operator Handbook Guidance on Adaptable Parameters Doppler Meteorological Radar WSR-

88D Handbook, Vol. 4, RPG. National Weather Service Radar Operations Center: Norman, Oklahoma.

NWS-ROC (National Weather Service Radar Operating Center) (2008a). Interface Control Document for the Archive II/User. National Weather Service Radar Operations Center: Norman, Oklahoma.

NWS-ROC (National Weather Service Radar Operating Center) (2008b). Interface Control Document for the RDA/RPG. National Weather Service Radar Operations Center: Norman, Oklahoma.

Rodriguez, A., McGrath, R.E., Liu, Y. and Myers, J.D. (2009). Semantic Management of Streaming Data, 2nd International Workshop on Semantic Sensor Networks at the International Semantic Web Conference, Washington, DC, October 25-29, 2009

Rouault, P., Schroeder, K., Pawlowsky-Reusing, E., and Reimer, E. (2008). Consideration of online rainfall measurements and nowcasting for RTC of the combined sewage system, Water Science and Technology, Vol. 57, No. 11, pp 1799-1804.

Sahoo, S.S., Sheth, A., Henson, C. (2008). Semantic provenance for eScience – Managing the deluge of scientific data. IEEE Internet Computing 12(4), 46-54.

Sempere-Torres, D., Corral, C., Raso, J., and Malgrat, P. (1999). Use of weather radar for combined sewer overflows monitoring and control. Journal of Environmental Engineering, 125(4), 372–380.

Seo, D.-J., Breidenbach, J.P., and Johnson, E.R. (1999) Real-time estimation of mean field bias in radar rainfall data. *Journal of Hydrology* 223, 131-147.

Sharif, H.O., Yates, D., Roberts, R., and Mueller, C. (2006). The use of an automated nowcasting system to forecast flash floods in an urban watershed. Journal of Hydrometeorology, 7, 190-202.

Smith, J.A. and Krajewski, W.F. (1991). Estimation of the mean field bias of radar rainfall estimates. *Journal of Applied Meteorology*, 30, 397-412.

Smith, J.A. and Krajewski, W.F. (1993). A modeling study of rainfall rate-reflectivity relationships. *Water Resources Research* 29, 2505-2514.

Smith, J.A., Baeck, M.L., Meirdiercks, K.L., Miller, A.J., and Krajewski, W.F. (2007). Radar rainfall estimation for flash flood forecasting in small urban watersheds. Advances in Water Resources, Vol. 30, pp 2087-2097.

Smith, J.A., Seo, D.J., Baeck, M.L., and D. Hudlow (1996). An intercomparison study of NEXRAD precipitation estimates. Water Resources Research, 32, 7, 2035 – 2045.

Steiner, M., and Smith, J.A. (1998). Convective versus stratiform rainfall: An ice-mircophysical and kinematic conceptual model. *Atmospheric Research* 47-48, 317-326.

Steiner, M., Smith, J.A., Burges, S.J., Alonso, C.V., and Darden, R.W. (1999). Effect of bias adjustment and rain gauge data quality control on radar rainfall estimation. *Water Resources Research,* 36(8) 2487-2503.

Thorndahl, S., Beven, K.J., Jensen, J.B., and Shaarup-Jensen, K. (2008). Event based uncertainty assessment in urban drainage modeling applying the GLUE methodology. Journal of Hydrology, 357, 421 – 437.

Tustison, B., Foufoula-Georgiou, E., and Harris, D. (2003). Scale-recursive estimation for multisensor Quantitative Precipitation Forecast verification: A preliminary assessment. Journal of Geophysical Research, 198(D8). doi: 10.1029/2001JD001073.

Tustison, B., Harris, D., and Foufoula-Georgiou, E. (2001). Scale issues in verification of precipitation forecasts. Journal of Geophysical Research, 106, 11, 775-11, 784.

Vieux, B.E. and Farafalla, N.S. 1996. Temporal and spatial aggregation of NEXRAD rainfall estimates on distributed storm runoff simulation. In Proceedings of The Third International Conference/Workshop on Integrating GIS and Environmental Modeling, National Center for Geographical Information and Analysis.

Vieux, B.E. and Vieux, J.E. (2005). Statistical evaluation of a radar rainfall system for sewer system management. Atmospheric Research 77, 322-336.

Wang, X., Xie, H., Sharif, H., and Zeitler, J. (2008), Validating NEXRAD MPE and Stage III precipitation products for uniform rainfall on the Upper Guadalupe River Basin of the Texas Hill Country, Journal of Hydrology, 348, 73-86.

Xiao, R. and Chandrasekar, V. (1997). Development of a neural network based algorithm for rainfall estimation from radar observations. IEEE Transactions on Geoscience and Remote Sensing 35 (1997), pp. 160–171.

Zhu,J. (1004). Conversion of Earth-centered Earth-fixed coordinates to geodetic coordinates. IEEE Transactions on Aerospace and Electronic Systems, vol. 30, pp. 957-961.

Figure 1: Hierarchy of data products produced by the National Weather Service from WSR-88D weather radar.

Figure 2: Layered Architecture of the Virtual Sensor System

Figure 3: Provenance graph for point-based rainfall virtual sensor. Arcs indicate relationships between entities within the graph.

Figure 4: Screen shot of Cyberintegrator GUI showing Chicago sewershed averaged rainfall virtual sensor.

Figure 5: Sewershed-averaged rainfall rates during a rain event in the Chicago study area. The unit of the rainfall rates is mm/hr.

Figure 6: Illustration of the PointVS virtual rainfall sensor.

Figure 7: Illustration of the PolyVS virtual sensor .

Figure 8: Illustration of polygon averaging. The Cartesian grid cells marked with a dot will be averged to calculate the polygon-averaged rainfall rate.

Figure 9 Location of five tipping bucket raingauges and WSR-88D weather radar station (KLOT) in study region.

Figure 10: Comparison of radar and raingauge observation of rainfall accumulation for 30 hour period beginning on June 4, 2007, at 16:00 UTC. Gauges A-E cover an area of approximately 55 square miles.

Figure 11: Comparison of radar and raingauge observation of rainfall accumulation for 8 hour period beginning on August 23, 2007, at 18:00 UTC. Gauge B is

malfunctioning during this time period and the radar goes off-line around 23:40 UTC. Gauges A-E cover an area of approximately 55 square miles.