electronics

MDPI

Article

AgPi: Agents on Raspberry Pi

Tushar Semwal * and Shivashankar Bhaskaran Nair *

Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, 781039, India

* Correspondence: t.semwal@iitg.ernet.in (T.S.); sbnair@iitg.ernet.in (S.B.N.); Tel.: +91-908-528-5069 (T.S.); +91-361-258-2356 (S.B.N.)

Academic Editor: Mostafa Bassiouni

Received: 4 June 2016; Accepted: 30 September 2016; Published: 19 October 2016

Abstract: The Raspberry Pi and its variants have brought with them an aura of change in the world of embedded systems. With their impressive computation and communication capabilities and low footprint, these devices have thrown open the possibility of realizing a network of things in a very cost-effective manner. While such networks offer good solutions to prominent issues, they are indeed a long way from being smart or intelligent. Most of the currently available implementations of such a network of devices involve a centralized cloud-based server that contributes to making the necessary intelligent decisions, leaving these devices fairly underutilized. Though this paradigm provides for an easy and rapid solution, they have limited scalability, are less robust and at times prove to be expensive. In this paper, we introduce the concept of *Agents on Raspberry Pi* (AgPi) as a *cyber* solution to enhance the smartness and flexibility of such embedded networks of *physical* devices in a decentralized manner. The use of a Multi-Agent System (MAS) running on Raspberry Pis aids agents, both static and mobile, to govern the various activities within the network. Agents can act autonomously or on behalf of a human user and can collaborate, learn, adapt and act, thus contributing to embedded intelligence. This paper describes how *Tartarus*, a multi-agent platform, embedded on Raspberry Pis that constitute a network, can bring the best out of the system. To reveal the versatility of the concept of AgPi, an application for a Location-Aware and Tracking Service (LATS) is presented. The results obtained from a comparison of data transfer cost between the conventional cloud-based approach with AgPi have also been included.

Keywords: Multi-Agent Systems; Cyber Physical Systems; Mobile Agents; Raspberry Pi; Internet of Things (IoT); BLE (Bluetooth Low Energy); Fog Computing

1. Introduction

The advent of the Internet of Things (IoT) [1] has facilitated devices to be connected with ease and enhanced to communicate and share data. Gartner Inc. (Stamford, CT, USA) [2] has predicted that by 2020, the IoT will form the basis for most business processes and systems. It has also conjectured that, by this year, more than 6.4 billion such devices will become connected. This drastic increase in connected devices is bound to revolutionize and greatly enhance Information and Communication Technologies (ICT) [3]. The Internet serves as an easy, reliable and accessible means for communication but is not without issues. Two of the major issues that crop up in the implementation of a typical IoT are security and the cost incurred in cellular communication. For applications such as a cab enquiry and booking system, which involves devices spread across an enormous geographic area, the use of the Internet can be traded off with some aspects in security. This may not be true for critical areas such as in military applications, hospitals, industries, smart buildings, etc. where security could be the major concern. Current IoT architecture [4,5] makes use of cloud-based solutions for imparting services to the users. The integrity, safety and insecurity of data stored in a cloud, along with the associated services for sensitive domains like medical and

industrial ones, remain matters of concern. The other issue is that in the conventional cloud-based IoT architecture, a device communicates through a central server supporting the cloud platform. This increases the cellular communication costs. A set of devices within a networked infrastructure can communicate locally and also perform computations, thus preventing a very large number of interactions with the cloud [6]. For scenarios such as an IoT for military or health care application, an Intranet based solution could perform effectively. Issues like security and communication expenses in an Intranet can be greatly contained. Another important issue is data privacy which is crucial in the case of medical hospitals, government and also for a consumer. Leakage of personal information and data ownership are at risk in a cloud-based centralized architecture.

In cloud-based systems, most of the data and intelligence churning activities are performed by a server hosted elsewhere in a centralized manner. For an Intranet-based solution, a framework that can facilitate this in a decentralized manner needs to be evolved. The devices participating in such an *Intranet of Things*, could include a range of connected embedded devices with their associated interfaces that connect them to the real *physical* world through sensors and actuators. The word "things" in an IoT refers to passive devices which seldom inherit any form of smartness within them. This is due to the fact that it is the cloud which is responsible for the intelligence and not the actual device. What is thus required to make an intelligent Intranet of Things is a *cyber* counterpart that can induce and embed intelligence into these devices. Multi-Agent Systems (MAS) [7] can act and provide as a fitting solution for realizing embedded intelligence. If such agents are made to operate on top of each embedded device, they can make decisions autonomously at the lower levels, thus transforming a network of such devices into a smart Cyber-Physical System (CPS). Figure 1 depicts such a CPS wherein the core comprises the real *physical* world being sensed and controlled via the sensors and actuators. The actual decision making and intelligence churning process is carried out by the agents (static and mobile) within the *cyber* world. These agents are programs that run on the connected embedded devices.

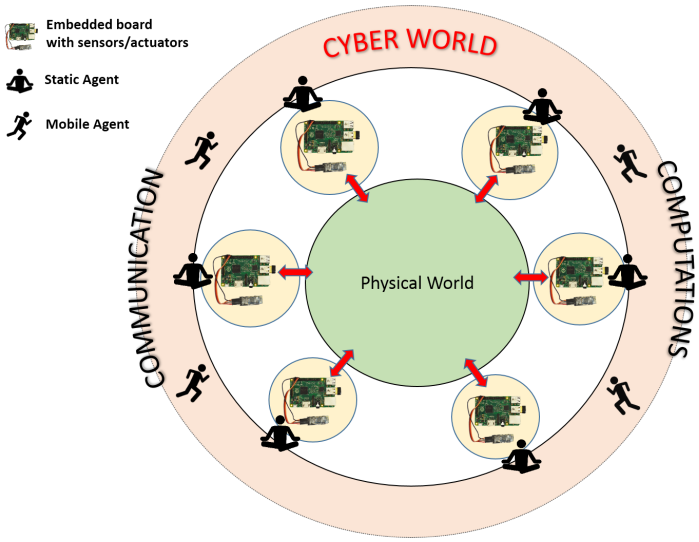

Figure 1. An agent-based Cyber-Physical System.

The concept of using agents in an Intranet of Things is very similar to an implementation of a *Fog Computing* environment [8]. The cloud is extended to the user side and constitutes a set of distributed and decentralized computing nodes which form the edge of the network. Such a concept has several advantages which include:

1. *Privacy*: Most of the cloud servers are owned by multinational corporations such as Amazon (Seattle, WA, USA), Google (Mountain View, CA, USA), Microsoft (Redmond, WA, USA), Cisco (San Jose, CA, USA), etc. which continuously receive data from the user side. Leakage of

personal information and data ownership becomes a critical issue when all of the user's data is collected for analytics purpose in the cloud [6]. A safer solution would be to have a local infrastructure on which the user has more control than the cloud server. This would allow local data filtering and computation before sending it over to the cloud. An agent-based system could be a better solution for ensuring privacy.

2. *Cost*: Cloud services follow a "Pay-as-you-go" model which adds to the cost as the storage and network communication increases [8]. In a local computational infrastructure model, these costs can be reduced if the data collected is filtered locally and only pertinent information is sent to the cloud.
3. *Network Latency*: A cloud has inherent latency issues and thus may not be a viable solution for applications such as live video streaming in connected vehicles, real-time data analytics in smart grids [8], etc., all of which require a rapid response. An Intranet of Things that uses agents, on the contrary, can provide fast local computations, thereby decreasing latency.
4. *Energy*: As already mentioned, agents in an Intranet of Things can filter the acquired data prior to sending it over to the cloud. Since this reduces communication overheads, it also reduces the energy consumed and consequently increases the battery life of the devices constituting the network [9].

In this paper, we emphasize the importance of agents (both static and mobile) and describe the use of *Tartarus* (Version 1.1, Robotics Lab., IIT Guwahati, India) [10], a Multi-Agent platform, on the Raspberry Pi (Raspberry Pi Foundation, London, UK). With a Location-Aware and Tracking Service (LATS) as a CPS application using *Tartarus* running on Raspberry Pi (henceforth, in this paper, *Pi* strictly refers to the Raspberry Pi) boards, we demonstrate the viability and versatility of the use of agents. The *Tartarus* agents are responsible for monitoring and tracking people within an indoor environment. Providing LATS is a challenging task in a dynamic environment [11]. Such scenarios call for queries that relate to *where* and *when* a person was or is in the area being monitored, *what* is the direction of the person's movement, etc. Firing queries to a database of related information stored centrally is fairly simple. However, if the person being tracked is in continuous motion, the database becomes dynamic in nature, which makes the task of querying, a complex one. This complexity further increases when the devices that track and store the data are numerous and have limited computational and storage resources. Data, in this case, is thus both dynamic and distributed across a network. Furthermore, new queries may also need to be fired at any point of time, which adds to the complexity of the system. This agent-based LATS portrays how agents, both static and mobile, can aid in satisfying such queries.

The rest of the paper is organized as follows. Section 2 provides a brief overview of Multi-Agent Systems (MAS) and the related platforms, while Section 3 gives the background on earlier realized LATS applications. Section 4 describes the architecture of AgPi and is followed by the LATS application in Section 5. The paper culminates with the results obtained and conclusions reached.

2. Multi-Agent Systems (MAS)

Agents are software entities that are capable of performing task(s) on behalf of a user [12]. They are autonomous and possess the ability to make their own decisions and drive themselves towards a goal. Maes et al. [13] refer to agents as computational systems that can sense and act autonomously in an environment in order to realize a set of goals. Just as human beings and robots form entities in the *Physical* world, these agents can be considered to be their counterparts in the *Cyber* world.

Multi-Agent Systems (MAS) can be defined as a compendium of different agents with their own problem solving capabilities and goals [14]. An MAS aids in abstracting a complex system into subsystems, each of which is represented by an agent. It is not just a collection of agents, but a system where agents coordinate to achieve a common goal. An agent may in addition possess the ability to migrate from one node to another in a network. Such Mobile Agents carry all their functionalities

with them to enable execution at remote locations. Since the work described herein exploits mobile agents to accomplish data processing and dissemination, a brief description of such agents has been provided in the next subsection.

2.1. Mobile Agents

A mobile agent [15] is basically a piece of code that has the ability to migrate from one node in a network to another and carry out certain task(s). In addition to exhibiting mobility, a mobile agent can also clone and multiply itself, carry a payload (data or a program), make local decisions, execute a program on a remote site or node, etc. Mobile agents can also be used to churn out and carry intelligence along with them as they migrate within a network [16]. They have been used in a wide range of applications which include wireless sensor networks [17], robot control [18,19], e-commerce [20], security [21,22], e-learning [23], robotics [24,25], IoT [10,26], etc. Some of the major advantages of using mobile agents are:

1. *Bandwidth and latency reduction*: A mobile agent has the innate ability to carry the computation in the form of code to a remote site. Instead of fetching the whole raw or unprocessed data from a remote site, the *mobility* allows for the computing program or logic to migrate to this site and process the data therein. This results in reducing network traffic and latency.
2. *Discontinuous operation*: In a dynamic network where the devices are mobile, it is rare that a continuous connection is maintained between two nodes for a long time. In a conventional client-server system, a sudden disconnection may cause the server to resend the whole data, making it an expensive affair. On the contrary, in a mobile agent-based scenario, migration occurs only when a connection is established. The mobile agent then resides in the new node till the connection to the next node is available. Unlike the large amount of data to be processed, a mobile agent is comparatively lightweight. Thus, a failure in migration does not compound into large losses in bandwidth and time.
3. *Adaptivity and flexibility*: In a traditional centralized system, any upgrade would require the system to be brought down, changes made and then restarted. In a mobile agent-based system, upgrades could be packaged within the mobile agent and released into the network. This *On-The-Fly Programming* (OTFP) [10] support facilitates a higher amount of flexibility. Agents have the ability to sense and perceive their environment and change their behaviours accordingly. A mobile agent can add new behaviours in the form of a payload and can also adapt to different situations.

Mobile agents thus have the potential to provide a viable distributed solution to problems related to a network [27].

2.2. Multi-Agent Frameworks

Agent related processes such as its creation, programming, migration, cloning, etc., require a software environment or framework that runs on the supporting hardware platform. A Multi-Agent Framework (MAF) provides for such an environment and facilitates the rapid development and deployment of agent-based systems. These frameworks allow users to create, program and release mobile agents into a network and also aid the execution of the relevant programs within JADE [28], JADE-LEAP [29], TACOMA [30], Agent TCL [31], AgentSpace [32], Aglets [33], etc., are Java based MAFs. Mobile-C is an agent framework that is written purely in C/C++ programming language. Its light footprint makes it ideally suited to small embedded systems. Some of the real world deployments where such frameworks have been used include a taxi booking system developed over JADE-LEAP (Multi-Agent Systems Group, University Rovira i Virgili (URV), Tarragona, Spain) [34], a multi-agent traffic control system [35], etc. C/C++ and Java are basically structural and functional programming paradigms on which such event-based applications are developed. A majority of MAFs are based on such languages and thus do not inherit the semantic structure available in logical

languages such as Prolog [36]. Prolog is widely used in applications involving Artificial Intelligence (AI) [37] techniques, natural language processing [38], intelligent searching in databases [39], rule based logical queries [40], etc. Some of the multi-agent platforms built over logical languages include Jinni [41], ALBA [42], IMAGO [43], Typhon (Robotics Lab., IIT Guwahati, India) [44] and *Tartarus* [10]. In this work, we have used *Tartarus*, a multi-agent platform developed using SWI-Prolog (Version 7.2.3, University of Amsterdam, Amsterdam, Netherlands) [45]. *Tartarus* and its former version Typhon has been used in a variety of applications ranging from multi-robot synchronization [46], learning using sharing [47], realizing a green-corridor for emergency services [48], rescue robots [10], monitoring from a remote base station [49], etc. It thus forms a fitting *cyber* counterpart for applications that involve AI, distributed data processing and search.

3. Location-Aware and Tracking Service (LATS)

Since the work described herein uses LATS as an agent-based application embedded on *Pi*, a brief survey on the same is presented below. Location-dependent services are part of a dynamic model where either the object or the observer or both can be mobile with respect to their geo-location [50]. Some of the classical approaches for tracking of a moving object include the use of GPS, RFID, camera, etc. The most popular method of positioning is by using a GPS on board a mobile phone. This method is however, effective mainly outdoors where the device can reach out to the satellites. Indoor localization using such GPS is unreliable due to the topology of the rooms and the erratic and low intensity satellite signals received within. This calls for an efficient yet cost-effective solution to provide for a reliable indoor positioning and tracking system. Catarinucci et al. [51] have proposed an IoT-aware architecture for smart healthcare. They have leveraged the use of combining UHF RFIDs [52] and WSNs [53] for deploying a healthcare system. Each patient has an RFID tag that transmits its data to an RFID receiver, which, in turn, transmits the data to the associated doctor. Since RFID tags are passive devices, the system uses minimum power and is thus quite efficient in terms of energy consumption. The major drawback is that, for proper data transfer, the patient has to be in very close proximity to the RFID receiver.

Bluetooth Low Energy (BLE) technologies [54] can offer a far more superior solution than RFIDs. Yoshimura et al. [55] portray a system for analyzing the visitors' length of stay in an art museum through the use of non-invasive Bluetooth based monitoring. In their work, eight Bluetooth sensors were installed in the busiest locations at the Denon wing of the Louvre museum. The data on the number of visitors visiting these places was collected for a period of five months and then analyzed to get meaningful results. They have claimed that the use of non-invasive technologies (such as Bluetooth) allows them to gather honest results. This is so since visitors change their behaviours if they are aware of the fact that they are being tracked.

Early work in location-aware services by Wolfson et al. [11] describe a mechanism for tracking moving objects through the use of database. They present a Database for Moving Objects (DOMINO) on top of an existing database, which allows the database management system to predict the future location of the moving object. Every time the object in motion updates its location, its future location is also predicted.

Wolfson et al. [56] has also proposed a trajectory location management system to model the moving object. They highlight the critical issues associated with the point-location management model [56]. A point-location model does not provide facilities for interpolation or extrapolation of location data of the moving object and is not accurate.

In a trajectory location model, an estimate of the source and destination of a moving object is determined. This information is coupled with an electronic map and a trajectory is constructed based on the travel time information. In the real world, the relevant data is not always available at a centralized location. Wolfson et al. [56] conclude that their model needs to be improved to suit scenarios where data is available in a distributed form. In the latter part of this paper, we show how LATS can be implemented when the data is distributed across a network of devices.

4. AgPi: The Cyber and Physical Confluence

The *Pi* is an inexpensive low footprint mini-computing device. It boasts of a System-On-Chip (SOC) architecture that includes a 64-bit microprocessor, a Graphical Processing Unit (GPU) and peripherals, making it compact in size. It can be used in conjunction with a TV or a computer monitor. The *Pi* has a range of peripherals to allow use of input and output devices [57]. It comes in different versions, the more recent ones being the Raspberry Pi 3 (Raspberry Pi Foundation, London, UK) and the Raspberry Pi Zero (Raspberry Pi Foundation, London, UK). The latter Zero version costs just around US $5, possibly making it one of the cheapest and most affordable mini-computing devices [58]. The availability of General Purpose Input and Output (GPIO) pins along with multiplexed I2C, SPI and UART pins, which can be easily accessed through an open source Linux operating system running on it, makes the *Pi* an appropriate device to sense and control an environment. Such high end features allow a user to create and deploy systems in the real world, making the *Pi* an ideal device for IoT applications.

Features such as autonomous decision making, robustness, flexibility, intelligence, etc. which are generally associated with agents are seldom found in current IoT solutions. In this paper, we have described the working of a full-fledged MAS-based IoT application by leveraging the use of *Tartarus* running on *Pi*. The application has been described with a view to enthuse *Pi* developers and users to create new and intelligent applications using the concept of *Agents on Pi* (AgPi). The mobile agent framework used, *Tartarus* plays the role of controlling the *Cyber* entities (agents), which, in turn, command their *physical* counterparts (sensors and actuators). *Tartarus* comes with a dedicated plugin to access peripherals within the *Pi*. This facility provides for a coupling between the *Pi* and its *cyber* counterpart, *Tartarus*. Figure 2 shows how several *Pi*s, each running *Tartarus*, are connected to form a network. It also depicts static agents residing at some nodes and migrating mobile agents. It may be noted that these mobile agents can move over to any node, sense the data within (as also actuate a motor for instance, if required), take a decision and then move on in the network. Figure 2 thus conforms to the agent-based CPS depicted in Figure 1. Unique behaviours could be programmed and embedded within each agent, thus allowing for an autonomous or semi-autonomous control of the *physical* world.

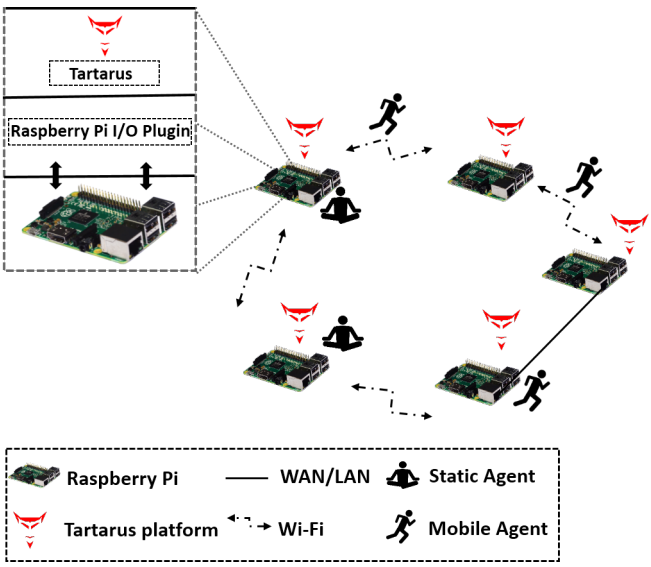

Figure 2. Top level architecture of AgPi (Agents on Pi).

In the next section, we describe an application that will throw more light on the benefits of using agents on a network of *Pi*s. This application is a multi-agent-based distributed and decentralized solution for LATS for an indoor environment using the AgPi concept.

5. AgPi in the Real World

A complex system can be divided into subsystems, each controlled by an agent. This form of abstraction eases the designing and realization of systems. With *Pi* in the scenario, such complex systems can now be deployed as real-world applications. One of these applications for LATS that uses the AgPi concept is described below.

5.1. AgPi based LATS application

As a proof-of-concept, we have implemented an LATS for dynamic tracking of users in a corridor of a building. The following subsections describe the detection mechanism and the main units that comprise the application.

5.1.1. Detection Mechanism

The lower portion of Figure 3 portrays the manner in which Pi-nodes have been deployed along the corridor. A Pi-node consists of a Pi interfaced to a BLE receiver and Wi-Fi adaptor. A Cyber Computing Unit comprising *Tartarus* and its associated plugins runs on the Pi. Each Pi-node within the corridor is connected to its neighbour(s) through Wi-Fi.

A person who is to be tracked (depicted as a stick figure with a red band on the wrist in the figure) needs to wear a BLE tag that emits beacons at a certain rate. This BLE tag along with the Pi-node forms a Wearable and Acquisition Unit (WAU).

Users who need to track a person(s) are provided with a User Interaction Unit (UIU) running on their respective computing machine. The functioning of the WAU, CCU (Bluetooth Low Energy) and UIU shown in the upper portion of the Figure 3 has been detailed in the subsequent subsections. As can be seen in the lower portion of the figure, the corridor is divided into virtual zones (indicated by different colours) whose areas are preset based on the RSS (Received Signal Strength) values from the BLE tag received by the associated Pi-node.

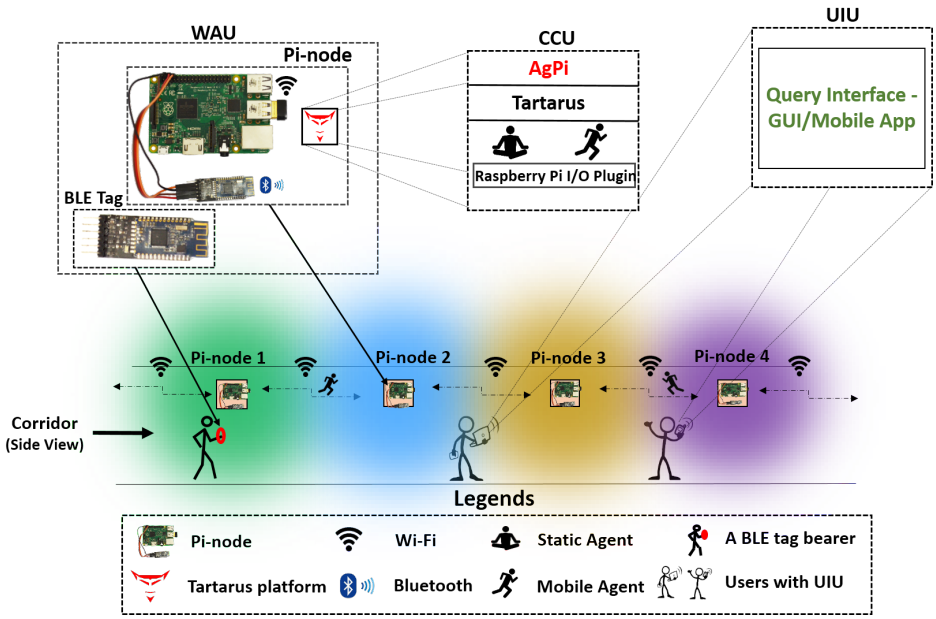

Figure 3. AgPi based LATS (Location-Aware and Tracking Services) application.

When a person enters a zone within the corridor, the BLE receiver of the Pi-node within that zone detects his/her presence in that zone. As the person moves away from this zone and enters the neighbouring one, the RSS in the new zone increases while in the former's decreases. This indicates the transition of the user from one zone to another. Eventually, when the RSS detected at the previous

zone becomes minimum and that at the next zone becomes maximum, the system detects the presence of the person in the latter zone.

5.1.2. Wearable and Acquisition Unit (WAU)

This unit includes a wearable Bluetooth Low Energy (BLE) device (HM-10) that emits data packets in the form of beacons at preset intervals. These packets are received by a BLE receiver interfaced to a *Pi* via its on-board UART module. Figure 4a,b show a BLE tag (comprising a BLE device and a battery) as a wearable unit (configured as a beacon transmitter) and a Pi-node comprising a *Pi* interfaced to a BLE receiver as the acquisition unit. The *Pi* also has a USB Wi-Fi adaptor. Each data packet transmitted by the wearable BLE device is 30 bytes long and contains five fields of information as given below:

1. Preamble: This read-only field is 9 bytes wide and contains the manufacturer's data.
2. Universally Unique Identifier (UUID): This field, which is 16 bytes wide, can be preset to contain the identity of the BLE device.
3. Major: This is a user writable field which helps in identifying a subset of such devices within a large group.
4. Minor: It is also a writable field which is used for specifying a subset of the Major field.
5. Tx Power: This field is a calibrated 2's complement value denoting the signal strength at 1 m from the device. This field is compared with the measured signal strength at the receiving end in order to ascertain the distance between the transmitter and receiver.

The BLE receiver extracts the information within these five fields and forwards it to the CCU.

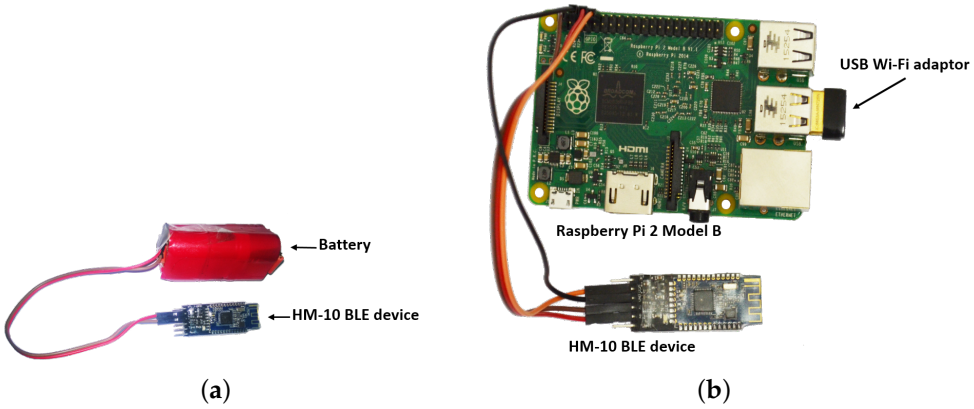

Figure 4. (**a**) a BLE (Bluetooth Low Energy) tag; (**b**) A Pi-node.

5.1.3. Cyber Computing Unit (CCU)

A CPS is a tight coupling between the *physical* and the *cyber* worlds. The *Tartarus* platform serves the purpose of a *cyber* unit which runs on top of the *physical* unit (*Pi* in the present case). *Tartarus* comes with a plugin to access the peripherals on board the *Pi*. A static agent named a Database agent within a *Tartarus* instantiation running on a *Pi* fetches the beacon data from the buffer register within the BLE receiver via the UART [59] interface. The Database agent then stores the data in an SQL database along with the time-stamp on the memory card in the *Pi*. If a user remains within a zone for a long period, there will be a large accumulation of data, most of which could be redundant. To avoid this, beacon data is read always but stored only under some conditions. Thus, data is logged only when there is considerable change in the RSS of the beacon. Furthermore, instead of making decision based on the normally noisy RSS values, three regions—*Beyond*, *Far* and *Near* have been used to describe the position of a user within a zone. The three regions can be defined as follows:

(i) *Beyond*: When the RSS value is zero, it means that the person is not detected and is beyond the concerned zone.
(ii) *Far*: This is a case when the person being tracked is far from the Pi-node. This is detected by a weak RSS value at the Pi-node of the concerned zone and would mean that the person wearing the BLE tag is in between 2 m to 5 m of the radial distance from the associated Pi-node.
(iii) *Near*: A strong RSS value indicates the person to be well within the range i.e., less than 2 m in the present case.

Each SQL entry comprises a total of six fields of information — the Timestamp, UUID, Major, Minor, RSS and Region. A sample snapshot of the data entered at a Pi-node is shown in Figure 5.

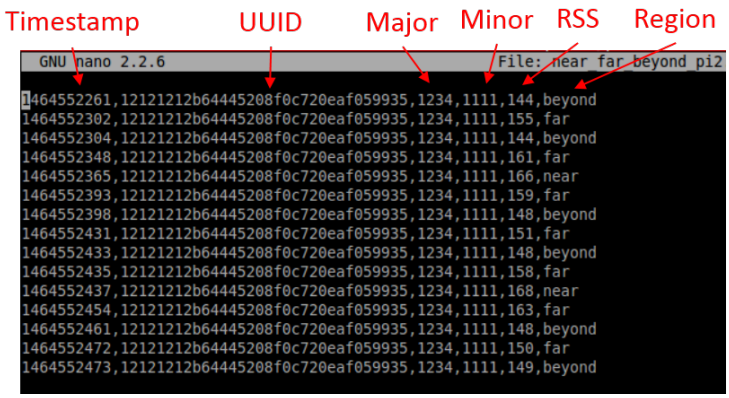

Figure 5. A sample snapshot of the part of the database maintained at a Pi-node.

An entry is made to the SQL database only when the value of the sixth field changes in terms of the Region. For instance, if the sixth field changes from *beyond* to *near*, an entry is logged with the new Region. If the next consecutive entry is also *near*, then no entry to the database is made. Similarly, if the sixth field changes to either *far* or *beyond*, an entry is made. It may be observed that from the database the information about the period of stay of a user in a particular region or zone can be easily computed. Furthermore, a person may also be tracked as s/he moves from one zone to another. One may also easily infer as to exactly when s/he entered a zone, the amount of time spent within that zone and when s/he exited the same. Thus, as a person passes through a corridor comprising several such zones, the respective Pi-nodes keep track of the next zone to which the person has moved. This is done through the concept of a Motion Vector which has been described below.
Motion Vector: Let $Z = Z_{P_1}, Z_{P_2}, Z_{P_3}, \ldots, Z_{P_n}$ be a set of zones, where P_j represents the j^{th} Pi-node and n is the total number of Pi-nodes in the network (one per zone). A Motion Vector $(\overrightarrow{MV})$ describes the movement of a person wearing the BLE tag, from one zone to another and is given by,

$$\overrightarrow{MV} = Z_{P_a} \rightarrow Z_{P_b} \; ; \; a, b \; \epsilon \; \{1, 2, \ldots, n\}.$$

Each Pi-node in a CCU stores and updates two types of Motion Vectors—Motion Vector Forward $(\overrightarrow{MV_F})$ and Motion Vector Backward $(\overrightarrow{MV_B})$. When a person wearing the BLE tag moves from the *far* region to the *beyond* region of a certain zone, say Z_{P_x}, the corresponding Pi-node, P_x within that zone, sends a message to all its neighbouring Pi-nodes announcing that the person bearing the specific UUID is now in the process of leaving its zone Z_{P_x}. If any of the neighbouring Pi-nodes, say P_y, detects this UUID within its zone, Z_{P_y}, it will acknowledge the presence of the person to the Pi-node, P_x. This causes the Pi-node, P_x to update its Motion Vector Forward, $\overrightarrow{MV}_F = Z_{P_x} \rightarrow Z_{P_y}$, against the associated person. Similarly, the Pi-node P_y updates its $\overrightarrow{MV}_B = Z_{P_x} \rightarrow Z_{P_y}$ and $\overrightarrow{MV}_F = Z_{P_y} \rightarrow Z_{P_y}$. The $\overrightarrow{MV}_F = Z_{P_y} \rightarrow Z_{P_y}$ represents a transition from Z_{P_y} to itself. This indicates that the user is currently in that zone and acts as a presence indicator. Table 1 shows the Motion Vectors at Pi-nodes

P_x and P_y after a user transits from zone Z_{P_x} to zone Z_{P_y} (zone Z_{P_x} is assumed to be the very first entry zone). Here, INFINITY represents that a user is not traceable at any of the zones, and thus can be considered to be outside of the infrastructure where agent-based LATS is deployed.

Table 1. Motion vectors at Pi (Raspberry Pi)-nodes P_x and P_y after an inter-zonal transition.

INFINITY $\rightarrow$	**Zone** Z_{P_x} $\rightarrow$	**Zone** Z_{P_y}
MVF	$Z_{P_x} \rightarrow Z_{P_y}$	$Z_{P_y} \rightarrow Z_{P_y}$
MVB	INFINITY $\rightarrow Z_{P_X}$	$Z_{P_x} \rightarrow Z_{P_y}$

The UUID and Major-Minor values allow for classifying a particular BLE device wearer. For example, one can track the faculty members and students in an academic department using the content within these fields. This makes the database contain finer details and thus allow a range of queries to be satisfied. As can be seen, the database agent thus manages the database and the Motion Vectors within the associated Pi-node.

5.1.4. User Interaction Unit (UIU)

This unit provides an interface for the users to access the tracking service of the agent-based LATS. The interface could be in the form of a mobile app or a Graphical User Interface (GUI) running on a *Pi*, a laptop or a PC, all connected to the same network as that of WAU. We have used a *Tartarus* instantiation running on a *Pi* and a laptop to fire queries to the system. To fire a query, a user can release an agent from the same *Tartarus* instantiation. The UIU was populated with mobile agent programs for a set of queries. Since *Tartarus* facilitates agent programming [10], users and developers could write custom mobile agent programs for a range of queries and add them to the UIU to improve its functionality. The code for the agent of the associated query shall be already available with the *Tartarus* as part of the UIU.

Querying: A mobile agent serves the purpose of query processing. Since the databases are distributed over the various Pi-nodes, these mobile agents move from one such node to another and search and retrieve the information that can satisfy the user's query. The mobile agent then aggregates the relevant data concerning the person being tracked and delivers it to the UIU for processing and rendering. A user wearing the BLE device or a third party may wish to query this LATS to gather a range of information which include:

1. *Where am I?*: Such a query invariably emanates from a person who is lost within the building or does not know how to move around or needs to convey his/her bearings to someone else. Under such conditions, the user can fire an SQL query packaged in a mobile agent to the nearest one-hop neighbouring Pi-node. Once the mobile agent enters this Pi-node, it executes its code and eventually lands up in the Pi-node of the zone in which the person is currently present. The agent then retrieves the location information stored a priori within this Pi-node and provides it to the user. A segment of the relevant mobile agent code is presented in Figure 6.
2. *Where is X?*: A query of this kind is required for a person to know whether X is within the building under consideration and, if so, where. This agent-based LATS allows for a non-intrusive mechanism to find the location of X. The user packs this query into a mobile agent and transmits it onto the *Tartarus* platform of the closest *Pi*, the one within the zone s/he is in currently. On reaching this *Pi*, the mobile agent scans the database within it to find whether X is/was in this zone. (i) If it discovers that X is within a particular zone currently, it retrieves the location information from the Pi-node and backtracks its path to the user's system and provides the information on X; (ii) if the agent finds a Motion Vector Forward for X in that zone, then it uses the vector to find the next zone visited by X and migrates to the concerned Pi-node of this zone. It continues to do so until it eventually lands in a Pi-node of a zone where X is currently present.

On reaching this, it retrieves the relevant information and retraces its path back to the user's system to provide the information on X. In case X has left the place, the Motion Vector Forward within the Pi-node in the zone where X was last present will point to INFINITY. The agent would then assume that X is no more in the area and report accordingly to the user; (iii) if no trace of X is found in the database, the mobile agent continues its migration along the Pi-nodes in a conscientious manner [60] (Appendix A) until it eventually finds that X has been within the zone of some *Pi* or left the place. It may be noted that a user who wishes to know the bearings of another can alter his query to extract a range of information on the person being tracked.

3. *Trace(X)*: This query will provide a list of locations associated with all those zones which X visited in order. The query can again be packed into a mobile agent and sent to the network of Pi-nodes to search the individual databases and retrieve the list. A mobile agent algorithm to trace the path of a BLE tag bearer is shown in Algorithm 1 and an example of mobile agent routing for the same is described in Appendix B.

```
Result: Path followed by X ;                          // X is a person whose path is to be traced
Stack S = Empty;
Queue Q = Empty;
while while Path followed by X is not retrieved by Agent ;          // Agent continues the search
                                                     // until the total path traced by X is found
do
    MVF(X) = Motion Vector Forward of X at visited Pi-node, P_v ;
    MVB(X) = Motion Vector Backward of X at visited Pi-node, P_v ;
    if (MVF(X) = Nil) OR (MVB(X) = Nil) ;                  // If trace is not found by the agent
    then
        Select a neighbouring node at random and migrate to it ;          // Agent migrates
                                                                          // to another node
    else
        if (MVF(X) = Z_{P_v} → Z_{P_v}) OR (MVF(X) = Z_{P_v} → INFINITY) ;   // If agent has found last node
                                                                             // visited by X
        then
            insertStack(S , v) ;                  // Agent inserts the node ID into its internal stack
            while X's starting position is not found do
                Use MVB of each earlier visited Pi-nodes to trace back the path;
                insertStack(S , Pi-nodes visited before v );
            end
            Path followed by X = getStack(S);
            return Path followed by X ;                    // Agent returns the path followed by X
        else
            if (MVF(X) = Z_{P_v} → Z_{P_w});          // If Agent finds the intermediate node visited
                                                      // by X
            then
                while X's starting position is not found do
                    Use MVB(X) of each earlier visited Pi-nodes to find the start position ;
                end
                while X's last/current position is not found do
                    Use MVF(X) of each next visited Pi-nodes to reach the last/current position ;
                    insertQueue(Q , Pi-nodes visited from the start position);
                end
                Path followed by X = getQueue(Q);
                return Path followed by X;
            else
            end
        end
    end
end
```

Algorithm 1: An algorithm performed by an agent to trace the path of a BLE tag bearer.

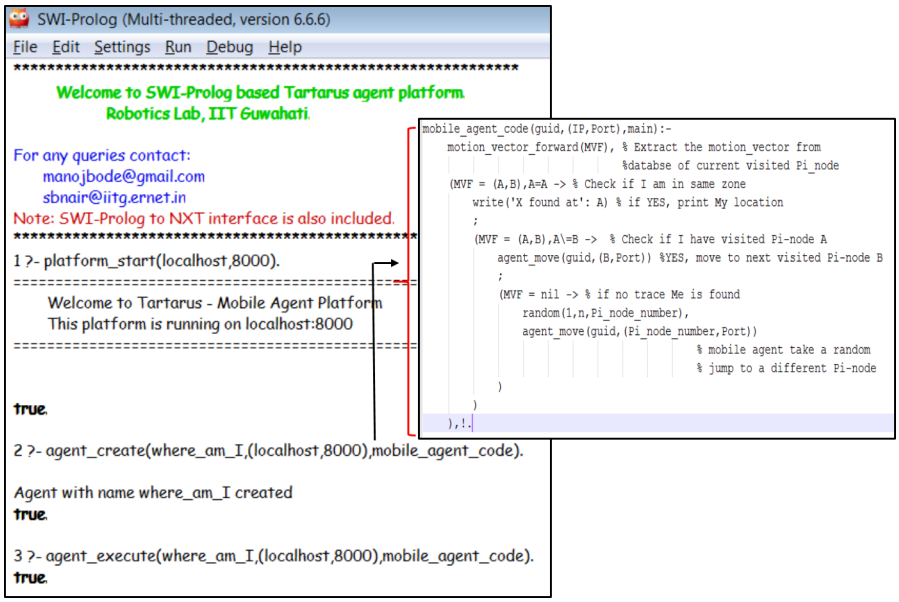

Figure 6. Mobile Agent code snippet for the query, *Where am I?*.

6. Experiments and Results

Experiments conducted involved users who were asked to move from one zone to another. In addition, experiments involving acquisition of raw BLE data were also conducted to get more insights into the behavior of the device. In subsequent sub-sections, we discuss the experiments conducted to acquire and store tracking information, which, in turn, are used and processed by mobile agents to satisfy user queries.

6.1. Data Acquisition

A BLE tag bearer was asked to move back and forth across the radial axis of a Pi-node. The actual RSS values received at the Pi-node nominally ranged from -40 dBm to +20 dBm (depending upon the manufacturer, the actual raw RSS values for a BLE device may range from -80 dbm to +25 dbm). In order to portray the graph in the 1^{st} quadrant for clarity, we biased these values by adding +200 dBm to each of the data points. Figure 7 shows the biased raw and filtered BLE data taken over a certain number of sample points. As expected, a trend similar to a sinusoidal wave can be observed in the figure thereby validating the performance of the BLE. The RSS received from a BLE device is subject to noise due to various reasons such as multi-path propagation, signal absorption, signal interference, etc. Based on the analysis by Faragher et al. [61], different filters may be applied to the raw BLE data. After a series of empirical experimentation on data filtering, it was found that a moving average filter with a window size of 6 samples at a time provides satisfactory results. Analysis revealed a rule of thumb that indicates that as the window size increases, the filtered data becomes more stable. However, this may take more time to produce tracking results. Hence, a compromise needs to be made in terms of accuracy and reactiveness of the deployed tracking system.

An experiment wherein each user was made to wear a BLE tag and asked to move from one zone to another in order to obtain their respective tracking profiles was performed. The experiment was conducted at the ground floor of the Department of Computer Science and Engineering block of the Indian Institute of Technology Guwahati. Since it is logical to assume that the profile generated between two consecutive zones can be extended to other such multiple consecutive zones, we describe herein the inter-zonal movement for a single user. The results portraying a user's movement within two zones, Z_{P_1} and Z_{P_2} along with the three regions, *beyond*, *far* and *near*, categorized on the basis of RSS is shown is shown in Figure 8. As in Figure 7, the y-axis denotes the filtered and biased

RSS values from the BLE receiver at the Pi-node, while the x-axis indicates the sampling index ranging from 1 to the number of samples taken at a sampling rate of 1 s.

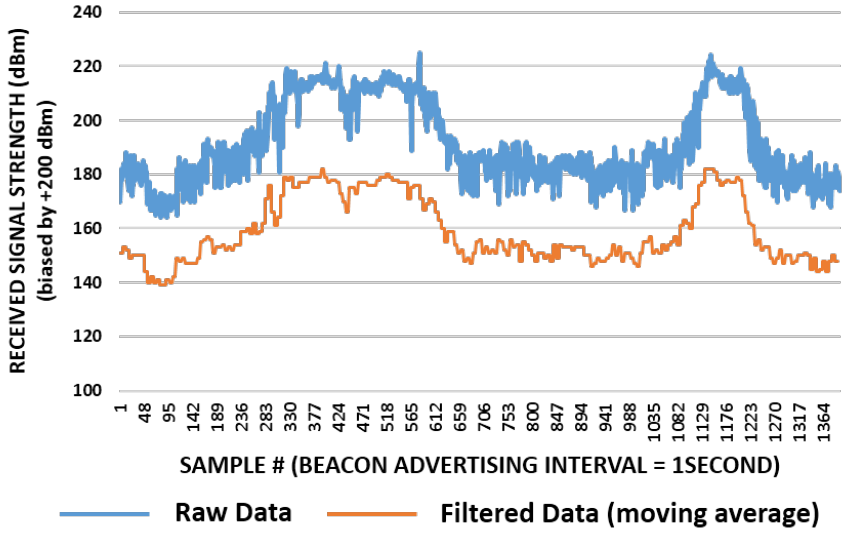

Figure 7. BLE raw and filtered data.

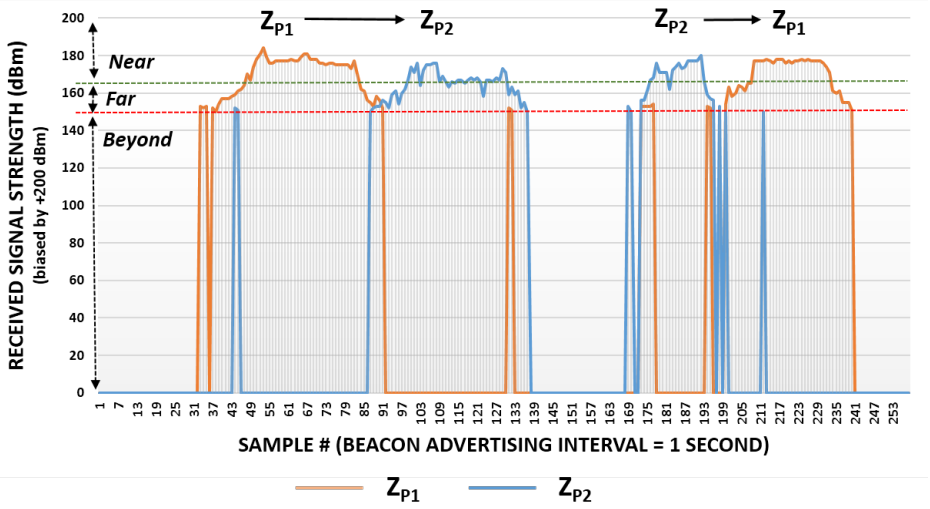

Figure 8. Graph showing inter-zonal movement for a single BLE tag bearer.

The graph shows two different coloured series each corresponding to the RSS at Pi-node within a particular zone. The orange coloured series denotes the same for Zone 1 (Z_{P_1}) while the blue coloured series indicates that for Zone 2 (Z_{P_2}). Initially, the user is outside the coverage area of both Z_{P_1} and Z_{P_2}. As seen from Figure 8, when the user starts moving towards Z_{P_1}, the RSS (orange colour) increases from sample number 41 onwards and attains a maximum when the user is nearest to the associated Pi-node of Z_{P_1}. It then starts to decrease as the user moves away from the Pi-node in Z_{P_1}. When the user enters the periphery of Z_{P_2}, where both the zones overlap to an extent, an increase in the RSS at Z_{P_2} is observed with a corresponding decrease of the same at Z_{P_1}. A similar pattern is exhibited when the user moves away from Z_{P_2} to the next neighbouring zone. A similar experiment that was conducted when the person moved from Z_{P_2} to Z_{P_1} is recorded with Z_{P_2} as entrance zone and Z_{P_1} as the exit zone. The relevant plots are depicted in the latter part of Figure 8. It may be observed that there are some random spikes generated due to noise and reflections. Since these peaks cross from

the *beyond* region to the *far* region and again go back within a second, the corresponding vectors are not stored in the database.

6.2. Query Processing

The post data acquisition step involves satisfying queries fired from the user side. In order to compare the results of query processing using the conventional cloud-based method and the distributed AgPi approach, we conducted experiments for the two scenarios described in this section. Since testing on a real system would mean the requirement of a large number of *Pi*s, for both of the experiments, we emulated a multi-floor building using a 50-node overlay network [62] using *Tartarus*, formed over a network of four Pi-nodes and two PCs. Each PC hosted 23 emulated Pi-nodes. The BLE tag bearers who move around in the building and need to be tracked, were emulated by mobile agents that move from one node to another. A total of 10 BLE tag bearer were introduced into the network, out of which six were made to move randomly within the building. The remaining four, designated as Head, Professor, Janitor and Guard, were programmed to have predefined movements. In addition, a separate dedicated server acted as the Cloud for both the systems. Figure 9 portrays the conceptual layout of the network deployed in a building. For the conventional cloud-based method, the Pi-nodes may or may not be connected to one another. For the AgPi approach (as shown in Figure 9), these connections are mandatory since there needs to be paths for the mobile agents to migrate.

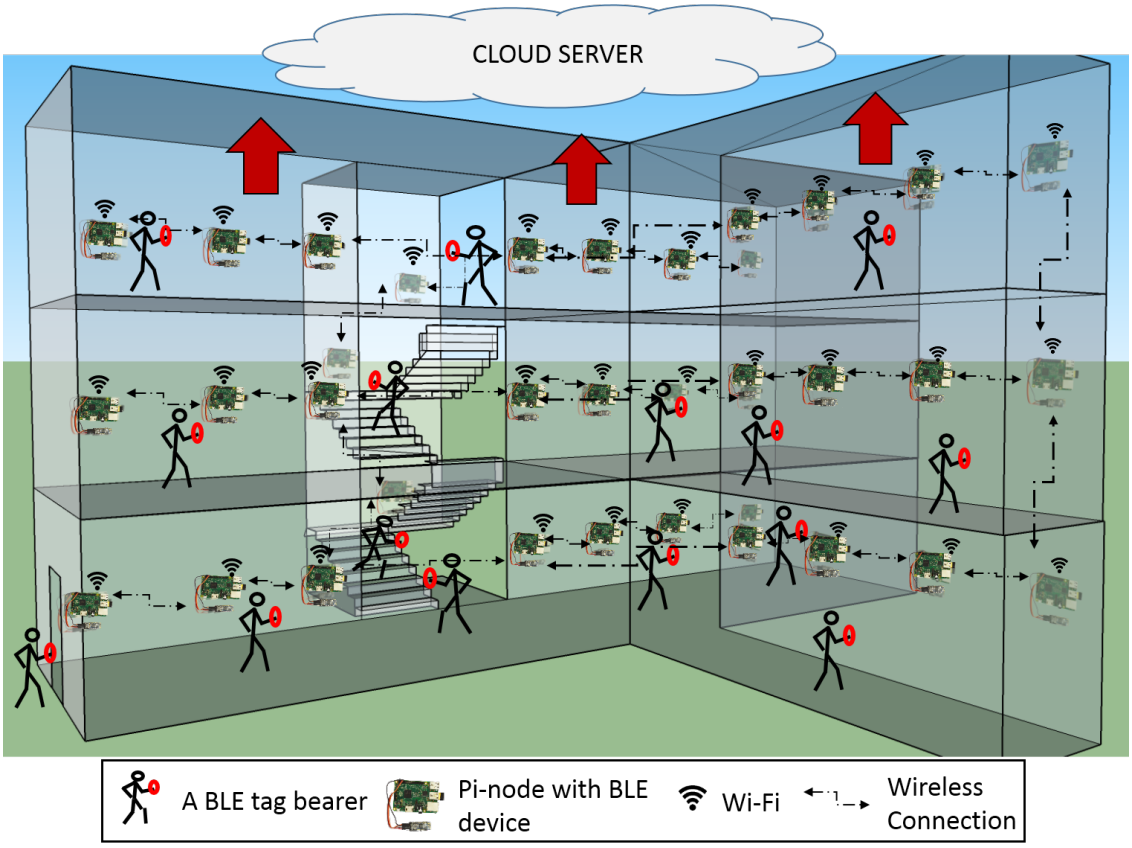

Figure 9. AgPi deployed in a multi-floor building.

Scenario 1: Conventional Cloud Approach

In this approach, every Pi-node was capable of directly communicating with the Cloud. As the BLE tag bearers (mobile agents) move around the building (network), all pertinent data within the Pi-node (such as Timestamp, UUID, Motion Vectors, etc.) are directly sent to the Cloud. This is

done by each of the Pi-nodes as and when new data is generated within them. Thus, all the data acquired and generated at the Pi-nodes is stored and managed at the cloud. All queries in this scenario are directly sent to the cloud, which are, in turn, processed at the cloud and returned to the concerned user.

Scenario 2: AgPi Scenario

In this scenario, a user fires a query in the form of a program within a mobile agent via the UIU. This agent then knits through the connected Pi-nodes in the network, performs the concerned task(s) and processes the data within these nodes, thereby processing the query. While doing so, it also sends the acquired data at each node to the cloud. It may be noted that, in this case, the cloud is updated only with the relevant information pertaining to the query. Unlike the previous centralized scenario, the cloud connectivity is made only from those Pi-nodes where the mobile agent finds query related information. This drastically reduces data traffic between the networked devices and the cloud.

Comparison of Scenario 1 with Scenario 2

Experiments were performed where queries were fired by the user in both the centralized cloud-based and AgPi scenarios. Data transfer cost in terms of the number of times the Pi-nodes connect to the cloud was logged in both these cases. Figure 10 shows the cumulative number of connections made between the Pi-nodes and the cloud server for a set of Trace (X) queries, where *X* is the person being tracked.

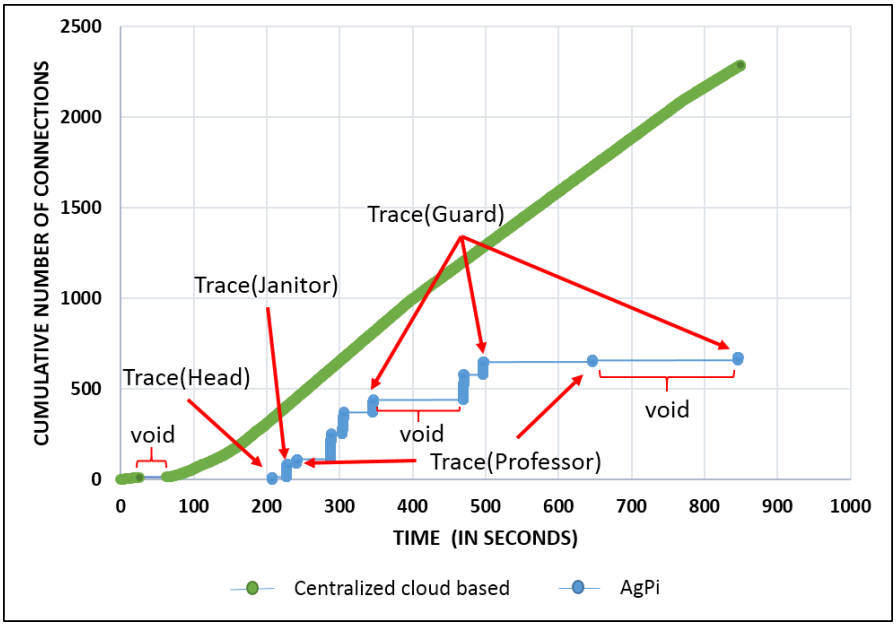

Figure 10. Traffic flow for centralized cloud-based and AgPi systems.

In the case of a centralized cloud-based approach, one can infer that the cumulative number of connections made to the cloud increases steadily with time. As mentioned earlier, this is because, for every new data generated at a Pi-node, a connection is made to the cloud.

On the contrary, in the case of AgPi, such connections are made to the cloud only when information found is relevant to the query fired. The cumulative number of connections made by the mobile agents during the execution of the queries Trace (Head), Trace (Janitor), Trace (Guard) and Trace (Professor) are shown in the figure. These numbers are far lower than that for the centralized scenario, clearly indicating the viability of the AgPi approach. The horizontal flat portions, termed as

voids in the curves, denote the absence of any connections made to the cloud. Such portions could occur in the centralized scenario when the BLE tag bearers are stationary i.e., when no new data is generated at the Pi-nodes. For the AgPi scenario too, such voids could occur provided no queries are fired.

7. AgPi: Applications Envisaged

The concept of AgPi opens up a plethora of areas where the characteristics of agents, both static and mobile, can be exploited. As mentioned earlier, our stress in this paper is to generate more interest in the use of agents on *Pi* and encourage the creation of real and working systems. The following are some of the areas where agents and *Pi* could blend well to produce such systems:

Health Care: One of the most sensitive areas where AgPi can be envisaged is in the unobtrusive monitoring of the health of a patient. For instance, a wearable Bluetooth based wrist band equipped with a pulse monitor, temperature sensor, pedometer and similar medical sensors could feed the live data to an agent on a *Pi*. This static agent could be programmed to cross check this data, make a report on the status of the health of the person and then forward the same to a doctor, if required. In an alternative scenario, a mobile agent can be programmed to continuously patrol the network of such Pi-nodes set up in a hospital, gather the health information and status of the concerned patients and deliver the reports to the concerned doctors on their mobile devices. Mobile agents could also track, trace and inform a doctor in case of an emergency. With AgPi on an Intranet of Things in a health care scenario, information could be filtered and then sent to a cloud, thereby ensuring the integrity and security of a patient's medical data.

Vehicular Networks: The concept of Vehicular Networks (VANET) and connected cars have opened up numerous application domains. Vehicles, which constitute a node in the network, could have an on-board *Pi* with all networking facilities. Such vehicles can form a network among themselves to allow inter-vehicle communications. This can aid in solving a variety of problems associated with urban traffic conditions. Agents within such Pi-nodes can migrate around the network to learn about the traffic conditions in advance and provide valuable route information to the driver. Such information could also be disseminated to other cars by the agents. Agents can also aid in the generation of partial green-corridors for the movement of emergency vehicles [48].

Robotics: An intranet of Pi-nodes connected to a network of robots can aid the latter in carrying out tasks in a coordinated manner. Semwal et al.[10] portray how agents can search and guide a rescue robot to an area where they are required. Mobile agents can also be used to synchronize tasks performed by a set of robots [46]. Sharing information using mobile agents can facilitate learning from within a network as has been described by Jha et al. [47].

The concept of AgPi thus can be used to churn out as well as embed intelligence in a network of embedded systems.

8. Conclusions

The paper introduces the concept of *Agents on Pi* (AgPi) and describes how it can be realized by using *Tartarus*, a multi-agent platform running on a Raspberry Pi. The platform supports both static and mobile agents and also allows them to access and control the various ports and peripheral devices on the *Pi* through a dedicated plugin. This allows for the creation of a CPS that has both the hardware and software constituents. An agent-based Location-Aware and Tracking Service (LATS) application has also been described to bring out the versatility of using agents on a *Pi*. The system can track people wearing a BLE device indoors with fair reliability and accuracy and also provide answers to a range of queries as regards the person being tracked. Experimental results reveal that the

AgPi approach seems to perform better than the conventional cloud-based method. In addition, the use of mobile agents allows multiple queries to be fired using multiple agents concurrently. Queries need not be pre-programmed or preset. Since mobile agents can be released even during run time, these queries can be fired on-the-fly. The use of agents on *Pi* can thus make a network of things smarter and flexible unlike those that do not use agents. Since agents in *Tartarus* can be created and released even during run-time [10], the AgPi based system can be scaled, upgraded and programmed to be adaptable. The range of diverse areas where agents have been used until the date makes the concept of AgPi a powerful mechanism to realize intelligent applications in the realm of embedded and connected devices.

Acknowledgments: The first author would like to acknowledge Tata Consultancy Services (TCS) and the Ministry of Human Resource Development (MHRD), Govt. of India for providing the support during the research reported in this paper.

Author Contributions: Both the authors, Tushar Semwal and Shivashankar B. Nair, together conceived the idea of AgPi based indoor localization portrayed in this paper. The making of the hardware test-bed and the associated programming were carried out by Tushar Semwal. Both authors were involved in the analysis of the data churned out, the interpretations of the outcome and also in the writing of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Pi: Raspberry Pi
AgPi: Agents on Pi
BLE: Bluetooth Low Energy
IoT: Internet of Things
CPS: Cyber-Physical System
MAS: Multi-Agent Systems
GPS: Global Position System
LATS: Location-Aware and Tracking Services
SQL: Structured Query Language
RSS: Received Signal Strength
PC: Personal Computer (Desktop)
UHF: Ultra High Frequency
RFID: Radio Frequency Identification
I2C: Inter-Integrated Circuit
UART: Universal Asynchronous Receiver Transmitter
SPI: Serial Peripheral Interface
WSN : Wireless Sensor Network

Appendix A. Conscientious Migration Strategy

In the Conscientious Migration Strategy [60], the mobile agents migrate to the neighbouring node only when that node has not been not visited or happens to be the least visited one. In order to keep track of the visited nodes, a mobile agent appends the recently visited node to a list, comprising the nodes already visited, maintained within itself. Thus, before moving to the next node, an agent checks if the next visited node is a member of this list. If so, it chooses another neighbouring node or the least visited neighbour.

Appendix B. Query Processing

Figure A1 shows 10 Pi-nodes connected to each other in a topology similar to the geographical map of a floor of a building. The 10 Pi-nodes are denoted by $a, b, c, d, e, f, g, h, i, j$ and their

corresponding zonal areas by Z_{P_a}, Z_{P_b},.., and Z_{P_j}, respectively. For the sake of simplicity, only part of the database relevant to agent routing is shown in each Pi-node. Let us assume that the path followed by a BLE tag bearer X is: $d \rightarrow e \rightarrow f \rightarrow i \rightarrow h$.

In the figure, MVF(X) and MVB(X) denotes Moving Vector Forward and Moving Vector Backward for X, respectively. Imagine a user fires a query from node a to trace X. The associated mobile agent now has three neighbouring nodes (b, c and d) to migrate. Since there is no motion vector for X in the node a, the mobile agent opts for the conscientious strategy and chooses one of these neighbours. Assume that it chooses node b and migrates to it. Since the motion vectors for X are absent in b, the conscientious strategy forces it to backtrack to a. If it now selects node d and migrates to it, the motions vectors of X within d will force the agent to switch to the strategy of following motion vectors. From d onwards, the motion vectors will guide the agent through nodes e, f, i and h in that order and thereby retrieve the trace for X.

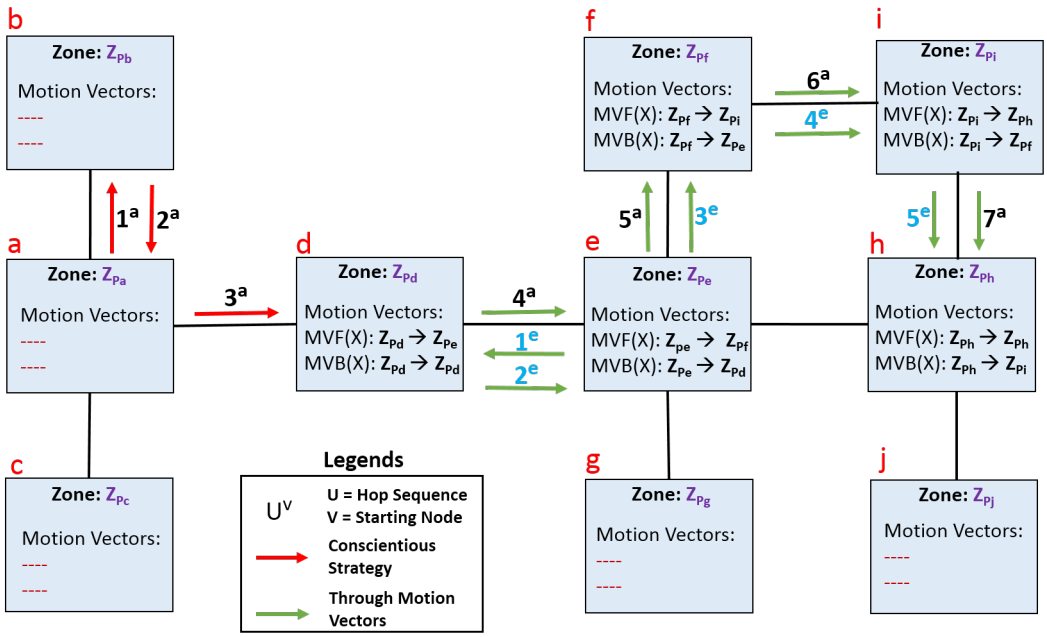

Figure A1. Agent migration an AgPi network.

If the query was fired from node e, since the MVB for X point to node d, the agent migrates to d, after which it follows the MVFs to discover the trace $d \rightarrow e \rightarrow f \rightarrow i \rightarrow h$ as in the previous case.

Handling Failures

In the current implementation of AgPi system, handling node failures are implemented in a very naive manner. A simple handshake protocol is used where mobile agents first ping the neighbouring node. If an acknowledgment is received from this node, the mobile agent migrates to that node; otherwise, it informs a local server about this potentially faulty node and chooses another neighbouring node to migrate.

References

1. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805.
2. Gartner. Gartner Says By 2020, More Than Half of Major New Business Processes and Systems Will Incorporate Some Element of the Internet of Things, 2016. Available online: http://www.gartner.com/newsroom/id/3185623 (accessed on 3 June 2016).

3. Schreyer, P. The contribution of information and communication technology to output growth. *OECD Sci. Technol. Ind. Working Pap.* **2000**, doi:10.1787/18151965.
4. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660.
5. Khan, R.; Khan, S.U.; Zaheer, R.; Khan, S. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In Proceedings of the 10th International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, 17–19 December 2012; pp. 257–260.
6. Vaquero, L.M.; Rodero-Merino, L. Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 27–32.
7. Wooldridge, M. *An Introduction to MultiAgent Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
8. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog Computing and Its Role in the Internet of Things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC' 12, Helsinki, Finland, 17 August 2012; ACM: New York, NY, USA, 2012; pp. 13–16.
9. Calle, M.; Kabara, J. Measuring Energy Consumption in Wireless Sensor Networks Using GSP. In Proceedings of the 2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications, Helsinki, Finland, 11–14 September 2006; pp. 1–5.
10. Semwal, T.; Bode, M.; Singh, V.; Jha, S.S.; Nair, S.B. Tartarus: A Multi-Agent platform for integrating Cyber-Physical Systems and Robots. In Proceedings of the 2015 Conference on Advances in Robotics, Goa, India, 2–4 July 2015.
11. Wolfson, O.; Sistla, P.; Xu, B.; Zhou, J.; Chamberlain, S.; Yesha, Y.; Rishe, N. Tracking moving objects using database technology in DOMINO. In *Next Generation Information Technologies and Systems*; Springer: Berlin, Germany, 1999; pp. 112–119.
12. Franklin, S.; Graesser, A. Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents. In *Intelligent Agents III Agent Theories, Architectures, and Languages*; Springer: Berlin, Germany, 1996; pp. 21–35.
13. Maes, P. Artificial Life Meets Entertainment: Lifelike Autonomous Agents. *Commun. ACM* **1995**, *38*, 108–114.
14. Ferber, J. *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1999.
15. White, J.D.; Davies, M.; Mcgeachie, J.; Grounds, A.D. Mobile Agents. In *Software Agents*; AAAI/MIT Press: Palo Alto, CA, USA, 1997; pp. 437–472.
16. Harrison, C.G.; Chess, D.M.; Kershenbaum, A. *Mobile Agents: Are They a Good Idea?* IBM TJ Watson Research Center Yorktown Heights: New York, NY, USA, 1995.
17. Chen, M.; Gonzalez, S.; Leung, V. Applications and design issues for mobile agents in wireless sensor networks. *IEEE Wirel. Commun.* **2007**, *14*, 20–26.
18. Kambayashi, Y.; Takimoto, M. Higher-order mobile agents for controlling intelligent robots. *Int. J. Intell. Inf. Technol. (IJIIT)* **2005**, *1*, 28–42.
19. Takimoto, M.; Mizuno, M.; Kurio, M.; Kambayashi, Y. Saving energy consumption of multi-robots using higher-order mobile agents. In *Agent and Multi-Agent Systems: Technologies and Applications*; Springer: Wroclaw, Poland, 2007; pp. 549–558.
20. Maes, P.; Guttman, R.H.; Moukas, A.G. Agents That Buy and Sell. *Commun. ACM* **1999**, *42*, 81–91.
21. Boukerche, A.; Machado, R.B.; Jucá, K.R.; Sobral, J.B.M.; Notare, M.S. An agent-based and biological inspired real-time intrusion detection and security model for computer network operations. *Comput. Commun.* **2007**, *30*, 2649–2660.
22. Machado, R.B.; Boukerche, A.; Sobral, J.; Juca, K.; Notare, M. A hybrid artificial immune and mobile agent intrusion detection based model for computer network operations. In Proceedings of the 19th IEEE International Proceedings on Parallel and Distributed Processing Symposium, 4 April 2005; pp. 1–8.
23. Kawamura, T.; Sugahara, K. A Mobile Agent-Based P2P e-Learning System. *IPSJ J.* **2005**, *46*, 222–225.
24. Godfrey, W.W.; Nair, S.B. An Immune System Based Multi-robot Mobile Agent Network. In *Artificial Immune Systems*; Springer: Berlin, Germany, 2008; pp. 424–433.
25. Godfrey, W.W.; Nair, S.B. Mobile Agent Cloning for Servicing Networked Robots. In *Principles and Practice of Multi-Agent Systems*; Springer: Berlin, Germany, 2010; pp. 336–339.

26. Godfrey, W.W.; Jha, S.S.; Nair, S.B. On a mobile agent framework for an internet of things. In Proceedings of the International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India, 6–8 April 2013; pp. 345–350.
27. Satoh, I. MobileSpaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system. In Proceedings of 20th International Conference on Distributed Computing Systems, Taipei, Taiwan, 10–13 April 2000; pp. 161–168.
28. Bellifemine, F.; Poggi, A.; Rimassa, G. JADE: a FIPA2000 compliant agent development environment. In Proceedings of the Fifth International Conference on Autonomous Agents, Madrid, Spain, 10–11 September 2001; pp. 216–217.
29. Bellifemine, F.L.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE*; Volume 7; John Wiley & Sons: Hoboken, NJ, USA, 2007.
30. Johansen, D.; Renesse, R.; Schneider, F.B. *An Introduction to the TACOMA Distributed System*; Technical Report; University of Tromsø and Cornell University: Ithaca, NY, USA, 1995.
31. Kotz, D.; Gray, R.; Nog, S.; Rus, D.; Chawla, S.; Cybenko, G. Agent Tcl: Targeting the needs of mobile computers. *IEEE Internet Computing* **1997**, *1*, 58–67.
32. Silva, A.; Da Silva, M.M.; Delgado, J. An overview of AgentSpace: Next-generation mobile agent system. In *Mobile Agents*; Springer: Berlin, Germany, 1998; pp. 148–159.
33. Lange, D.B.; Oshima, M.; Karjoth, G.; Kosaka, K. Aglets: Programming mobile agents in Java. In *Worldwide Computing and Its Applications*; Springer: Berlin, Germany, 1997; pp. 253–266.
34. Moreno, A.; Valls, A.; Viejo, A. *Using JADE-LEAP to implement agents in mobile devices;* Universitat Rovira i Virgili: Tarragona, Spain, 2003.
35. Chen, B.; Cheng, H.H.; Palen, J. Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems. *Transp. Res. Part C Emerg. Technol.* **2009**, *17*, 1–10.
36. Clocksin, W.; Mellish, C.S. *Programming in PROLOG*; Springer Science & Business Media: Berlin, Germany, 2003.
37. Bratko, I. *Prolog Programming for Artificial Intelligence*; Pearson Education: Upper Saddle River, NJ, USA, 2001.
38. Gal, A.; Lapalme, G.; Saint-Dizier, P.; Somers, H. *Prolog for Natural Language Processing*; Wiley: Chichester, England, 1991.
39. Ceri, S.; Gottlob, G.; Wiederhold, G. Efficient database access from PROLOG. *IEEE Trans. Softw. Eng.* **1989**, *15*, 153–164.
40. van der Wilt, K. Knowledge systems and Prolog: A logical approach to expert systems and natural language processing. *Mach. Transl.* **1990**, *4*, 329–331.
41. Tarau, P. Jinni: Intelligent mobile agent programming at the intersection of Java and Prolog. In Proceedings of Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM), London, UK, 19–21 Apirl 1999; Volume 99, pp. 109–123.
42. Devèze, B.; Chopinaud, C.; Taillibert, P. Alba: A generic library for programming mobile agents with prolog. In *Programming Multi-Agent Systems*; Springer: Berlin, Germany, 2006; pp. 129–148.
43. Li, X. Imago: A Prolog-based system for intelligent mobile agents. In *Mobile Agents for Telecommunication Applications*; Springer: Berlin, Germany, 2001; pp. 21–30.
44. Matani, J.; Nair, S.B. Typhon - A mobile agents framework for real world emulation in Prolog. In *Multi-Disciplinary Trends in Artificial Intelligence*; Springer: Berlin, Germany, 2011; pp. 261–273.
45. Wielemaker, J.; Schrijvers, T.; Triska, M.; Lager, T. Swi-prolog. *Theory and Practice of Logic Programming* **2012**, *12*, 67–96.
46. Jha, S.S.; Godfrey, W.W.; Nair, S.B. Stigmergy-Based Synchronization of a Sequence of Tasks in a Network of Asynchronous Nodes. *Cybern. Syst.* **2014**, *45*, 373–406.
47. Jha, S.S.; Nair, S.B. On a Multi-agent Distributed Asynchronous Intelligence-Sharing and Learning Framework. In *Transactions on Computational Collective Intelligence XVIII*; Springer: Berlin, Germany, 2015; pp. 166–200.
48. Bode, M.; Jha, S.S.; Nair, S.B. A Mobile Agent-based Autonomous Partial Green Corridor Discovery and Maintenance Mechanism for Emergency Services amidst Urban Traffic. In Proceedings of the First International Conference on IoT in Urban Space, Rome, Italy, 27–28 October 2014; pp. 13–18.

49. Semwal, T.; Nikhil, S.; Jha, S.S.; Nair, S.B. TARTARUS: A Multi-Agent Platform for Bridging the Gap Between Cyber and Physical Systems (Demonstration). In Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems, Singapore, 9–13 May 2016; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2016; pp. 1493–1495.
50. Kinnunen, J.; Krishnamurthy, G.; Huhtanen, K.; Jussila, P.; Ratschunas, K. Location Dependent Services. U.S. Patent 6,813,501, 2 November 2004.
51. Catarinucci, L.; De Donno, D.; Mainetti, L.; Palano, L.; Patrono, L.; Stefanizzi, M.L.; Tarricone, L. An IoT-Aware Architecture for Smart Healthcare Systems. *IEEE Internet Things J.* **2015**, *2*, 515–526.
52. Dobkin, D.M. *The RF in RFID: UHF RFID in Practice*; Newnes: Burlington, VT, USA, 2012.
53. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 2292–2330.
54. Gomez, C.; Oller, J.; Paradells, J. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors* **2012**, *12*, 11734–11753.
55. Yoshimura, Y.; Krebs, A.; Ratti, C. An analysis of visitors' length of stay through noninvasive Bluetooth monitoring in the Louvre Museum. *arXiv* **2016**, 1605.00108.
56. Wolfson, O.; Chamberlain, S.; Kalpakis, K.; Yesha, Y. Modeling moving objects for location based services. In *Developing an Infrastructure for Mobile and Wireless Systems*; Springer: Berlin, Germany, 2001; pp. 46–58.
57. Raspberry Pi — Wikipedia. Available online: https://en.wikipedia.org/wiki/Raspberry_Pi (accessed on 2 June 2016).
58. Raspberry Pi Zero: The $5 Computer. Available online: https://www.raspberrypi.org/blog/raspberry-pi-zero/ (accessed on 3 June 2016).
59. Michael, M.S. Universal Asynchronous Receiver/Transmitter. U.S. Patent 5,140,679, 18 August 1992.
60. Minar, N.; Kramer, K.; Maes, P. Cooperating mobile agents for mapping networks. In Proceedings of the First Hungarian National Conference on Agent Based Computing, Budapest, Hungary, 29–31 May 1998.
61. Faragher, R.; Harle, R. An analysis of the accuracy of bluetooth low energy for indoor positioning applications. In Proceedings of the 27th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+'14), Tampa, FL, USA, 8–12 September 2014.
62. Lua, E.K.; Crowcroft, J.; Pias, M.; Sharma, R.; Lim, S. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Commun. Surv. Tutor.* **2005**, *7*, 72–93.

© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).