EAI Endorsed Transactions
on Context-aware Systems and Applications
Research Article EAI.EU

AndroCon: An Android-Based Context-Aware Middleware Framework

Jian Yu[1], Quan Z. Sheng[2], Olayinka Adeleye[1] and Chris Wang[1]

[1]Department of Computer Science,Auckland University of Technology, Auckland, 1010, New Zealand
[2]Department of Computing, Faculty of Science and Engineering Macquarie University Sydney, NSW 2109, Australia

Abstract

Mobile devices have become major sources of context-aware data due to their ubiquity and sensing capabilities. However, deploying mobile devices as dynamic, unabridged context data provider either locally or remotely is challenging, due to their limited computing capabilities. Moreover, mobile sensors are limited to physical context data acquisition and there is a need to integrate physical data provided by these sensors with social context data provided by various mobile applications. Such data integration is necessary in order to have a robust data sources for various context-aware applications. In this paper, we present AndroCon, an Android-based, context-aware middleware framework that enables mobile devices to acquire, integrate, manage context data and to provision the data to applications both locally and remotely. AndroCon enables integration of both raw physical and social related context data. Instances of AndroCon have been achieved by interpreting and storing high-level context knowledge locally and utilizing web service technologies for data provisioning. We perform extensive experiments using AndroCon to collect, provision and manage both social and physical context data from different sources. We have also analyzed AndroCon's performances based on its power consumption and CPU utilization.

Received on 10 August 2017; accepted on 03 December 2017; published on 14 March 2018

Keywords: Data Provisioning, Context-Awareness, Integration, Middleware, Android

doi:10.4108/XX.X.X.XX

1. INTRODUCTION

Over the years, the computing and sensing capacity of mobile devices has been constantly improving to support smart environment and robust applications. They have embedded capabilities to sense and capture events in their physical environments, thus, making them a popular source of context information. Smart devices such as Android Smart phones now incorporate high-level mobile application interaction that provides enormous social related data. These innovations together with the proliferation of pervasive devices have encouraged the development of more and more mobile-based, context-aware systems. While the volume of context data provided through various mobile applications and sensors are on rapid increase, collecting these data from various platforms and integrating them for efficient context-awareness processing are challenging. Context acquisition, processing and efficient utilization of context resources have been the major bottlenecks in developing context-aware systems. However, mobile devices face even more complexity due to their resource limitation when compared to high performance servers. They require more power, bandwidth, memory and storage to function independently as context provider in a robust context-aware architecture [1]. They rely on high-performance servers to function in such environment. Existing research approaches suggested offloading computational resources such as context data from mobile devices to cloud-based infrastructures [2]. Although, such approach has been considered expensive, due to high-quality network connection required to facilitate data transfer to cloud resources, yet, mobile devices have not been considered as viable context provider even with the increasing improvement in their computational capabilities. Moreover, in order to have robust context database that captures all the required activities necessary to determine a particular

*Corresponding author. Email: jian.yu@aut.ac.nz

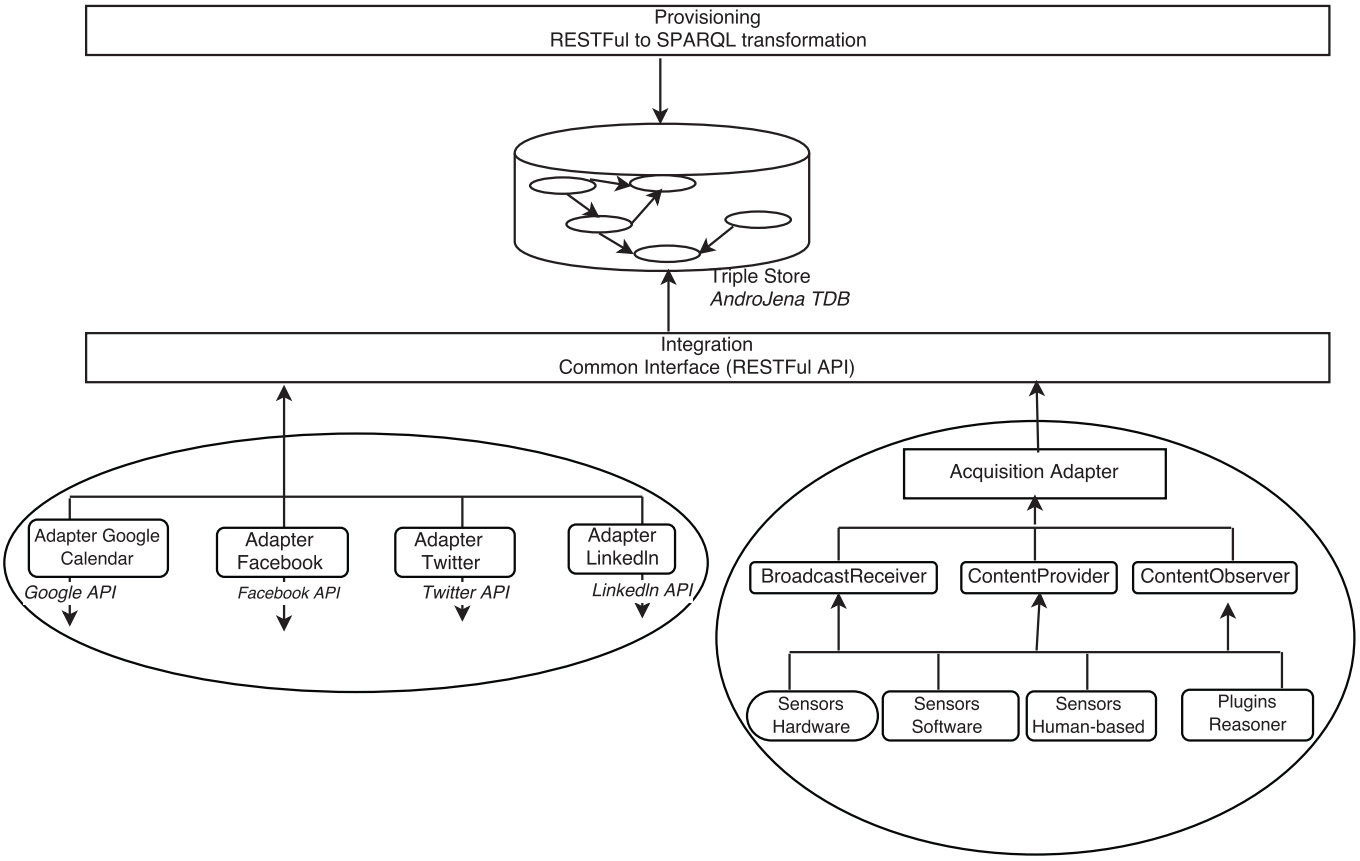

Figure 1. AndroCon System Architecture

of smart and wearable devices and the advancement of mobile technology, however, the implementation and realization of such systems is expected to encounter some specific challenges, which are addressed in this paper.

1. The acquisition and integration of heterogeneous context data from variety of sources into either centralized or distributed repository.
2. Developing a flexible approach of reusing mobile application context data in other platforms to provide support to web services and applications.
3. The provisioning of context data locally and remotely to applications running on different platforms, such as mobile devices, PCs and servers.

3. AndroCon Context-Aware Middleware Framework

This section presents the overview and fundamentals of the AndroCon framework. We introduce the aggregate components that enable Android mobile devices function as Web service provider, and elaborate on context integration components of our framework. We also present the procedures of managing both social and physical context data with semantic web technology. Firstly, we present the abstract architecture of AndroCon System as a layered architecture shown in Figure 1 below and we provide a general overview of our approach afterwards.

AndroCon system comprises of four main components as shown in figure 1: the context acquisition as the bottom layer, which incorporates series of adapters from SCIMS acquisition component and AWARE client [5,6]. The upper layer is the context integration layer. The integration layer passes data to the context storage layer, which is responsible for storing and retrieving context data. And above the storage layer is the context provision layer, which provides context data to applications through RESTful API.

3.1. Context Acquisition

AndroCon framework acquires both physical and social context data. While it leverages the AWARE framework to collect sensor data, its exploits SCIMS acquisition component for social context data acquisition. In order to provide accurate and robust knowledge base for external applications, AndroCon seamlessly integrates context data from these two sources.

Physical Context Acquisition . The AWARE client is a mobile application that runs on Android mobile device and collect raw data from sensors. It supports a long list of mobile sensors such as barometer, gyro, temperature, accelerometer, proximity, compass, gesture, heart rate etc. In order to allow other Android applications to have access to context data retrieved by AWARE plugins and sensors, it stores the data in a local relational database and encapsulates it with Android's *ContentProviderAPI*. For distributed computing purpose, it upload data to remote server but does not support local data integration. AWARE also supports *ContentObserver* approach, which allows messages to be sent only when context changes, to save mobile device power. Hence, AndroCon adopts *ContentObserver* approach to ensure that only modified context will be acquired and stored. Applications can also obtain data through passive ways, by simply subscribing to system broadcast by implementing a *BroadcastReceiver* object and listening to the broadcast. The functionality of the AWARE's core component is sensor data acquisition. At present, AndroCon framework does not include context interpretation, instead, it leverages the plugins currently included in AWARE. However, interpreters will be implemented by creating AWARE plugins in subsequent version of AndroCon solution.

Social Context Acquisition . SCIMS acquisition component retrieves raw social context via Online Social Network (OSN) APIs and has a preferred context interpreter. It runs on PCs and servers. Sources of social context include Facebook, Twitter, LinkedIn, etc. Therefore, it can be implemented to retrieve high-level social context data. However, SCIMS is a server-based framework only, which implies that social context data can only be collected via cross-platform approaches, such as RESTful API. This indicates that more network operations need to be invoked. In current version of AndroCon solution, an adapter is implemented to call SCIMS's RESTful API in order to acquire context data.

3.2. Context Integration

In order to provide a robust and rich context data source for context-aware processes, AndroCon framework integrates both physical and social context data acquire via AWARE and SCIMS frameworks respectively. Therefore, series of adapters are implemented in AndroCon System to obtain these context data. In this context, the data sources are classified into local and remote sources. For local sources, the adapters retrieve data through Android *ContentProvider* or *BroadcastReceiver*; for remote sources, the adapters will call their respective APIs, for example, Google Calender API is used to track and retrieve events information, which may include date, name of event, name of event location etc. as context data. SCIMS currently functions as remote context data source, and it supports retrieving data through RESTful API. Therefore, the adapter for SCIMS needs to implement the functionality of consuming SCIMS's service. On the other hand, AWARE client functions as a local resource, the adapters are expected to call Android APIs to retrieve data. The following subsection explained the semantic approach used to represent both type of context data. We used ontological approach to integrate and manage both data sources.

3.3. AndroCon OntologyDevelopment

Most context-aware scenarios require utilizing both physical and social context to generate more meaningful knowledge. The integration of these forms of data requires a more comprehensive and flexible data model to represent the context information and provide support for more complex knowledge interpretation. While there is no existing model for such data composition, AndroCon adopts ontology based approaches to represent context data. AndroCon's Ontology development follows the guidance of the *NeOn* methodology [10]. AndroCon adopts one of the authorsâĂŹ scenarios of reusing, merging and re-engineering ontological resources, to merge the acquired social and physical context data together. We reuse mIO! Ontology network as part of AndroCon's physical ontology. The mIO! ontology network, which consist of user's

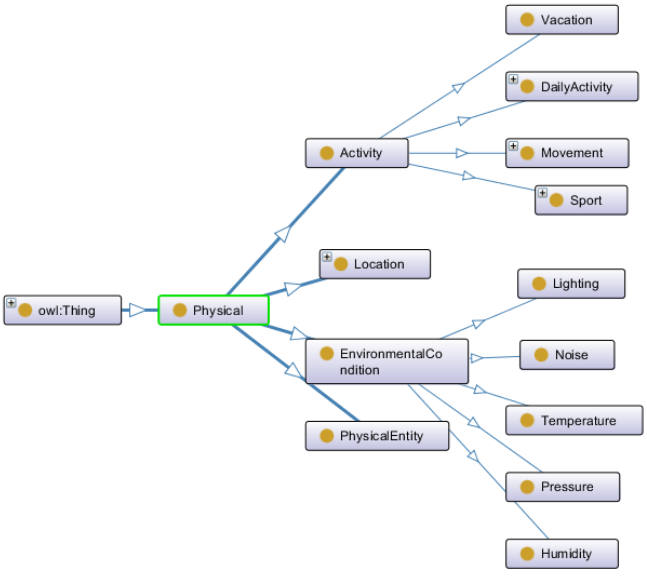

Figure 2. Physical Context Ontology

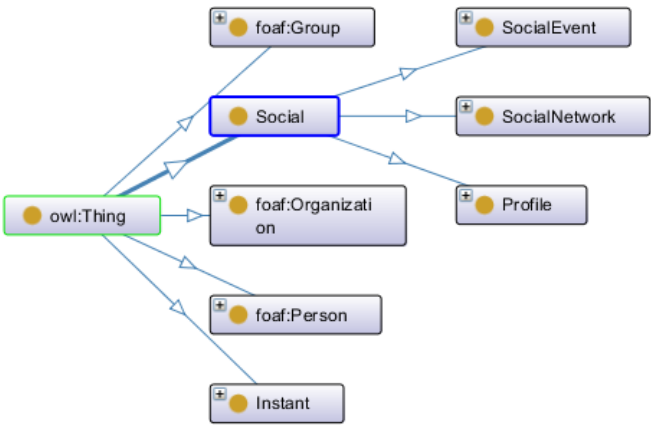

Figure 3. Social Context Ontology

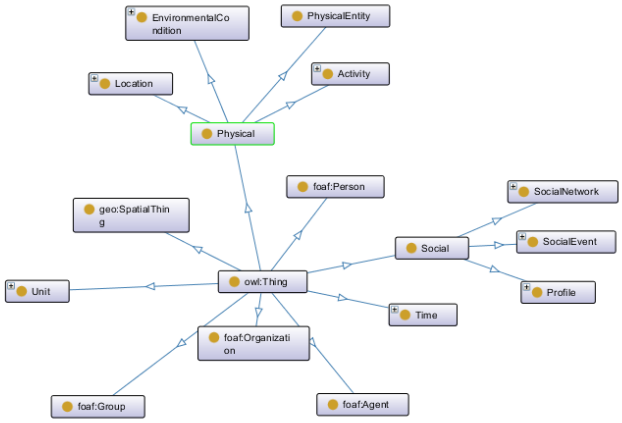

Figure 4. AndroCon Integrated Context Ontology

context information and ontologies that describe different sub-domains required for modelling the context information such as location information (position or date),Environmental information (temperature,humidity or luminosity) and some social information. The sub-domains included in mIO! network are Location, Services, Devices, Power, Role, Providers, Users, and Time [7]. The structure of the ontology follows the two-layer rule – a core layer, which represents the upper level knowledge and the domain-specific layer, which contains domain-related instances.

Figure 2 shows the restructured AndroCon's physical context ontology. As shown in the figure, the ontology is divided into two distinct layer of physical things. The upper level captures the basic concepts of physical abstracted from the real-world use case scenarios. This includes classes such as *Activity*, *Location*, *EnvironmentalCondition* and *Physicalentities*. The second layer comprises of the subclasses, which are related to the classes in upper layer. This hierarchical definition gives room for flexibility, as each sub-classes can be further extended or related to another physical entity. For example, Sport as a domain and subclass of activity can be further extended to form a type tie with sub-entities such as volleyball, Boxing and other sports.

The mIO! Ontology is re-engineered by removing some redundant classes and properties originally included, while merging SCIMS's social ontology to the network. On the other hand, since AndroCon's social context ontology design is compatible with the structure of SCIMS social ontology, the ontology is merged with the AndroCon's general ontology structure without being revised. The SCIMS social ontology is described in Figure 3. The entire network is reviewed and restructured over again after merging to ensure total elimination of redundancy. For example, both mIO! and SCIMS included FOAF and time ontology to their networks, such repetition has to be removed to minimize processing load and complicating reasoner's processing. The SCIMS ontology defines five upper-class entities, which comprises of three FOAF ontology used to define Person, Organization and group. The other components are Social class and Instant entities.

The resulting integrated ontology shown in Figure 4 is defined and stored as an .owl file, which is used as the schema of the database. We use the super class concept of thing to tie other class to a common entity and discard redundant classes or entities that are common to the two ontologies. Unlike other frameworks that uses relational databases to store context data, AndroCon adopts a RDF triple store to save context data.

3.4. Context Provisioning

One of the major distinguishing features of AndroCon context-aware framework that separates it from other related frameworks is its capability that enables mobile devices function as web service provider. Although there is no development framework such as JAX-RS [10] or WCF [11] to facilitate rapid web service development in mobile devices, AndroCon adopts the simple and easily consumed RESTful protocol to replace the memory and computational intensive SOAP protocol, which improves the efficiency of data provisioning mechanism. This approach built on authors [12] research suggestions in a more comprehensive way. The following subsections discuss the architecture of AndroCon's Web Service provisioning.

Service Architecture. The architectural design of the AndroCon System Web Service consist of two layers: The web server layer and the database layer. In order to facilitate HTTP responses and requests to and from

the Client, a web server must be deployed in the mobile device. Whenever the HTTP request reaches the server from the client, the server initiates a new session and stores the parameters in a buffer. A dedicated method in the server class takes the responsibility of a business logic handler. The method analyses the parameters and invokes corresponding method to serve the request and generate appropriate response. The web server upon receiving the request will analyze the URL of the request and find a RESTful resource that the client intend to access. Once this is achieved, the server immediately maps the resource to one of the classes within the *androconontology* package. Figure 5 shows the functional layers of web service architecture. The business logic handler will execute the next action. It generates different types of triple store transactions, based on the type of HTTP method invoked. For example, if the request is a HTTP POST method, the handler will collect a JSON string from the request body and de-serialized it, then invoke the insert method of the *AndroConAddress* object. The handler then passes the de-serialized JSON objects to the method as parameter. The database layer is implemented as a RDF triple store, which consists of a SPARQL engine and an underlying TDB instance. The *AndroConAddress* class object query and insert methods will transfer RESTful requests to SPARQL statement and forward the statement to SPARQL engine. The engine then interacts with TDB and return the query results. When the *AndroConAddress* object receives the SPARQL query results, it serializes the results into a JSON string and then pass the resulting string to the web server. The web server prepares the HTTP response message and returns it to the client.

RESTful API Design . The design of AndroCon's API follows the four basic design principles of RESTful services. These APIs are expected to invoke the four *HTTP* methods, namely; *GET*, *POST*, *PUT* and *DELETE* explicitly and in a consistent approach. In the AndroCon framework, *POST* is invoked to create resources, *GET* to retrieve a resource, *PUT* to update a resources, and *DELETE* to remove a resource. In AndroCon, a resource can be a list of context, an instance of context classes or a context triple. AndroCon's API requests are stateless. All the requests will contain all of the parameters, context, and data needed by the server to generate a response within the HTTP headers and body. Therefore, no request is dependent on other requests and server does not need to maintain the state session. The stateless design helps to simplify the business logic handler and improve performance of the entire service system. Furthermore, AndroCon uses static URIs, which follows hierarchical structure to represent resources. This reflects the relationships between resources. The client can request

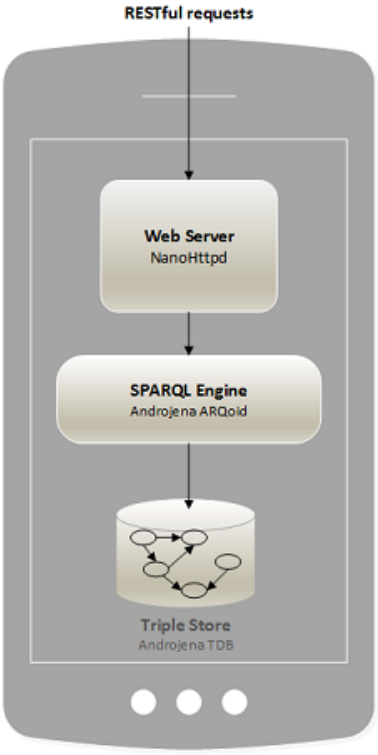

Figure 5. AndroCon Web Service Architecture

for a specific piece of data by inserting parameter values in the URI. For example, the */haslocation* resource returns list of all the locations a user has visited. If the client requires the locations user has been on 25*thDecember*2016, it can use parameters such as $https://localhost:8767/haslocation?date=20161225$. In addition, AndroCon exploits JSON friendliness to user environment to transfer data context data. Data presented in JSON format are readily accessible without overhead [13].

Mapping RESTful Requests to SPARQL Queries . As shown in Figure 1, the AndroCon's acquisition adapters and the web service interfaces interact with the triple store using SPARQL language. A mapping component on the server side is provided to support the client applications that call RESTful requests to retrieve data. This converts RESTful request to their respective SPARQL commands.

Table 1 shows the four types of RESTful methods supported with their respective SPARQL commands. Although the concept of mapping used in AndroCon is similar to the one stated in [14], AndroCon does not use script to map its ontology classes with their respective properties. Instead, AndroCon uses Java classes to represent all elements of the ontology, including classes. Properties and individual attributes. Each Java class contains methods to conduct the four types of operation stated in Table 1 above. By that means, the RESTful

Table 1. Mapping RESTful methods to SPARQL commands

RESTful Methods	SPARQL Commands
GET	*SELECT*
POST	*INSERT*
PUT	*UPDATE*
DELETE	*DELETE*

resources can be mapped into Java classes that are responsible for generating a fully functional SPARQL statement. Table 2 shows the mapping of the RESTful resources with their associated AndroCon's Ontology classes.

Table 2. Mapping RESTful Resources to Ontology Classes

RESTful Resources	Ontology Classes
location	*AndroConAddress*
locationincontext	*AndroConLocationContext*
activity	*AndroConTemporalEntity*
activitycontext	*AndroConTemporalContetxt*

The next section discusses the detail implementation of AndroCon functional components with specific use case.

4. Implementation

AndroCon middleware framework is built to enable mobile devices provision web service and to allow integration of multiple context data sources. In this section, we discuss the implementation of AndroCon's acquisition adapters, the context ontology OWL files, the context triple store, and the web service. The implemented functionalities will be described and each of the component will be verified. An experimental case study that show case how the solution is verified and the result is also presented.

4.1. Application Scenario

In order to verify and demonstrate how AndroCon tracks mobile user's location and activities by generating new context data and incorporating the data into other applications through RESTful API, we give in this section a location-based application scenario for University building premises. For example, if a mobile user $user_1$ is located in the University building tagged $AUT-WT$ at $3pm$ on $October11th2016$. AndroCon will detect the user's location and assign related attributes, if there is an existing instance $AUT-WT$ building in the triple store. If otherwise, AndroCon will generate an instance of $AUT-WT$ with five datatype properties: $addressCountry$, $adderessCity$, $addressStreet$, $addressNumber$, and $addressPostalCode$. On the instance $AUT-WT$, AndroCon will insert two triples to the triple store to represent the location knowledge - the first is $androcon:user1$ $androcon:hasLocation$ $androcon:AUT-WT$, the other is $androcon:AUT-WT$ $androcon:timestamp$, and. A sample of SPARQL query statement for obtaining a track of the location $user_1$ has been is displayed below:

PREFIX androcon: http://localhost/toplevel/androcon.owl#>
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: http://www.w3.org/2002/07/owl#>
SELECT ? location ? time
WHERE
{ androcon: User1 androcon: hasLocation? location androcon: ?location androcon: timestamp? time }

If $User_1$ has been in $AUT-WT$, $AUT-WG$ and another University building tagged $UoA-301-Fo53$ in the past three days, the expected output is as follows:

---

androcon: AUT-WT androcon: 20161009-9:33am
androcon: AUT-WT androcon: 20161010-10:21am
androcon: AUT-WT androcon: 20161011-9:05 am
androcon: AUT-WG androcon: 20161009-3:14 am
androcon: UoA-301-Go53 androcon: 20161011-9:05am

If $User_1$ wants to track the location context with a browser, the user will have access to the server via the resource address such as $https://localhost:8756/locationcontext$. $AndroConHttpd$ will parse this request to the SPARQL query statements described above and to the $AndroConDB$. $AndroConDB$ will query the triple store and return the result to $AndroConHttpd$, which then parse the results to JSON string and return to the browser.

4.2. Context Data Processing and RESTful Service

AndroCon adopts a location-based use case during experimentation, where physical context data were collected, as input to the system. The experimentation was done with AdroCon client application running on user's mobile device. The application tracks mobile user's locations and physical activities by generating new context data as their events or location changes. For example, when the AdronCon user visited a location with an existing instance in the triple store, it automatically detect the user location and relate it to other attributes. If the location is not an existing instance, AndroCon will generate an instance of the location with related properties.

The following resources were used to examine and verify the functionalities of AndroCon relative to the scenario discussed in section A.

1. **Datasets Used:** Dataset for verifying AdroCon system comes the existing Android application-AWARE client. AWARE provides a plugin reasoner for generating activity context data. AndroCon implemented other location context reasoner by calling Android API. AndroCon ontology Schema consists of five owl files, which includes *time.owl*, *androcon – environment.owl*, *space.owl*, *buildings.owl*, and *AndroCon.owl*. *AndroCon.owl* represents the top-level ontology of the schema. The schema is design by restructuring the mIO! ontology network and SCIMS social context ontology. Each of the owl files is pre-loaded into TDB during the triple store initialization phase.

2. **Modules Used:** AndroCon leverages four existing modules to implement context acquisition, context storage and web server. The AWARE client module provides about 21 Android sensor drivers and one context reasoner. This was used as the generator of physical context data. The SCIMS acquisition module, which has five adapters, was used as the generator of social context data. *Androjena* [15] is an Android porting version of Apache Jena project, was deployed as a supporting library for triple store. It consists of common *rdf* query libraries, a SPARQL engine (ARQ), and a graph store for triples (such as TDB). AndroCon implemented TDB in the testing of devices. *NanoHttpd* [16] is common light-weighted web server frequently used by Android developers as an HTTP server was also deployed in AndroCon system.

3. **Tools Used:** Different tools were used during the implementation of AndroCon. One of the key tools is *Protege* [17]. *Protege* is a prevalent ontology design and development tool among academia. It was used to create, modify, combine and restructure AndroCons context ontologies. RAML [19] was used for RESTful API design and modelling. AndroCon application was developed using Java and its verifiable on multiple devices.

Figure 6. Shows the three major functional classes implemented in AndroCon. Business processes relating to RESTful service were implemented within the web server. The class *AndroConDB* shown in figure 6 maintains the TDB triple store and supports CRUD operations through eight methods. The business process method in web server parses RESTful request from the clients to SPARQL queries. *AndroConDB* receives the queries and return results to the web server. In case where multiple adapters are needed to insert data into *AndroConDB* and single client querying *AndroConDB* with moderate workload, *AndroConDB* could efficiently contain and handle the request. Moreover, the *AndroConHttpd* class extends *NanoHttpd* and is responsible for request parsing, response generating, and service invocation. It is also in charge of business process. We envisaged that as more plugins are added, a dedicated class would be included for business process in subsequent versions. Location and activity context are currently supported by *AndroConHttpd*, which corresponds to *locationContext* and *activitycontext* in RESTful resources respectively. For verification purpose, HTTP forms were used as the client to initiate RESTful POST request and the context data were posted in JSON format [30]. Whenever the server receives the JSON string, it will parse it into the SPARQL INSERT statements and then pass the statements to *AndroConDB*. This database then queries the triple store and return the results to *AndroConHttpd*. *AndroConHttpd* will then parse the results to JSON string and return to the browser.

5. Evaluation

In this section, we describe our experimental set up, metrics for evaluation and the performance of AndroCon based on impact on mobile device battery life and time taken to execute specific context-aware tasks. We also report the experimental results.

5.1. AndroCon's Impacton Mobile device battery life

We evaluated AndroCon's impact on mobile device battery life to validate user experience and the performance in terms of power consumption. In order to study AndroCon's power consumption in real-time, we employed a diagnostic tool called Qualcomm's Trepn Profiler (*QTP*) described by authors [5] as an efficient tool to evaluate Android mobile applications power consumption and measure device battery life, processor load of each high performance sensor. *QTP* was used to measure power consumption of mobile

Table 3. AndroCon testing Results

Time(mins)	Battery Consumption %	Battery Power(mWh)	CPU Load %
5	4	305.67	4.04
10	7	365.33	4.10
15	9	335	4.12
20	12	579.56	5.17
30	13	545.26	6.27
60	19	737.56	5.85

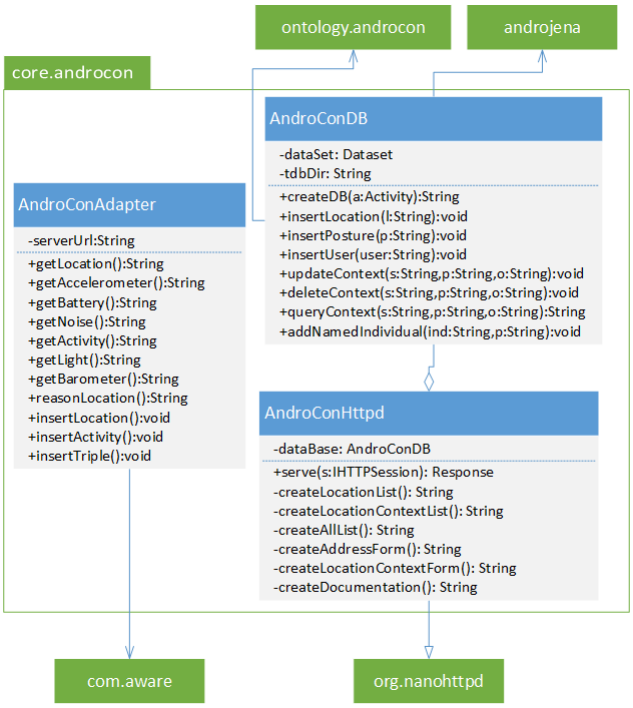

Figure 6. AndroCon Class Diagram

device that has AndroCon installed on it. We measured the power consumption for both idle and running period of the application and compared the results with exiting context-aware applications power consumption, specifically AWARE application. After several iterations with ten different sample readings taken at CPU frequencies 5Hz and 10Hz respectively using Nexus 6.0 Android phone, the results are shown in Table 3, which contains records for time taken to execute the entire AndroCon processes, Battery Consumption in percentage% ,Battery power in milliWatt per hour, and CPU load in percentage%.

5.2. Results and Analysis

The results derived from the experimental phase of our research were based on battery performance and CPU utilization of Android device running AndroCon application. As stated in the above subsection, we monitored and recorded the device battery usage, CPU loads and average power consumption when AndroCon is active and fully running. Table III and figure 7 shows the readings taken in real time and a graphical representation of the key resources involved in AndroCon context-aware processes execution. As expected, the power consumption of AndroCon increases from the acquisition stage of the context-aware processes to the provisioning. While some the stages consume a moderate amount of power, there is a huge surge in power consumption during acquisition stage, which precedes the start up stage. However, the ability of mobile device to withstand several iterations without losing up to 10% of its battery life justifies the sustainability of deploying mobile devices as Web services providers. In order to validate the ability of AndroCon to maintain certain level of power sustainability when running on mobile device, we compared our results with the AWARE performance results. Figure 8 shows the comparison between AndroCon's energy consumption and AWARE framework. Although AndroCon appears to consume more power compare to AWARE application, the increase is sustainable and justifiable with a worthwhile mobile-based Web service provisioning. Figure 9 shows AndroCon's CPU loads over 180minutes. The Normalized CPU load reports the usage of the application with respect to the

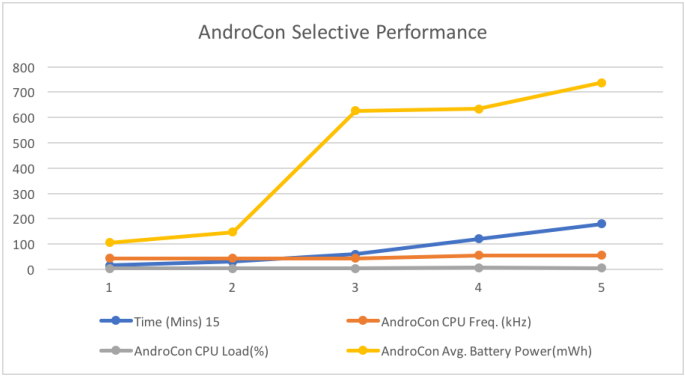

Figure 7. AndroCon CPU Utilizationand battery performance

Table 4. : Comparison of Five Context-Aware Middleware Framework

Framework	SCIMS	Hydrogen	SOCAM	AWARE	AndroCon
Platform	*Server*	*Mobile*	*Server*	*Hybrid*	*Mobile*
Type	*Social*	*Physical*	*Activity*	*Physical*	*Both*
Sources	*OSN*	*Sensors*	*Sensors, apps*	*Sensors, apps*	*Sensors, apps*
Modelling	*Ontology*	*Objects*	*Ontology*	*RDBMS*	*Ontology*
Reasoning	*5Reasoners*	*N/A*	*Interpreters*	*N/A*	*N/A*
Storage	*RDF files*	*Objects*	*KnowledgeBase*	*SQLites, MySQL*	*TDB*
Query	*SPARQL – DL*	*N/A*	*SPARQL – DL*	*SQL*	*SPARQL*
Provisioning	*RESTful*	*P2P*	*RESTful/SOAP*	*Upload/Download*	*RESTful*

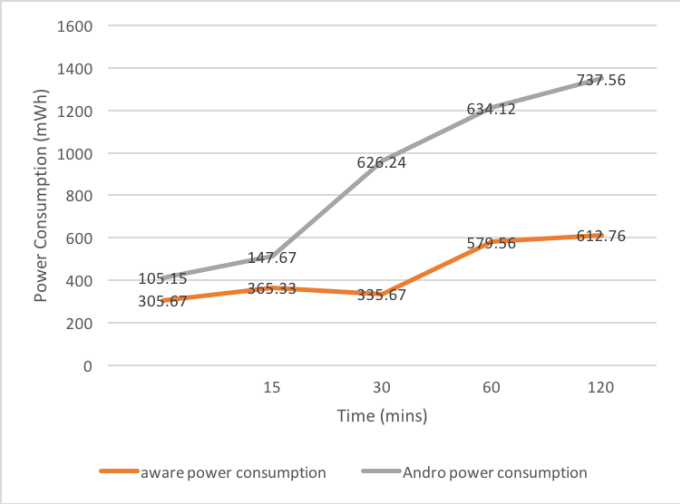

Figure 8. AndroCon Vs AWARE power Consumption

maximum potential of the CPU. As shown in the figure, the application started to use about 4.5% of CPU loads during the testing period of the first 15 minutes and later decreases as the time increases up to 60 mintues, and then there is an increase from 60 to 180 minutes.

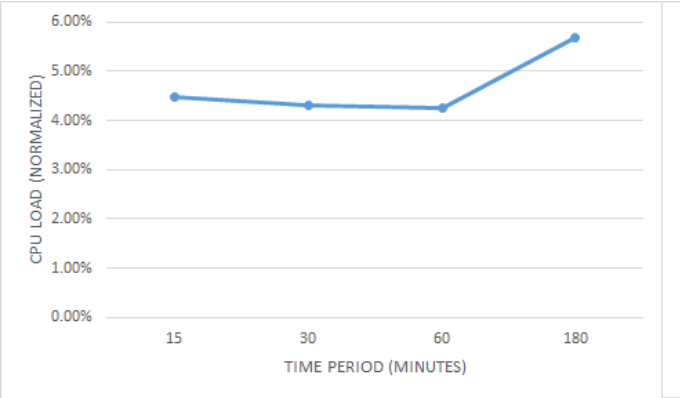

Figure 9. AndroCon CPU Load (Normalized)

But during the whole testing period of 180 minutes, the CPU load is always below 6%.

6. Related Work and Discussion

In this section, we first discuss the related context-aware middleware frameworks in general, and then we analyze and compare four related and most comprehensive context-aware middleware frameworks with AndroCon, including the ones leveraged in the framework.

A number of context-aware middleware solutions have been proposed to address various challenges facing context-aware and pervasive computing. Since existing solutions have varied features, they only contributed partially, for context, data, or mobile service management related application development [25]. There seems to be no single middleware solution that can address in entirety the challenges of context-aware computing due to diverse underlying complexities. An early approach is the Context Toolkit proposed by [26]. Context toolkit is a context-aware middleware framework, with the definitions of key functionalities of context acquisition, context reasoning, context integration and context provisioning. It is a conceptual framework, which consist of three abstractions-widgets, aggregators, and interpreters. Widget collects context data and generates higher level of knowledge such as location and activity. The widget sends context data to aggregator through uniform interface. The aggregator integrates the data on distributed platform, then provision context data to applications. The interpreter turns low-level context information such as raw sensor data into higher-level knowledge through rules and algorithms. CORTEX [27] is a framework designed for mobile devices. CORTEX framework proposes a context-aware middleware for mobile applications, leveraging the Sentient Object Model. CORTEX provides an abstractions layer for retrieving sensor data so that the applications do not need to directly interact with low-level sensor drivers. It provides a context reasoning mechanism based on a hierarchy of contexts.

In order to make CORTEX more suitable for mobile environment, context acquisition module was event-based. This implies that only changes of the context-data will be retrieved. CoCaMAAL [28], is another context-aware middleware for ambient assisted living. It is designed as a cloud-based model to address issues related to management of sensor data and derivation of contextual information as well as monitoring user's activities and service discovery. CoCaMAAL implemented a service-oriented architecture for unified context generation, which is achieved by aggregating raw sensor data and timely selection of appropriate service using context management system. Moreover, a larger number of context-aware middleware have their management systems running on PCs and servers including frameworks such as SOCAM [20], SCIMS, Aura [29] and Context Toolkit. SOCAM is a service oriented middleware project for building rapid prototypes of context-aware mobile services. Its core component, context interpreter acquire context data from distributed context providers and interprets the data into higher-level context information, and provision for data to the applications. SOCAM uses ontology-based model and focuses on context reasoning. SCIMS is a social context information management system, which specializes in acquiring raw social data from multiple sources. SCIMS includes other functionalities such as an ontology-based model for classifying, inferring and storing social context information. SCIMS also includes a query interface for accessing and utilizing social context information.

In mobile computing, factors such as scalability support for distribution, interoperability, self-adaptivity, support for mobility, modularity, plug-ability, etc. are of particular interest [26]. Next we compare SCIMS, Hydrogen, SOCAM, and the AWARE framework with AndroCon based on some of these criteria. This is done mainly to position AndroCon in the big picture of the context middleware frameworks, thus emphasis on the idea of AndroCon's contribution.

SCIMS is a context management system, which specializes in social context. It incorporate various functionalities such as the ability to acquire raw social data from multiple sources , an ontology based classification capability, inferring and storing social context information, especially social relationship and status. SCIMS also uses an ontology based policy model for authentication and authorization, in order to control access to the context data, it implements a query interface for accessing and utilizing social context information. SCIMS stores context data in RDF file and adopts derive SPARQL-DL query engine with description logic reasoners. It combines five different DL reasoners (Pellet, HermiT, TrOWL, Fact++ and RacePro) with the query engine. Unlike AndroCon, SCIMS is not mobile based and only runs on PCs and servers.

The Hydrogen context-framework [27] is implemented to run on mobile devices. Rather than designing a centralized context repository, it proposes a peer-to-peer context sharing. It adopts a three-layer architecture, which are all located on the same mobile device. Hydrogen presents a context framework, which comprises of five types of context (time, location, device, user, and network), and it is designed to be extensible. It is implemented using the PersonalJava virtual machines Jeode and the J2ME J9 on iPAQs (3660 and 3870) with the PocketPC 2002 operating system. Unlike AndroCon, it provisions context data via context-sharing technology and not through web services. This technology requires all the parties involved in context sharing to have a context server component. According to [27], context data is stored in a Java executable object tagged ContextServer. This indicates that there is no specific database used for storing context data.

SOCAM [12] is a service oriented context-aware middleware framework, which focuses on context reasoning and context modelling using ontological approach. SOCAM is an architecture for the rapid building and prototyping of context-aware mobile services. It uses a central server called context interpreter, which acquire context data through distributed context provider and provides logic reasoning services used to process context information. SOCAM has an Intel Celeron 600M CPU gateway, which runs on Linux 2.4.17 kernel. The context interpreter initiates this gateway to attached context data sources to the gateway. Context data sources such as vehicle sensors, computers and mobile devices are attached to this gateway via the Ethernet, WLAN, and Bluetooth and so on. The context aware mobile services are located on top of the architecture, thus, they make use of the different levels of context and adapt their behaviours according to the current context.

AWARE is one of the frameworks exploited in AndroCon framework. It focuses on context acquisition, particularly through mobile sensors. Although, AWARE has is limitations, it takes mobile power consumption as an important evaluation metric. It only support context storage and provisioning through remote servers. Unlike Hydrogen and SOCAM, AWARE does not use ontology-based context modeling and only provides limited reasoning functionalities.

AndroCon exploits AWARE and SCIMS framework to improve on the limitations of these context aware systems. All its architecture layers (context acquisition, context integration, context storage, context provisioning) resides on the same mobile device. It acquires both social and physical context data, and supports ontology-based context modeling both physical and social context data. AndroCon uses RDF triple store to manage context CRUD operations and uses SPARQL as the query language. A RESTful service, which provision context data to upper application is implemented

to support distributed context-ware systems. Table IV. compares AndroCon with other context-aware middleware.

7. Conclusion and Future Works

In this paper, we have presented AndroCon, a mobile-based context-aware middleware framework for acquiring, integrating, managing and provisioning both physical and social context. AndroCon differs from the previously noted approaches in that it is an android-based framework and leverages two key frameworks to acquire a more robust context data. It does not only uses sensor data but also reuses existing ontological networks and social context data. It also show case in its implementation how to provision data locally and remotely to applications running on different platforms, such as mobile devices, server or PCs. . The framework extended the AWARE framework by adding more efficient context acquisition module into it. AndroCon provides several advantages such as reusability, extensible, interoperability and standardization. It reuses the mIO! Context ontology network and incorporated social context ontology into it. AndroCon provides a richer context database by integrating physical context with a set of restructured social network ontology. In the implementation of AndroCon, we proved and demonstrated the concept of deploying RDF triple store in mobile devices and making mobile devices web service provider. Our next step work is to carry out case study and assess AndroCon's effectiveness in supporting the development and operation of context-aware mobile applications in the promising Internet of Things area.

References

[1] H. Vahdat-Nejad, K. Zamanifar, and N. Nematbakhsh, Context-aware middleware architecture for smart home environment. International jour-nal of smart home, 7(1), 77-86. 2013.

[2] E. Cuervo, A. Balasubramanian, D. K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: making smartphones last longer with code offload. In Proceedings of the 8th international conference on Mobile systems, applications, and services (pp. 49-62, 2010.)

[3] M. J. OSullivan, and D. Grigoras. Context aware mobile cloud services: A user experience oriented middleware for mobile cloud computing. In Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2016 4th IEEE International Conference on (pp. 67-72). 2016 IEEE.

[4] V. Raychoudhury, J. Cao, M. Kumar, and D. Zhang. Middleware for pervasive computing: A survey. Pervasive and Mobile Computing, 9(2), 177-200. 2013.

[5] D. Ferreira, V. Kostakos, and A. K. Dey. AWARE: mobile context instrumentation framework. Frontiers in ICT, 2, 6. 2015.

[6] M. A. Kabir, J. Han, J. Yu, and A. Colman SCIMS: a social context information management system for socially aware applications. In International Conference on Advanced Information Systems Engineering (pp. 301-317). Springer Berlin Heidelberg, 2012.

[7] Poveda Villalon, M., SuÃąrez-Figueroa, M. C., GarcÃŋa-Castro, R., GÃşmez-PÃĺrez, A. (2010). A context ontology for mobile environments.

[8] D. Bottazzi, A. Corradi, and R. Montanari. Context-aware middleware solutions for anytime and anywhere emergency assistance to elderly people. IEEE Communications Magazine, 44(4), 82-90. 2006.

[9] P. Bratskas, N. Paspallis, and G. A. Papadopoulos. An Evaluation of the State of the Art in Context-Aware Architectures. Information Systems Development, 1117-1128, Springer. 2008.

[10] T. Gu, H. K. Pung and D. Q. Zhang. A middleware for building context-aware mobile services . In Vehicular Technology Conference. VTC Spring. IEEE 59th (Vol. 5, pp. 2656-2660). IEEE. 2004.

[11] K. Peffers, T. Tuunanen, M. A. Rothenberger and S. Chatterjee, A design science research methodology for information systems research. Journal of management information systems„ 24(3), 4577. 2007.

[12] A. Cadenas, C. Ruiz, I. Larizgoitia, R. Garca-Castro, C. Lamsfus, I. Vzquez,... and M. Poveda,Context management in mobile environ-ments: a semantic approach. In Proceedings of the 1st Workshop on Context, Information and Ontologies (p. 2). ACM 2009.

[13] M. Hadley and P. Sandoz. JAX-RS: Java API for RESTful Web Services. Java Specification Request (JSR), 311. , 2009.

[14] A. E. Look. Introducing Windows Communication Foundation. ‚.âĂİ David Chappell, Chappell and Associates, September (2005).

[15] S. N. Srirama,M. Jarke and W. Prinz). Mobile web service provi-sioning. International Conference on Internet and Web Applications and Services/Advanced International Conference on (pp. 120-120). IEEE. 2006

[16] K. Alexander RDF in JSON: a specification for serialising RDF in JSON. SFSW. , SFSW. 2008. A. Jena. A free and open source Java framework for building Semantic Web and Linked Data applications. Available online:jena.apache.org/ (accessed on 27 march 2017. SuÃąrez-Figueroa,

[17] M. C., GÃşmez-PÃĺrez, A., FernÃąndez-LÃşpez, M. (2012). The NeOn methodology for ontology engineering. In Ontology engineering in a networked world (pp. 9-34). Springer Berlin Heidelberg.

[18] V. Raychoudhury, J.Cao, M. Kumar and D. Zhang. Middleware for pervasive computing: A survey Pervasive and Mobile Computing, 9(2), 177-200. 2013.

[19] A. K. Dey,and G. D. Abowd. The context toolkit: Aiding the development of context-aware applications. In Workshop on Software Engineering for wearable and pervasive computing (pp. 431-441). 2000.

[20] G. Biegel and V. Cahill. A framework for developing mobile, context-aware applications. In Pervasive Computing and Communications, PerCom 2004. Proceedings of the Second IEEE Annual Conference on (pp. 361-365). IEEE. 2004.

[21] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann and W. Retschitzegger. Context-awareness on mobile devices-the hydrogen approach. . In System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on (pp. 10-pp). IEEE. 2003.
[22] A. Forkan, I. Khalil and Z. Tari. âĂİCoCaMAAL: A cloud-oriented context-aware middleware in ambient assisted livingâĂİ. , Future Generation Computer Systems 35 (2014): 114-127. 2014.
[23] D. Garlan, D. P. Siewiorek, A. Smailagic,and P. Steenkiste. Project aura: Toward distraction-free pervasive computing. , 1(2), 22-31. 2002.
[24] H. Mller, L. Cabral, A. Morshed, and Y. ShuFrom RESTful to SPARQL: a case study on generating semantic sensor data In Proceedings of the 6th International Conference on Semantic Sensor Networks-Volume 1063 (pp. 51-66). CEUR-WS. org. 2013.
[25] R. Yusr, C. Paspallis, G. Esteban,F. Bobillo, and E. Mena Android goes Semantic: DL Reasoners on Smartphones In Ore (pp. 46-52). 2013.
[26] J. Elonen and K. TogiasNanoHTTPD. 2007.
[27] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe.). A Practical Guide to Building OWL Ontologies Using the ProtÃŕgÃŕ-OWL Plugin and CO-ODE Tools Edition 1.0. University of Manchester. 2004.
[28] Workgroup. R. A. M. L. âĂİRAML-RESTful API Modeling Language. 2015.
[29] D. Crockford. The application/json media type for javascript object notation (json). 2006.
[30] H. Hamad, M. Saad, and R. Abed.Performance Evaluation of RESTful Web Services for Mobile Devices. Int. Arab J. e-Technol., 1(3), 72-78. 2010.
[31] P. Bratskas, N. PaspallisSFSW. and G. A. Papadopoulos. An Evaluation of the State of the Art in Context-a. In Information Systems Development (pp. 1117-1128). Springer US. 2009.