International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:12, 2015

AMBICOM: An Ambient Computing Middleware Architecture for Heterogeneous Environments

Ekrem Aksoy, Nihat Adar, Selçuk Canbek

***Abstract*—**Ambient Computing or Ambient Intelligence (AmI) is emerging area in computer science aiming to create intelligently connected environments and Internet of Things. In this paper, we propose communication middleware architecture for AmI. This middleware architecture addresses problems of communication, networking, and abstraction of applications, although there are other aspects (e.g. HCI and Security) within general AmI framework. Within this middleware architecture, any application developer might address HCI and Security issues with extensibility features of this platform.

Keywords---AmI, ambient computing, middleware, distributed-systems, software-defined networking.

I. Introduction

TODAY almost everyone own mobile or fixed devices that help them to accomplish tasks in order to ease daily life. In near future, those devices surrounding (i.e. interacting) people will increase in both number and capabilities. Ambient Intelligence (AmI) studies aim the problem of creating some form of intelligence into this interactivity. For instance, devices with sensing (e.g. temperature or humidity measuring) or acting capabilities (e.g. controlling valves, switches or handling items, etc.) that might have "Intelligence" could sense or act without interrupting day life of person, while easing tasks. In order to achieve this intelligence, devices should have (1) situational awareness; (2) discovery of other devices; (3) collaboratively compute features. On the other hand, AmI also incorporates Human-Computer Interaction and Security issues. In this paper, we are going to propose middleware architecture aims to address communication and collaboratively computing problems. The middleware architecture leaves sensor input, HCI and security aspects to application developer where these components are mostly application-specific.

In Section II, an overview of problems and related topics will be presented. Then, the general architecture will be presented. In Section IV, details of components within architecture will be presented. In Section V, related works will be covered.

II. Problems and Related Topics

A. Managing Communication Complexity and Interoperability

Ambient devices require ubiquitous computing capabilities. Since computing power and communication capabilities of these devices are increasing while size form-factors are decreasing, it is obvious that in near future many day life objects a person interact with will be increase.

The main problem of creating such a connected environment is to be able to handle different way of communications for different communication requirements. These differences may arise at different levels of OSI model [1]. Today, Computer Networking practice manages this "fragmentation" with covering functionalities within bag of protocols and embedding these functions into devices capabilities [2].

Embedding each function into a device capability creates complexity of management and interoperability. For instance, a communication link cannot be established with two devices both having Bluetooth protocol stack but not having the same profile (interoperability problem). On the other hand, an enterprise device network might require monitoring of traffic flow and intercepting a suspicious flow (managing communication at large problem) [3]. In both cases, current toolset is not adequate enough to keep simplicity.

Another problem of establishing communication network is to be able to proxying (or delegation) of traffic among different sub-networks. Although this problem also implies reliability issues, communication aspects of this problem also have to be handled within management of network [4].

In general, AmI devices form two communication networks from a device's perspective: (1) Line of Sight (LoS) network (a network of direct communication can be established) or (2) Beyond Line of Sight (BloS) network. LoS network is formed with devices within neighborhood. Neighborhood in this aspect is created with ability to communicate, thus a device having more protocols (i.e. capabilities) is highly possible to have large neighborhood. BloS network is formed with delegation. A device might proxy communication traffic to its neighborhood on behalf of a device that cannot reach those devices within its neighborhood. BloS communication is important for reasons like reliability and interoperability. For instance, a device A might require capabilities of device C and form a communication link over device B. Otherwise, device C's capabilities is not at service for device A.

Managing such a complex network while keeping interoperability, requires new approaches to Networking like

Ekrem Aksoy and Nihat Adar are with the Department of Computer Engineering, Eskisehir Osmangazi University, Eskisehir, Turkey (e-mail: eaksoy@ogu.edu.tr, nadar@ogu.edu.tr).

Selçuk Canbek is with the Department of Computer Engineering, Eskisehir Osmangazi University, Eskisehir, Turkey and the Department of Computer Engineering, Ahmet Yesevi University, Turkestan, Kazakhstan (e-mail: selcuk@ogu.edu.tr).

Network Function Virtualization (NFV) or Software-defined Networking (SDN) [5].

B. Context-Awareness with Semantics

In order to accomplish any given computing task in a connected environment, devices should be context-aware. Context-awareness might be implemented statically (i.e. application-specific perspectives) or dynamically (i.e. device discovers the context within). Although static context awareness could be implemented easily, the flexibility, reliability, and extensibility of such applications fall short of those having dynamic context discovery. In static context-awareness, a platform has to be handling application stacks where in dynamic context-awareness either application or platform is adaptive to context changes. There are studies covering both schemes. Specifically, for dynamic context discovery, Semantic technologies could be used (1) to create ontologies for devices to use discovery of context, and (2) to describe capability requirements of tasks and capabilities of devices. Moreover, micro-kernel-like architectures [6] or Sensor-network OSes [7], [8] define static scheme for application stacking (hence, context-awareness).

A general AmI framework should support both schemes of context discovery as proposed in related works [9], [10].

C. Reliability

The most prominent feature of an AmI platform is reliability. The pervasive and ubiquitous nature of AmI challenges requirement of any given task guaranteed to be accomplished if enough capabilities exist within environment.

The nature of AmI might require additional focus on Quality of Service (QoS) oriented computing, since different applications require different QoS requirements. Moreover, QoS scheme within platform has to be flexible and dynamic.

Fortunately, NFV and related studies [11], [12] provide scheme for QoS in communication as well as OMG DDS [13] like studies are provide QoS mechanisms within Distributed Systems (DS) framework. In addition, when reduced into a DS problem, almost all reliability schemes are covered at Birman's work [14].

III. AMBICOM MIDDLEWARE ARCHITECTURE

AMBICOM, Ambient Computing Middleware architecture is a decentralized, QoS oriented data-centric communication middleware architecture providing scheme to create modular application stacks and API to integrate sensory and QoS requirements.

In this section, we pair general features with problem areas described in Section II.

A. Communication in AMBICOM

AMBICOM uses Data-centric Publish/Subscribe scheme to handle distributed communications and relies on L2/L3 networking control function virtualization and provides application developers to use either TCP or UDP as well as other protocols in applications. The main motivations are:

- L4-L7 protocols are mostly application specific. For instance, a reliable transport might required for application A (sensory data file transfer) and an unreliable transport might required for application B (video streaming). The former application will choose TCP where the latter might prefer UDP. Although Data-centric communication among nodes is mostly UDP and Multicast, QoS policies and ACL-like policies provide security.
- L2/L3 is providing minimum set of manageable networking required for AmI devices. This provides flexibility and extensibility. Most AmI devices rely on TCP/UDP on L4-L7.
- LoS/BloS networking management require at least L2/L3 networking services.

The moderate AmI device will likely to have a CPU and a communication stack of at least L1/L2. AMBICOM might cover these AmI devices with fundamental data link services either built-in to device or by platform itself:

- Connectionless or Connection-oriented virtual data link,
- Error and Flow control,
- MAC and Multiple-access control for Ethernet, Wireless, Bluetooth, RFID,
- Data Link Switching (LoS/BloS),
- Anycast Multicast/Broadcast Routing (LoS/BloS),
- Quality of Service.

1. LoS Communication Management

Line of Sight communication starts with learning mechanism. Learning in this context is ability to obtain a unique identifier of another device after a communication package (or frame in networking terms) arrives from it. This could be MAC in case of Ethernet, Wifi, Bluetooth, and similar protocols. Each node keeps flow tables for LoS communication

2. BloS Communication Management

BloS communication management is handled through Proxy'ing of communication on behalf of initiator. This feature is built-in to each AMBICOM node and each node keeps proxy flows with additional flow tables. This proxy flow tables is a bit different than forwarding tables since no TCAM involved to implement tables.

BloS communication also implies AMBICOM application developer to choose a routing scheme dynamically. RIP [15] or OSPF [16] could be chosen for routing scheme or a custom routing scheme could be implemented.

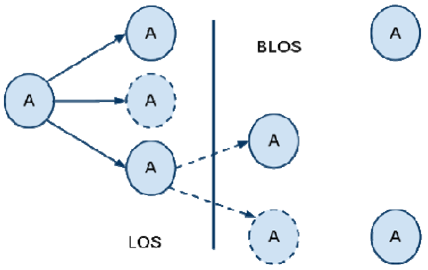

Fig. 1 LoS and BloS Communication in AMBICOM

TABLE I
ACCP COMMAND LIST

Command Name	Description
OA	Objective Asked
OG	Objective Granted
OHT	Objective Handle True
OHF	Objective Handle False
ODT	Objective Delegated True
ODF	Objective Delegated False
OD	Objective Delegate

From dynamic routing perspective, AMBICOM nodes are a bit different than usual SDN switches like Open vSwitch [17] or any other implementation like Ethane [18]. Moreover, the usual schemes for communication are Multicast or Broadcast. Therefore, different node replying to same message possibly answer and receiving node could choose one of the messages based on different metrics including custom developed ones.

Beyond L2/L3 level networking, there is also an L4-L7 protocol to provide communication in AMBICOM context.

3. AMBICOM Communication and Control Protocol

AMBICOM Communication and Control Protocol (ACCP) is connectionless application protocol providing control mechanisms for forming a Task Groups. ACCP is much like MQTT protocol [19] for that sense. On the other hand, MQTT is based solely on TCP and it requires brokers like CORBA [20]. AMBICOM deploys any L4-L7 protocol on datapath where UDP transport will be used in implementation as a proof of concept.

ACCP has commands for both LoS and BloS communication. In general, ACCP provides, communication establishment for generating topologies on-demand since AMBICOM assumes no information prior to link established and assumes changes happen in any moment.

Acting in such scheme makes TCP and central mediation functions useless. Table I shows the list of commands in protocol.

ACCP is a connectionless protocol and very lightweight in that sense. Implementing ACCP over any L2/L3/L4 networking takes minimal effort.

How the ACCP scheme works will be explained in Section IV.

B. Context Awareness in AMBICOM

AmI applications are generally defined within their own context. AMBICOM in general depends on Data-centricity and QoS-based computing to provide Context-awareness mechanisms to application developers.

Each service request (i.e. a task) given to any node in AMBICOM environment defines capability requirements. These capability requirements are abstracted from AMBICOM "fabric" with ACCP.

Capability Requirements are associated with Task in Input Data Objects (IDO's). An IDO is a context specific object that defines input data set, capability requirements and steps to complete task. As an example, an IDO for "Check e-Mails" task might include:

a) *Input Data: email credentials,*
b) *Capability Requirement: SMTP communication,*
c) *Steps: login, read headers, return (each defined dependent to the context of application).*

The granularity of each IDO is defined in application context. IDO's are tranported within AMBICOM Data-centric Publich/Subscribe framework.

The IDO results are published through Data-centric Pub/Sub framework (DCPS framework) and also context-dependent.

C. Reliability

Reliability in AMBICOM is achieved through QoS-based computing. QoS policy imposes each role of an AMBICOM node. Although details of each role will be explained in the Section IV, roles in AMBICOM are:

1. *Primary Coordinator Role,*
2. *Shadow Coordinator Role,*
3. *Task Member Role.*

Each role has its own set of QoS policies:

1. *TaskSpreadPolicy,*
2. *TaskDispatchPolicy,*
3. *TaskMonitorPolicy,*
4. *ShadowCoordinatorPolicy,*
5. *TaskRepoReplicationPolicy,*
6. *TaskAcceptancePolicy,*
7. *TaskProgressReportPolicy.*

Each application has its own set of QoS policy implementations through AMBICOM API, and these policies are imposed over roles of each node through configuration distribution.

Reliability in AMBICOM is defined within this context as:

> "If there is enough capabilities exist within an AMBICOM environment for enough duration, any given task will be accomplished."

This implies that, for any given task at a given node, it is accomplished if required capabilities exist for certain amount of time for that task to be completed within that node's LoS or BloS neighborhood. Therefore, QoS policies and Data-centric computing schemes provides reliability to AMBICOM.

IV. AMBICOM MIDDLEWARE ARCHITECTURE IN DETAIL

In this section, AMBICOM components and mechanisms are described.

A. Architectural Concepts in AMBICOM

Main concept and top most abstraction of AMBICOM is "Task Group". A Task Group is group of node, formed under task's requirements and QoS policies, where each node has a role within a Task Group.

Within a Task Group, each participant node assumed to have one Role:

1. *The node where the task is defined gets Primary Coordinator Role,*
2. *The node selected based on ShadowCoordinatorPolicy becomes Shadow Coordinator,*
3. *Based on other QoS policies, other participant(s) become(s) Team Member.*

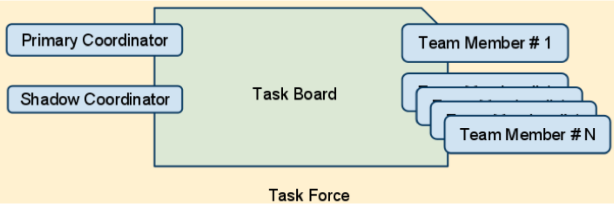

Fig. 2 Task Group in AMBICOM

Task Board shown in Fig. 2 is a logical representation; however, there is no Black Board approach as defined in Multiagent Systems. The Task Board simply abstracts the notion of task progress monitoring. In fact, Task Board is the progress table that kept on Primary Coordinator, and updated by Task Member(s) actually doing the task due to TaskProgressReportPolicy

Each node in AMBICOM can participate more than one Task Group, but a node can only be Primary or Shadow Coordinator in a group, not both at the same time.

Beside AMBICOM fabric (mechanisms to communicate and control plane functions, data-centric publish/subscribe topics for task acquisition and policy filtering), each node has pre-defined set of policies based on application context. Policies are related to Roles with Policy Filter API. Therefore, any node could dynamically configure itself based on control inputs and pre-defined filters.

B. Mechanisms of AMBICOM

In order to develop applications on top AMBICOM and program against AMBICOM API, mechanisms of different states within AMBICOM should be known.

1. Task Acquisition

Task Acquisition is done through defined Data-centric publish/subscribe framework. Each node has capability to acquire a task IDO. IDO is defined through a set of API and is context-dependent.

When an IDO is defined for a node, receiving node configures itself into Primary Coordinator (PC) Role and set of PC policies are loaded dynamically, related tables are setup and PC thread is started. Each PC thread has a defined asynchronous events and queue for nonblocking operation.

Then PC tries to establish a Task Group. Based on TaskSpreadPolicy and TaskDispatchPolicy, and using ACCP commands, PC tries to find nodes, ask them to handle the task or delegate it BloS.

Any node receiving a task request, based on TaskAcceptancePolicy, generate a reply through ACCP channel.

2. Shadow Coordinator

PC selects a Shadow Coordinator (SC) based on ShadowCoordinatorPolicy. When a node is selected to SC Role, it loads TaskRepoReplicationPolicy, creates SC threads to monitor PC based on policy (e.g. heartbeat intervals, etc.), and if PC fails, SC goes on monitoring progress and accepting results of task.

3. Task Progress

Any node receiving a task request and accept it based on TaskAcceptance policy becomes Team Member (TM). TM loads TaskProgressReportPolicy and creates TM thread. Within thread, IDO is processed and results are published to Data space of AMBICOM (i.e. Data-centric Pub/Sub Framework).

4. Control-plane

Each AMBICOM node has ACCP as a built-in feature. Moreover, control-plane (CP) for forming Task Group is seperated from data-plane (DP) where DP is implemented within Data-centric Publish/Subscribe framework. CP layer in a node provides ACCP implementation and flow tables beside TaskAcceptancePolicy.

5. Data-Plane

Data-plane (DP) within AMBICOM is provided through Data-centric Publish/Subscribe (DCPS) framework where each node participates into a global data space. Global data space is segmented with topics.

There are two kinds of topics: i) built-in topics and ii) context topics. Application developer can provide context topics through DCPS API and these context topics are published as one of the built-in topics.

Built-in topics for AMBICOM are:

a) IDOTopic: Defines IDO(s) for application context. Each node publishes its own topic names to receive IDO and response the result.

b) TaskBoardTopic: Defines Task Board topic name of the context. Task Board is related to PC and TM(s) publishes their Task Progress Reports to that topic.

Context-dependent topics are defined through these built-in topics, especially for IDO and task result.

In Section IV *C,* it is described how the example use-case from Section III is achieved within AMBICOM.

C. An Example Use-case of AMBICOM

An application context (IDO topic, policies, ACCP end points) of E-Mail Communication is developed by an application developer and deployed in an AMBICOM environment. A device receives “Check e-Mails” request (e.g. user presses a button), but it’s not capable of communicating either with Mail Server or SMTP at all. Steps might take place:

1. *PC Device publishes IDO through it’s IDO topic. Since this topic is already know to other AMBICOM devices via built-in IDOTopic, it’s already has subscribers. OA request is broadcasted.*
2. *Subscriber devices get OA command and also aware of context, process IDO (especially capability requirements). Since capability requirements are defined as SMTP and Mail Server access, devices check their capabilities and reply back with OHT or OHF based on TaskAcceptancePolicy. If any node returns OHT, Task is assigned to that node with OG reply. The node then subscribes to context dependent IDO topics and starts processing.*

3. *If all nodes in LoS returns OHF, PC issues OD command to delegation of task to BloS. All nodes receiving OD re-issues OA to their LoS and get OHT/OHF replies. If OHT reply is received, ODT reply is issued back to PC and nodes PC selects those nodes (ODT, OHT) based on TaskSpreadPolicy and TaskDispatchPolicy and a Task Group is formed. After OG command is issued back, accepted ODT and OHT nodes are become Team Member (TM).*
4. *TM process email check request (i.e. connects to mail server via SMTP and logs in with user credentials and reads all headers) and returns the result (email headers) through publishing on its result topic.*

Although this example is very simple for sake of AmI capabilities of AMBICOM, it basically explains how AMBICOM works and how an application developer interacts with AMBICOM to develop applications. Moreover, although this example of checking new email headers are incorporates discrete data payload (i.e. string data representations of each new email), stream type of data payload can also be returned via defining stream endpoints and related input parameters in context dependent result topic.

V. Related Works

There are increasing efforts toward defining a common platform for AmI applications, thus AmI Middleware. For instance aWESoME [24] is a web service based middleware utilizing W3C XML Web Services approach. AmbientDB is a P2P data management platform [25] and focused on data sharing and management aspects and based on Distributed Hash Tables as in P2P networks. HYDRA [26] also provides a middleware for AmI, and focuses on Service oriented architecture approach. LAICA [27] is another framework and it is based on Multi-agent Systems.

Many other studies focus on services that an AmI should offer [28]-[30]. Moreover, semantic (as in Semantic Web) definition of AmI is also a popular subject, for example [31], [32]. On the other hand, communication services are usually studied under WSN studies like [7], [8]. Middleware requirements are studied in [33], however, only generic middleware approaches are described.

VI. Implementation Status and Conclusion Remarks

AMBICOM is in implementation phase and early observations and comments are presented in this section.

AMBICOM is being implemented on commodity x86 and ARM based computers with IP (Ethernet of WiFi) communication capabilities.

AMBICOM core thread and role based threads are implemented on BSD-style sockets where event based processing is asynchronous. BSD-style kqueue and kevents are being used. This provides high throughput as suggested in [21]. ACCP is also being implemented in this way.

For DCPS, OMG DDS is being considered. DDS provides DCPS in high performance and, reliable transportation could be achieved with QoS policies. DDS also utilizes UDP as transportation but can provide reliability. PGM [22] and JGroups [23] can also be used, but higher reliability and performance observed with DDS. Currently, UDP based sockets and IPC are being used.

For Task Board, Symas Lightning Memory-Mapped Database (LMDB) [34] is being used. LMDB is a lightweight key-value based datastore and it is used in OpenLDAP project.

Stream type results (like streaming voice or video as a result) need to be well defined in IDO’s like resolution requirements or codec requirements. This slows Task Group forming process significantly. Therefore, streaming type of result is left out of scope of this work.

Since ACCP is lightweight protocol implemented over L3-L4 (UDP/IP), it lacks L2 functionality as of now, and definitely, L2 functionality needed for small devices or appliances. UDP Broadcast techniques are used in proof of concept implementation and each packet is limited with 1024 bytes of payload data unit (PDU) with 31 bytes of ACCP header, thus max total of 1055 bytes in line with UDP packet.

Developer productivity is not targeted for Proof-of-Concept work, but IDO/Result API and ACCP Northbound API should be well defined and documented. As of now, the system boundaries are not clear.

Current implementation has ~3200LoC of C code and working on UNIX like operating systems.

Performance of current implementation will be studied in detail; however, rough performance figure in 3 node environment consisting of 3 x86 PC’s (each with i5 CPU and 2GB RAM) connected through a standard WiFi switch is shown in Table II.

TABLE II
Performance

Test	Completion Duration
1MB data ingestion	0.57sec
10MB data ingestion	7.1sec
10MB with 1 Node Failover (down-up)	9.3 sec

References

[1] Zimmermann, Hubert. "OSI reference model--The ISO model of architecture for open systems interconnection." *Communications, IEEE Transactions on* 28.4 (1980): 425-432.

[2] Shenker, Scott, et al. "The future of networking, and the past of protocols." *Open Networking Summit* 20 (2011).

[3] Koponen, Teemu, et al. "Onix: A Distributed Control Platform for Large-scale Production Networks." *OSDI*. Vol. 10. 2010.

[4] Kim, Hyojoon, and Nick Feamster. "Improving network management with software defined networking." *Communications Magazine, IEEE* 51.2 (2013): 114-119.

[5] Nunes, Bruno, et al. "A survey of software-defined networking: Past, present, and future of programmable networks." *Communications Surveys & Tutorials, IEEE* 16.3 (2014): 1617-1634.

[6] Will, Heiko, Kaspar Schleiser, and Jochen Schiller. "A real-time kernel for wireless sensor networks employed in rescue scenarios." *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*. IEEE, 2009.

[7] Akyildiz, Ian F., et al. "Wireless sensor networks: a survey." *Computer networks* 38.4 (2002): 393-422.

[8] Dunkels, Adam, Björn Grönvall, and Thiemo Voigt. "Contiki-a lightweight and flexible operating system for tiny networked sensors." *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004.

[9] Ramos, Carlos, Juan Carlos Augusto, and Daniel Shapiro. "Ambient intelligence—The next step for artificial intelligence." *Intelligent Systems, IEEE* 23.2 (2008): 15-18.

[10] Preuveneers, Davy, et al. "Towards an extensible context ontology for ambient intelligence." *Ambient intelligence*. Springer Berlin Heidelberg, 2004. 148-159.

[11] Casado, Martín, et al. "Virtualizing the network forwarding plane." *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*. ACM, 2010.

[12] Pfaff, Ben, et al. "Extending Networking into the Virtualization Layer." *Hotnets*. 2009.

[13] Pardo-Castellote, Gerardo. *OMG Data-Distribution Service (DDS): Architectural Overview*. REAL-TIME INNOVATIONS INC SUNNYVALE CA, 2004.

[14] Birman, Kenneth P. *Reliable distributed systems: technologies, web services, and applications*. Springer Science & Business Media, 2005.

[15] Hedrick, Charles L. "Routing information protocol." (1988).

[16] Moy, John T. *OSPF: anatomy of an Internet routing protocol*. Addison-Wesley Professional, 1998.

[17] Pfaff, Ben, et al. "Extending Networking into the Virtualization Layer." *Hotnets*. 2009.

[18] Casado, Martin, et al. "Ethane: Taking control of the enterprise." *ACM SIGCOMM Computer Communication Review*. Vol. 37. No. 4. ACM, 2007.

[19] Banks, A., and R. Gupta. "MQTT Version 3.1. 1." *OASIS Standard* (2014).

[20] Ben-Natan, Ron. *Corba: a guide to common object request broker architecture*. McGraw-Hill, Inc., 1995.

[21] Lemon, Jonathan. "Kqueue-A Generic and Scalable Event Notification Facility." *USENIX Annual Technical Conference, FREENIX Track*. 2001.

[22] Gemmell, Jim, et al. "The PGM reliable multicast protocol." *Network, IEEE* 17.1 (2003): 16-22.

[23] Ban, Bela. "JGroups, a toolkit for reliable multicast communication." *URL: http://www. jgroups. org* (2002).

[24] Stavropoulos, Thanos G., et al. "aWESoME: A web service middleware for ambient intelligence." Expert Systems with Applications 40.11 (2013): 4380-4392.

[25] Fontijn, Willem, and Peter Boncz. "AmbientDB: P2P data management middleware for ambient intelligence." Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on. IEEE, 2004.

[26] Eisenhauer, Markus, Peter Rosengren, and Pablo Antolin. "Hydra: A development platform for integrating wireless devices and sensors into ambient intelligence systems." The Internet of Things. Springer New York, 2010. 367-373.

[27] Cabri, Giacomo, et al. "The LAICA project: Supporting ambient intelligence via agents and ad-hoc middleware." Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on. IEEE, 2005.

[28] Anastasopoulos, Michalis, et al. "Towards a reference middleware architecture for ambient intelligence systems." ACM conference on object-oriented programming, systems, languages, and applications. 2005.

[29] Bogdanowicz, Marc, et al. Scenarios for ambient intelligence in 2010. Office for official publications of the European Communities, 2001.

[30] Cook, Diane J., Juan C. Augusto, and Vikramaditya R. Jakkula. "Ambient intelligence: Technologies, applications, and opportunities." Pervasive and Mobile Computing 5.4 (2009): 277-298.

[31] Klein, Michael, Andreas Schmidt, and Rolf Lauer. "Ontology-centred design of an ambient middleware for assisted living: The case of soprano." Towards Ambient Intelligence: Methods for Cooperating Ensembles in Ubiquitous Environments (AIM-CU), 30th Annual German Conference on Artificial Intelligence (KI 2007), Osnabrück. 2007.

[32] Preuveneers, Davy, et al. "Towards an extensible context ontology for ambient intelligence." Ambient intelligence. Springer Berlin Heidelberg, 2004. 148-159.

[33] Georgalis, Yannis, Dimitris Grammenos, and Constantine Stephanidis. "Middleware for ambient intelligence environments: Reviewing requirements and communication technologies." Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments. Springer Berlin Heidelberg, 2009. 168-177.

[34] Chu, Howard. "Mdb: A memory-mapped database and backend for openldap." LDAPCon'11 (2011).