Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 3159-0040
Vol. 2 Issue 6, June - 2015

A Sensor Fusion Architecture For Human-Centric Sensing Applications

Sanjay K. Boddhu
Synthos Technologies, A division of Qbase LLC
Dayton, Ohio, USA
sboddhu@qbase.com

Robert L. Williams
Discovery Labs, U.S. Air Force Research Labs
Wright Patterson AFB, Ohio, USA
robert.williams@wpafb.af.mil

Xiao-Guang Yue
School Resources and Environmental Engineering,
Wuhan University of Technology
Wuhan, China
xgyue@whut.edu.cn

***Abstract*—The recent advances in smart devices technology and their profuse availability have made the prospective of human-centric sensing and computing paradigms a viable reality. There already exist various operational intelligent systems in different domains like defense, healthcare, energy and disaster management that have been developed by employing human-centric sensing as their backbone. But, to support building more complex or novel human-centric based systems that have to integrate with existing sensors/devices and possible future sensors, there exists practical issue like accommodating disparate data formats, modality and connectivity interfaces. These low-level issues make integration of different sensing devices and fusion of sensed data a challenge and time consuming process, delaying the high-level implications, which can be targeted towards solving real world problems. In this paper, a generic sensor fusion architecture has been presented that has been developed to solve the mentioned challenges and support seamless integration and development of human-centric sensing devices and also platforms.**

Keywords—Situational Awareness, Human Centric Sensing, Situational Understanding, Sensor Fusion Architecture

I. INTRODUCTION

Human-centric sensing (HCS) provides a new perspective and understanding to the real world sensing problems like situational awareness, environmental monitoring, health care, over various traditional sensing processes, by collecting grass-root level data thru the smart devices carried by humans. These HCS can be participatory or opportunistic in nature, but the genuine value of the human centric sensing platform can be possible only thru a real-time forensic data processing with near real-time collaborative human validation that also leverages archived information for a given sensing task at hand. The real-time collaborative constraint on the human centric sensing requires deployment of a scalable and adaptive communications framework that can provide a composable fusion of standard internet and hybrid intranet communication networks.

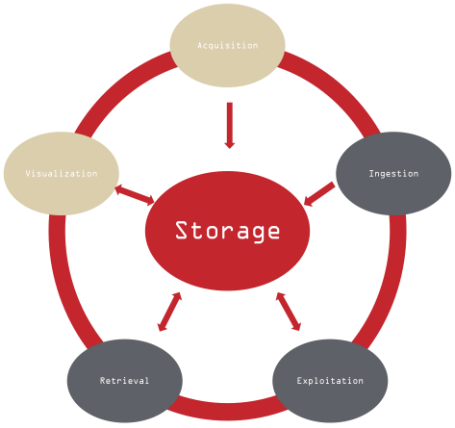

Fig. 1: Conceptual view of the Persistent Sensor Storage Architecture

Further, as one can perceive, the type of data collected thru smart devices can vary dramatically from simple unstructured data (text, image, video, etc…) powered thru social media networks or complex multi modal data (like heart rate, ECG, Pulse rate, CO2 levels, etc…) powered by custom developed connectivity frameworks, based on the situational analysis and prediction tasks in a given domain, and this data can grow astronomically in certain applications. Thus, the smart devices participating in the HCS systems would require a unified interface that is agnostic to their modality, collection rate and connectivity and provide seamless mechanisms to amass massive amounts raw data from the smart devices, process the data in real-time and disseminate the data to all the involved stack holders in a collaborative fashion. The data storage and forensic data mining has to reside in a secured internet/intranet or hybrid environments, as currently the storage and computing on massive data is not practical on hand held smart devices. There already exist various successful HCS based applications that are custom and proprietary built for specific vision and mission [6] [7] but building a generic architecture that can provide a platform to seamlessly develop and deploy HCS based applications is still an active research area that has to address the aforementioned

challenges. In recent years, the researchers at AFRL Discovery Lab have been working on developing a scalable and "plug-n-play" generic architecture called Persistent Surveillance and Storage Architecture (PSSA), that can normalize disparate sensor data and provide a unified and seamless sensor data fusion, processing and storage capabilities for persistence surveillance and situational awareness problems related to defense and homeland security domains. Recently, the efforts to extend the architecture with smart devices and smart phone integration capabilities over heterogeneous networks employing the core components of the architecture as yielded in promising implications to serve as a generic architecture for Human Centric Sensing applications.

In this vein, this paper provides the logical and conceptual views of PSSA, followed by PSSA SDK and PSSA's scalability implications towards building HCS applications. Finally the discussion focusing on the brief description of successfully developed and deployed HCS applications will be presented.

II. PERSISTANCE SURVEILLANCE AND STORAGE ARCHITECTURE

A. *Conceptual View*

The PSSA supports persistent surveillance and human centric sensing activities by providing a platform for storing, processing, and disseminating data captured via remote sensors. As depicted in Fig. 1, storage is central to our architecture. The storage subsystem provides a scalable, high performance, fault- tolerant and flexible repository for all of the sensor data ingested into the system. The ring around the storage subsystem represents a messaging capability that allows sensor data to be simultaneously communicated synchronously or asynchronously to all of the other services surrounding the storage subsystem.

Services can be "plugged" into this event-based backbone (or cloud) as needed to support new sensors, smart devices and respective sensor processing without having to rebuild or even shut down the system. Services defined as part of the PSSA provide the means for sensor data to be ingested, exploited, stored and retrieved. Although storage is central to the PSSA, the other services are optional and can be plugged into or removed from the PSSA system without affecting any other service or requiring shutdown or restart of the system. Acquisition and Visualization Services can be developed for specific sensor applications and connected directly to the PSSA cloud and communicate with the system via standard or customized Ingestion and Retrieval services.

B. *Logical View*

From a logical perspective, the PSSA consists of four customizable software layers: Ingestion Services, Application Services, Data Services, and Dissemination Services as shown in Fig. 2. In addition, the PSSA includes a common Core Services layer which is responsible for providing the infrastructure required to support the integration of the components in the customizable software layers. The PSSA SDK provides access to the services provided by the Core Services layer. The diagram below provides a logical view of these software layers and examples of some of the components that might be present in these software layers.

Although the Fig. 2, only shows four types of sensor sources, the PSSA is designed to support the storage and exploitation of data from any sensor source by "plugging" in new ingestion services. Similarly, the PSSA is designed to support the distribution of sensor data and related exploitation data to any visualization platform by "plugging" in standards based and custom built dissemination services.

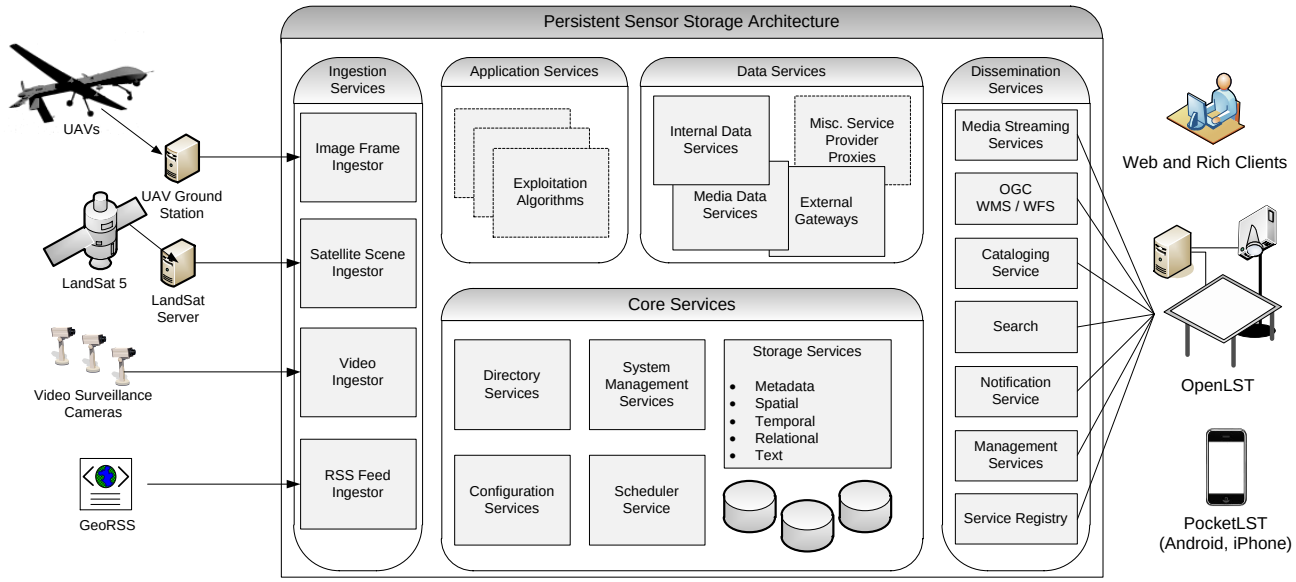

Fig. 2: Logical view of the Persistent Sensor Storage Architecture

The PSSA is designed to allow components in the customizable software layers to be added or removed from the system, dynamically, without interrupting the operation of other components of the system. This is accomplished through the use of a message bus approach. The PSSA message bus provides the following messaging modes:

Publish/Subscribe: Source components publish messages on the PSSA message bus as “topics”. Sink components subscribe to “topics” to receive messages in which they are interested. Filters can be applied by sink components to reduce the number of messages received by the component. Any number of sink components can subscribe to the same “topic”. This is the typical communication mode for pluggable components as it allows them to be completely decoupled from one another.

Request/Response: A sink component issues a data request to a source component and the source component sends its response back to the sink component. This process is typically how components communicate with core services that are guaranteed to be present and external gateways that are used to retrieve data from external systems.

Stream Request/Response: A sink component requests streaming data from a source component. The source component sends the requested data stream to the sink component until the request is satisfied or the sink component tells the source component to stop streaming. This process is typically used by visualization components that want to display live or recorded streaming media.

By decoupling the components from each other using the message bus architecture, four important goals are achieved:

1. Operators and developers can add and remove components from the system dynamically without impacting other components of the system (“plug and play”)

2. Data is “broadcast” to several components at the same time allowing processing of the data to take place in parallel

3. Processing is distributed across multiple compute nodes providing virtually unlimited horizontal scaling of the system

4. Sensor data becomes available for visualization as soon as it is ingested by the system, without waiting for it to be stored and indexed

The Fig. 3 summarizes the key components and types of services that comprise the PSSA along with the typical flow of data into and out of the PSSA messaging system which we refer to as the PSSA or PSSA Cloud (represented by the cloud in the center of the Fig. 3). The reason we refer to the PSSA messaging system as the “Cloud” is because it hides the internal implementation of the core services and integration mechanisms from the user developed components of the system. Developers of PSSA components need only know the public interfaces exposed by the PSSA SDK in order to participate in the event collaboration model of integration used by a PSSA-based system.

A brief description of each of the software layers and their functions is provided below:

Ingestion Services – Receive or retrieve sensor data from external sensors and bring the data (sensor readings and associated metadata) into the PSS system. The main responsibilities of these components are to communicate with the external sensors or sensor systems and to make the data available to other components of the system, via the PSSA message bus. The PSSA reference implementation provides several different types of ingestion components that can be used as examples for creating custom ingestion components.

Application Services – Exploit one or more sensor data stream or exploitation data stream and, optionally, external data sources to turn the sensed data into useful information (e.g. identify anomalies or other data of interest, analyze the quality of the data, improve the quality of the data, etc.). These components do not need to know how to communicate with the sensors, but they do need to understand the semantics of the data generated by the sensors in order to exploit it. Application service components implement algorithms to exploit data to which it is subscribed on the PSSA message bus. Results of the exploitation processing are published to the message bus and are thus available to be stored, further exploited or displayed using a visualization tool. The PSSA reference implementation includes several examples of application service components that can be used as examples for creating new components using the PSSA SDK.

Data Services – Store and retrieve data from the PSSA internal data stores and/or external systems for use by other components of the system. A data service component encapsulates the access to data by providing high-level message based interfaces to the

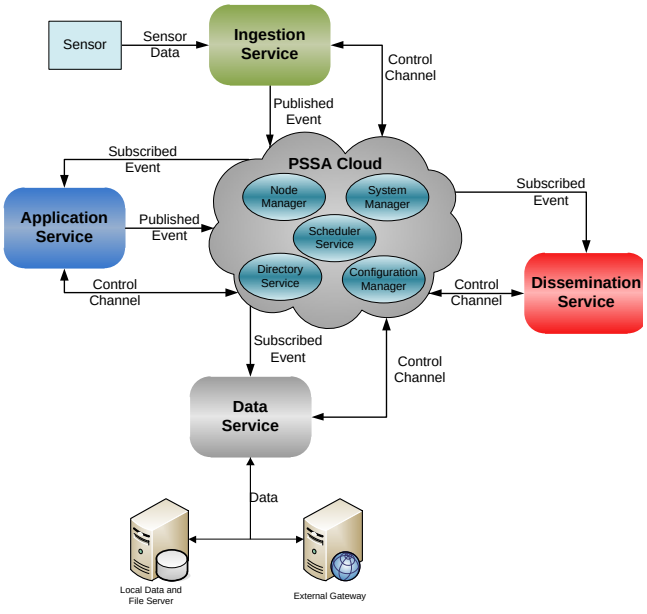

Fig. 3: Key components of the Persistent Sensor Storage Architecture

```
{
   "sensorID": "camer001",
   "ipAddress": "172.04.13.15",
   "sensorDescription": "Camera onboard patrol vehicel",
   "location": {
      "lat": 38.898556,
      "long": -77.037852,
      "datum": "WGS84"
   },
   "topic": "videoImage",
   "time": "2016-04-21T20: 00",
   "timeZone": "AST",
   "eventData": {
      "type": "encodedImage",
      "data":
"R0lGODlhUAAPAKIAAAsLav///88PD9WqsYmApmZ
mZtZfYmdakyH5BAQUAP8ALAAAAAABQAA8AAAPb
WLrc/jDKSVe4OOvNu/9gqARDSRBHegyGMahqO4R0
bQcjIQ8E4BMCQc930JluyGRmdAAcdiigMLVrApTYW
y5FKM1IQe+Mp+L4rphz+qIOBAUYeCY4p2tGrJZeH9
y79mZsawFoaIRxF3JyiYxuHiMGb5KTkpFvZj4ZbYeCi
XaOiKBwnxh4fnt9e3ktgZyHhrChinONs3cFAShFF2Jhv
CZlG5uchYNun5eedRxMAF15XEFRXgZWWdciuM8G
CmdSQ84lLQfY5R14wDB5Lyon4ubwS7jx9NcV9/j5+g4J
ADs=",
      "height": 15,
      "width": 80,
      "encoding": "base64"
   }
}
```


Fig. 4: Event schema for the mongoDB document, which can support spatial and temporal queries, filtered by the event topics for better event aggregation and situational understanding.

data. This allows the consumer of the data to focus on how to use the data rather than on how to get the data. It is the responsibility of the Data service component to fulfill the data request of the other component making the data request. Data service components hide the details of communicating with internal and external databases and with other sensor storage systems from PSSA components that require the data to do their processing. The PSSA message interface to retrieve data from a data service component will typically be a request/response message. For storing data, data service components subscribe to the message topic for which they are responsible and perform whatever processing is required to store the data. As a reference implementation, mongodb [8] is employed with sharded cluster architecture for horizontal scalability to handle massive amounts of sensor event data. Further, a low-level event document schema has been built to transform the messages published to the PSSA cloud for archival purposes. A sample reference event schema of the document is provided in Fig. 4. The data services supports range queries that can span across spatial and temporal context of the archived events, with filters across the topics of the events. This feature and provide aftermath analysis of any events that can contribute to better understanding of the scenarios in the context and thus better situational understanding.

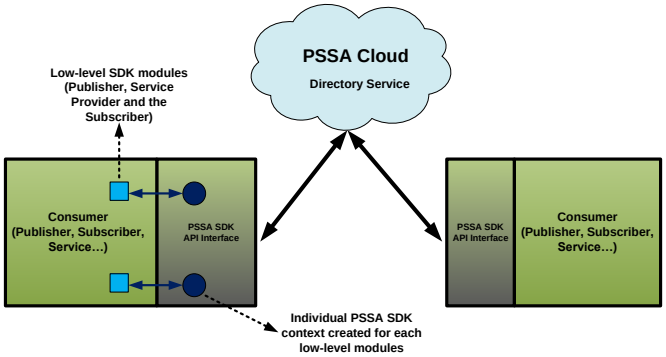

Fig. 5: Pictorial representation of PSSA SDK Interface

Dissemination Services – Provide standard and custom built interfaces to provide access to data ingested by and/or stored within the PSSA system to users or external systems. A PSSA dissemination component can either push the data to a remote system or wait for the remote system to request the data. The PSSA reference implementation includes several examples of PSSA dissemination components use the PSSA message bus to subscribe to topics generated by other components of the system. Typically, the interfaces provided by these components utilize a synchronous request/response message mode to send data from the PSSA based system to other systems. However, some dissemination components support the retrieval of live data as well as previously stored data using the stream request/response message mode. In addition, a dissemination component can be developed to provide event notifications or alerts asynchronously when certain events occur.

Core Services – Provide the infrastructure required to connect the other components of the system together and to store the data ingested by the system. This infrastructure includes a directory service that allows sensor processing components to be dynamically added to and removed from the system; a scheduling service to ensure optimal use of the computing resources by load balancing and sequencing tasks; a message bus for moving data between components of the system; a configuration and management service to allow processing components and compute nodes to be added and removed from the system; and a storage service to provide storage, indexing, and association of data required by the system.

III. PSSA SDK & EVENT FORMAT

A. PSSA SDK

As part of the effort to design and develop the PSSA reference implementation to ingest, exploit, store and visualize disparate real-time sensor data, a software development kit (SDK) has been developed in C++ language, using a light-weight messaging middleware library called ZeroMQ. This developed PSS SDK aptly provides the required API interfaces for any consumer of the PSSA architecture, to build low-

level PSSA modules like Publisher, Subscriber and Service Provider. These low-level PSSA modules can be grouped (according to the custom business logic) to build high level customized components like sensor data Ingestor, Exploitation service, Data Storage service and any helper services to interact with third party services (on heterogeneous networks) by bridging it to PSSA instance. Further, the PSSA SDK has been design to effectively abstract the architecture and communication (along with few media processing routines) complexity from the consumer and aid him/her in building the high-level business logic components that can effectively leverage the PSSA framework.

The PSSA SDK was originally developed in C++, with pertinent classes and structures, later wrapped with C "DLL" procedures to make it cross-language compatible. Further, the JAVA and C# bindings/wrappers are also available for the present version of the PSSA SDK. The PSSA SDK has proven to be beneficial to develop PSSA components in JAVA, MATLAB and C# languages for various capability demonstrations (discussed in this report). This sub-section will provide the functional description of the APIs that are available as part of the C-DLL interfaces which reasonably extrapolates into the other language bindings. Fig. 5 is a pictorial representation of employing PSSA SDK API interfaces to build customized (Business Logic) components. The developed PSSA SDK provides the required API interfaces for any consumer of the PSSA Cloud architecture, to build low-level PSSA components like Publisher, Subscriber and Service Provider. These low-level components can be grouped (for building Exploitation algorithms that would require Subscriber and Publisher components in them) appropriately to build customized components like mobile Ingestor, Video ingestor, Media Storage Service and other Exploitation Algorithm Services (discussed in detail in; [5]).

B. PSSA Event Formats

For the reference implementation of the PSSA system, we defined the event message to be a multi-part mime that includes a header and one or more mime data segments to transmit the sensor data. The header is comprised of a context section that contains the common data described above and a content section that contains metadata associated with the subsequent mime data segments. This works very well within the context of our messaging system.

In brief, PSSA event message consists of two main elements in it; Event Data (data section) and Event Topic (context section). Event data can be any binary or xml encoded data packet that can be customized for given sensor and its processing and subscribed components and an event message can have multiple data parts in it to accommodate a multi-modal sensors. The same can be applied to the event topic (containing metadata of the real data) defined by the component developer(s) to meet the processing needs of the component. But, based on the prior evaluation and standards, common metadata elements included in the context section of all event topics that is required by the PSSA-based system to index, correlate and query data across disparate sensors in a common manner. This common metadata contained in the Event context header includes the following data elements:

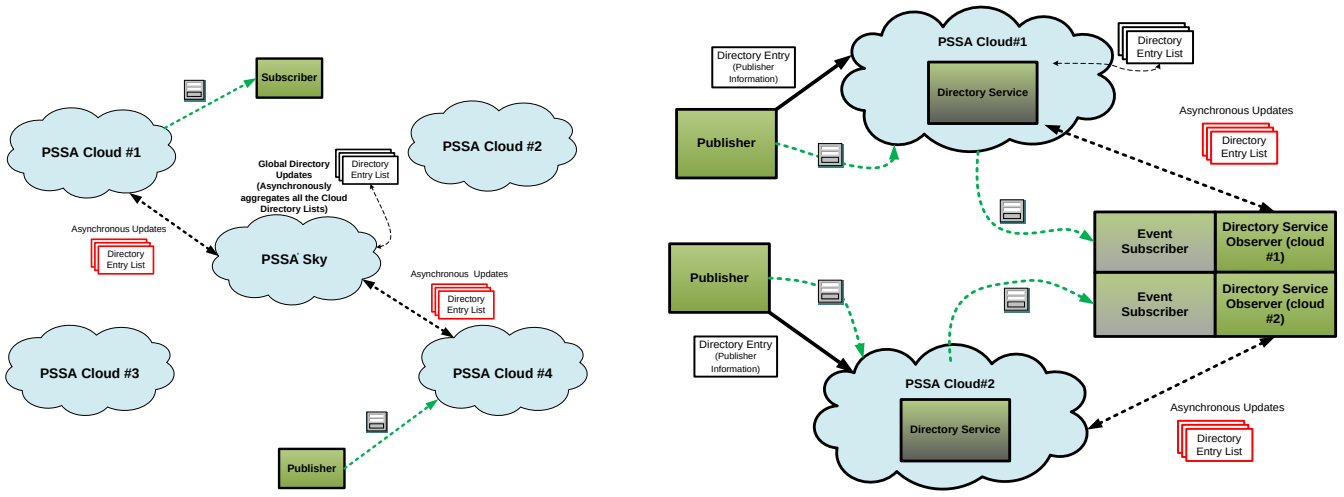

Fig. 6: PSSA sky Architecture Overview. (A)Integration of multiple PSSA clouds (B) Multi-cloud connectivity

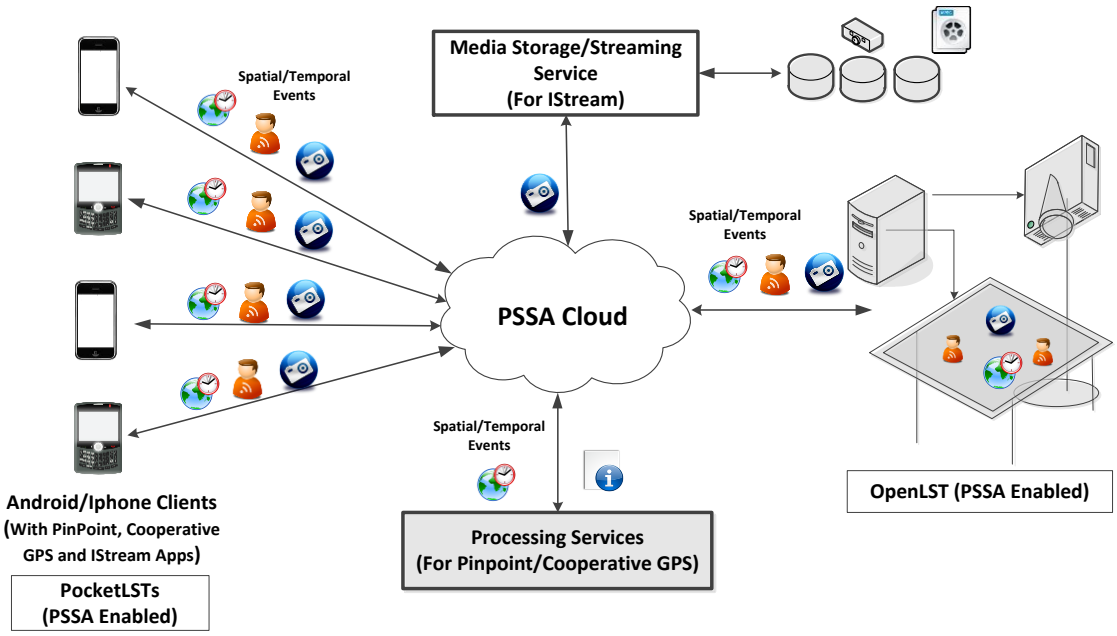

Fig. 7 : A conceptual view of a deployed PSSA cloud architecture along with processing services like PinPoint, Cooperative GPS and Media streaming services , subscribing and publishing spatio-temporal events like chat, information and media, from smartphone sensors, installed with PocketLST applications (I.E. PinPoint, Cooperative GPS, IStream and Guardian Angel apps). The PSSA enabled OpenLST and smartphone applications can register as dissemination and ingestion service simultaneously and collaborate among all the involved modules in cloud by communicating the live data and also the data generated by the services in near real-time fashion, thus enhancing the situational awareness and understanding of a given mission and demonstrating the feasibility of employing PSSA to support seamless development of Human Centric Sensing systems.

Producer: The unique identifier of the component generating the event. This data is automatically supplied in the context header by the PSSA SDK.

Reference Sensor ID: The unique identifier of the sensor from which the sensor reading or derived information originated. This could include multiple sensors if the payload of the event contains information derived from multiple sensors.

Presentation Time: The Universal Time Coordinated (UTC) time index of the data contained in the event payload. This is represented as a TimePoint structure which includes a point in time (in UTC) and the precision associated with the time (+/-) based on where the time originated. This is the time by which all of the data associated with the event will be indexed. The method of determining this time may be different for every ingestion component. In some cases the sensor may have a highly accurate time included with the metadata it generates for each reading. In other cases, the sensor may have no time data at all and the ingestion component must provide an educated guess as to the time the data was acquired based on the current system time.

Acquisition Time: The time that the data contained in the event was received or generated by the system. Ingestion components capture the time at which the sensor data was read/detected by the system (e.g. for a polling component, this is the time at which the request returned with the data, for an asynchronous component, this is the time at which the data was received, for a file detection, this is the time at which the new file was detected). Exploitation components that generate new data use the latest acquisition time contained within the events that were used to generate the new data (e.g. some exploitation events may require multiple frames of data to identify an object or they may use data from one or more different types of events). All other exploitation components leave the acquisition time unchanged.

Event Time: This is the time that the event was published by the component (automatically inserted by the PSSA SDK).

Processing Duration: This is the amount of time it took for the component to process the data to generate the event (automatically inserted by the PSSA SDK).

Sensor Location: This represents the (point) location at which the sensor was located when the sensor reading was captured. Commonly this will be provided as metadata with the sensor reading. If the sensor is attached to a non-moving platform, the location data may be part of the configuration data for the sensor. In either case, the event message must contain location information for indexing purposes in the database.

Sensor Location Accuracy (Optional): If known, this is included as part of the common event data. Downstream exploitation components may be able to improve the location accuracy or, if the accuracy is unknown, it may be possible to estimate the accuracy of the sensor location metadata using historical data or data from other sources. If this is the case, a new event will be published with the corrected location metadata and/or accuracy information.

Footprint (Optional): This represents the coverage area of the sensor. For some types of sensors this information may not be known at ingestion time and may be calculated by downstream exploitation algorithms. In other cases, the footprint might be calculated to some degree of accuracy by the sensor or sensor ground-station and included in the metadata received by the ingestion component.

Footprint Accuracy (Optional): If known, this is included as part of the common event data. Downstream exploitation components may be able to improve the footprint accuracy or, if the accuracy is unknown, it may be possible to estimate the accuracy of the sensor footprint metadata using historical data or data from other sources. If this is the case, a new event will be published with the corrected footprint metadata and/or accuracy information.

IV. PSSA SCALABILITY

A PSSA cloud instance is defined by its core services, which are intrinsically characterized by the "Plug & Play" capabilities. This flexible feature of the PSSA architecture facilities the possibility to further group multiple individual PSSA clouds in a higher level "Plug & Play" sky architecture shown in Fig. 6(A). This sky architecture is possible by extending the functionality of the currently existing directory service (one of the core services in a PSSA cloud instance) to register itself (when it becomes available) with a global directory service (which defines the sky instance) and also asynchronously push its local directory updates of publishers and services (registered with it) into the sky's directory services. Multiple PSSA clouds be grouped to build a PSSA sky architecture, which allows the possibility to make an Ingestion Service (i.e. Publisher) information and events from one instance of the cloud in the sky, available to the exploitation service (i.e. Subscriber) from another cloud in the same sky.

Additionally, the Sky's global directory service is responsible to push all the aggregated updates (from all the clouds) back to all the clouds (which will have duplicate elimination process in it) present in the Sky. Thus, an exploitation service registered with one of the clouds in the sky would have information to subscribe to the PSSA sensor events that are being published in another instance of the cloud in the same sky. Further, from implementation perspective the PSSA SDK provides API calls to build the low-level communication modules (like Publisher, Subscriber, Directory Service Observer and the service provider) which are independent of each other in any given PSSA component (a PSSA-based application). Hence, it is also possible to build complex Multi-Cloud PSSA components that can be part of more than one PSSA cloud at any given time, like as shown in Fig. 6(B). By leveraging the flexibility of the PSSA and using current PSSA SDK, it is possible to build complex Multi-Cloud PSSA components that can be part of more than one cloud. This flexibility makes PSSA architecture an ideal candidate to employ in developing and deploying Human Centric Sensing Applications.

V. PSSA ENABLED HCS APLLICATIONS & DISCUSSION

In last two years, the collaborative teams at AFRL discovery lab undertake research projects that were aimed at evaluating two interesting concepts in enhancing the situational awareness that are directly related to Human Centric Sensing;(1) To employ smartphone's display capabilities to effectively communicate information from command center to the personnel in the field and (2) To employ smartphone's existing sensors or further enhance them with external sensors to build human augmented sensors in the field (I.E. enabling the personnel/soldier/first responder as a sensor, communicating the data back to the command center) . As part this effort PSSA mobile engine service was developed based on robust presence protocol that can span across varied forms of connectivity, (internet, intranet, Bluetooth, cellular) that can manage dynamic registration , ingestion and subscription of smartphone devices as virtual sensors with multi-modal data capture capabilities. This developed mobile engine complies with PSSA conceptual view and provides "plug-n-play" capabilities [1] [2] [5]. Further, the OpenLST (Open Layered Sensing Test bed) is a dissemination platform developed to provide visualization of various in-field and experimental sensors being developed at the facility.

These PSSA enabled smartphone real-time collaborative smartphone applications called PocketLSTs (a mobile version of OpenLST), have been field tested and the real-time operational integration of these smartphone sensors with the PSSA could architecture demonstrated that the achievability of above two Human Centric Sensing concepts has indeed enhanced the overall success of a mission in the context [1] [2]. Continuing to demonstrate the effectiveness of these two concepts, research teams have employed smartphones and built various HCS based smart applications, to tackle real-world scenarios, with PSSA as the event centric backbone architecture, as pictorial represented in Fig. 7. Some of the major projects included; (1) PinPoint- A collaborative smart application that uses a network of three smartphones and their audio sensors to perform sound localization., (2) IStream- A smart application that uses the camera sensor on the smartphone and streams the camera feed to a localized video server for archival and real-time streaming to Layered Sensing Visualization platforms (like OpenLST command center)., (3) Cooperative GPS- A collaborative smart application that uses the GPS sensor on the smartphone along with external sensor hardware to accurately determine a more precise GPS location [4] .,(4) Guardian Angel - A collaborative smart

application that uses the GPS, Accelerometer, camera sensors on the smartphone to accurately monitor and predict path of a hostile or unidentified drone in a given air space [3] .

Thus, because of the flexibility introduced in the PSSA cloud design, an instance of the PSSA cloud can be deployed with minimal resources using commodity machines or can be extended over multiple enterprise grade servers, depending on the data size and processing requirements. This flexibility along with unified and customizable data normalization provision makes PSSA an ideal option for developing and deploying Human Centric Sensing applications.

REFERENCES

[1] Boddhu S. K. (2013). "Increasing Situational Awareness Using Smartphones". Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR III, SPIE's Defense, Security and Sensing Conference.

[2] Boddhu, S. K., McCartney, M., Ceccopieri, O., & Williams,R. L. (2013, May). A collaborative smartphone sensing platform for detecting and tracking hostile drones. In SPIE Defense, Security, and Sensing (pp. 874211-874211). International Society for Optics and Photonics.

[3] Boddhu, S. K., Dave, R. P., McCartney, M., West, J. A., & Williams, R. L. (2013, May). Context-aware event detection smartphone application for first responders. In SPIE Defense, Security, and Sensing (pp. 874213-874213). International Society for Optics and Photonics.

[4] Feitshans, T. (2013). "Smartphones for Distributed Multi-Mode Sensing: Biological & Environmental Sensing and Analysis". Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IV, SPIE's Defense, Security and Sensing Conference.

[5] Wasser E., Boddhu S. K., Kode N., Berkey T. Analyst Performance Measures, Volume I: Persistent Surveillance Data Processing, Storage and Retrieval, Final contract report, AFRL-RH-WP-TR-2012-0094, available at www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA567802.

[6] D. Lymberopoulos, A. Bamis and A. Savvides (2013), "Extracting Spatiotemporal Human Activity Patterns in Assisted Living using a Home Sensor Network", UAIS 2011.

[7] D. De, S. Tang, W.-Z. Song, D. J. Cook, S. K. Das (2012), "ActiSen: Activity-aware Sensor Network in Smart Environments, Pervasive and Mobile Computing", Vol. 8, No. 5, pp. 711-731, Oct 2012.

[8] Membrey, Peter, Eelco Plugge, and Tim Hawkins. The definitive guide to MongoDB: the noSQL database for cloud and desktop computing. Apress,2010.