An Architecture for Using Metadata to Manage Ubiquitous Communications and Services: A Position Paper

John Strassner, Sung-Su Kim
IT Convergence Engineering Division
Pohang University of Science and Technology
Pohang, Korea
{johns, kiss}@postech.ac.kr

Tom Pfeifer, Sven van der Meer
Telecommunications Software & Systems Group
Waterford Institute of Technology
Carriganore, County Waterford, Ireland
{tpfeifer, vdmeer}@tssg.org

Abstract—Ubiquitous communication depends on distributed machine intelligence that relates the needs of the tasks to be performed to the available system services and resources. A critical part of this process is the understanding of contextually-sensitive data that can influence the offering and delivery of these resources and services. This paper describes a new extension of the FOCALE autonomic architecture that is designed specifically to manage ubiquitous computing and communications systems. This extension is focused on embedding intelligence into computing nodes, which enables individual nodes to communicate intelligently with the FOCALE autonomic manager. In this way, each node can contribute to the collective knowledge of the system while having its behavior orchestrated by FOCALE.

Keywords-autonomic architecture; collective intelligence; FOCALE; management; semantics; ubiquitous communications

I. INTRODUCTION

Mark Weiser described one of the salient characteristics of ubiquitous computing as follows: “In such a world, we must dwell with computers, not just interact with them. . . . Interacting with something keeps it distant and foreign. . . . Dwelling with computers means that they have their place, we have ours, and we co-exist comfortably.” [1]

This has inspired much of the work in the FOCALE [2] architecture, but particularly the development of its context-aware policy model [3]. This model enables the FOCALE architecture as a whole to create a robust model of context and contextual data, so that it can sense changes in context and determine if the current set of policies that it is using are still appropriate if the context of the task changes.

Knowledge representation systems for a given organisation or domain typically have been centralized, requiring all users to use the same tools to share the same definition of common concepts that exist in a common vocabulary. Network management is a good example of this; operational and business support systems are typically chosen because they are the “best of breed” in performing that particular type of function [4]. This means that the repositories of these applications are optimized to meet the needs of that particular application, but not other applications; this in turn means that different applications cannot easily share the same information about common managed entities. Our approach to solving this problem in FOCALE is to build a set of mappings that translate vendor-specific data into a normalized form that can be analyzed and processed; this normalized form is then converted back into vendor-specific commands for reconfiguring the device.

FOCALE itself is a comprehensive architecture, meant to be deployed and distributed on a set of servers. This begs the question, “how does each node communicate with FOCALE, and how is knowledge shared”? It is clearly unreasonable to expect a “mini-clone” of FOCALE to operate in any device; for example, the ontological processing and machine learning portions of the FOCALE architecture can require a significant amount of resources and are computationally intensive. We want any device, even a resource-constrained sensor, to be able to communicate effectively in a FOCALE-based environment.

This paper describes early work in building a companion architecture to FOCALE that is suited for embedding in small devices and machines. The remainder of this paper is organized as follows. Section II describes an overview of the architecture being built, and describes its conceptual model. Section III describes our approach for defining, managing, and using metadata to enable knowledge sharing among heterogeneous nodes in the FOCALE environment. Section IV describes a preliminary approach to incentive-based knowledge sharing. Section V briefly discusses related work, and Section VI provides conclusions and a glimpse into our future work.

II. ARCHITECTURAL OVERVIEW

In traditional computing architectures, devices belonging to different users and domains will function independently. This makes knowledge processing very inefficient. For example, if the same query is issued by two different user devices even in the same domain, they are processed independently, and the result for one device cannot get reused by the other.

Several projects have improved on this. For example, one common approach, which is shown in Figure 1, is to define a coordinator to manage the data sent from and commands sent to a given device. The functionality of the coordinator is augmented by providing additional managers, such as a context manager and/or a policy manager, to control what types of data get coordinated when, where, and by whom. The use of a shared knowledge base, such as an ontology, enables in principle knowledge to be reused. However, all ontologies are constructed with an inherent bias towards using and representing knowledge. Without explicit mapping
functions, there is in general no guarantee that the use of an ontology by itself will enable knowledge to be interoperable. This is the general subject of ontology mapping and merging; see for example [5] [6] [7] for more information about this subject.

The above topology suggests that intelligent caching could be advantageously used. For example, if needed data is cached in devices that are “close” to the source of a query, then the query can be fulfilled more quickly and efficiently. In addition, the caching service will benefit from incentive-based proximity, but rather has the broader meaning of being able to deliver the requested information quickly.

This approach is the beginning of a truly collaborative environment, which is one of the fundamental goals of ubiquitous systems. A collaborative environment is one in which each node in the system is able to play a part in enabling the system as a whole to understand its current environment as well as work together to solve a common set of tasks. Such an understanding is crucial in order to remove, or at least minimize the involvement of, the human administrator from the information processing loop. As such, we incorporate the following goals into our architecture:

- A common set of metadata should be defined, so that each node can contribute to the collective knowledge of the system
- Information processing nodes should maintain a list of context-sensitive concepts that are described by metadata to enable semantic searching to be used
- Metadata are widely distributed to coordinate knowledge acquisition and searching
- Contextual data are locally acquired and only moved to the locations where they are needed
- Incentives should be provided to encourage nodes to help find required data for other nodes in order to foster cooperation between nodes

In essence, we are using the FOCALE architecture as a physically distributed, but logically centralized, manager. The FOCALE manager governs a federated collection of knowledge sources, organized as a collection of separate semi-autonomous domains. Each node in the federation is part of its own local “smart domain”, but can also contribute to global knowledge. Each local domain agrees to abide by a set of common policies, which is the requirement for it to join the federation. However, each local domain also has its own local policies for making decisions that are used when the global policies are not applicable.

For example, a network can be visualized as a set of subnetworks. If our approach is applied, then each subnetwork is a “smart local domain”, where each node in the subnetwork works in cooperation with other nodes in that subnetwork to accomplish a task (e.g., delivering traffic from a source node to a destination node). However, that same traffic may need to travel outside of this local subnetwork in order to reach its final destination; hence, the original subnetwork can cooperate with other subnetworks as needed to accomplish this goal.
We use metadata to aid in describing, finding, and managing knowledge; this is described in the next section.

III. METADATA DESIGN AND MANAGEMENT APPROACH

Network data is predominantly derived from one or more types of data models. Therefore, we use the DEN-ng information model [9] to provide an overarching definition of the characteristics and simple behavior of managed entities of interest. An information model is independent of platform, language, and protocol, whereas a data model is dependent on at least one (and usually more) of these factors. We thus use an information model as the “ultimate source of truth” in defining the characteristics and behavior of a given entity, so that we can derive one or more data models as required to model vendor-specific functionality. This gives rise to the multi-stage mapping approach shown in Figure 3.

The first stage maps data represented in an information model to a standards-based data model; the second optional stage maps data from a standards-based data model to a vendor-specific data model. For example, a system could use different types of repositories – a directory for storing characteristics of its users, and a relational database for storing different types of analytical data that have complex relationships. Another system could use a directory for both types of data. Without an information model, there is no easy way to equate data defined in one type of repository to an equivalent form suitable for storage in a different type of repository. Similarly, there is no way to find common concepts that involve the same entity (e.g., accounting data stored in one repository with provisioning data stored in a second repository). A standards-based data model, such as one based on SQL92, encourages a high-level form of interoperability; however, most database implementations rewrite SQL queries to use a vendor-specific implementation of SQL to increase efficiency. Thus, the second set of mappings is also used in our system where appropriate.

However, a problem exists – information and data models do not have the ability to represent data using formal logic. This limits their ability to be used for inferencing and reasoning. In contrast, ontologies are defined using a form of logic, which is usually a description logic [10] or a form of first order logic [11]. Hence, FOCALE augments the knowledge provided by using information and data models with ontological knowledge. Conceptually, model data defines facts, and ontological data defines formal semantics for these facts, including definitions and meanings. Practically, we resolve differences in common data used among different vendor-specific devices by performing pattern, structural, and/or linguistic analysis of the data, as described in [4] [8].

We use this novel knowledge representation to represent metadata to explicitly describe data semantics. Metadata are, in general, widely propagated (subject to the policies of a particular domain). In DEN-ng, metadata can be attached to any managed entity, as shown in Figure 4. Through the distribution of metadata, an entity can advertise its data, as well as its capabilities or services. Hence, in this environment, two parallel types of knowledge are initially retrieved: characteristics of the entity itself, and metadata that describe that entity. This combined knowledge is then augmented by selectively adding ontological data to each, according to the context, nature of the task being performed, and of course the nature of the data itself. This enables knowledge to be incrementally built, and hence not interfere with other operations that the entity is performing.

We can now use metadata to enable semantic searching. Instead of having to rely on simple syntactic matching of keywords, we can make use of the metadata describing data we are interested in. This enables us to first, parse a query to retrieve its subject, and then search the list of subject keywords for metadata that have the closest semantic similarity (described in the next paragraph). Results can then be used to forward the query to the set of devices that are best able to respond to the query. This provides several different possibilities:

- A node contains the answer to the query, so forward the query to that specific node
- A set of nodes contains the answer to the query; either pick one at random, or pick all and compare their results, or choose the best one to forward the query to, based on the ranking of their semantic relatedness [12]
- No one node contains an answer to the query; however, different nodes contain pieces of the answer; hence, it may be possible to construct an answer or partial answer by aggregating different data from different nodes

There are several different tools available that can be used to determine the degree that metadata $M_2$ is similar to metadata $M_1$. The choice of the particular tool used depends on whether we are interested in a simple syntactic match, or in a more complex semantic match. Syntactic matching is well known, so will not be discussed further due to space constraints.
Suppose that we have two different terms, such as “pass” and “forward”, that are used in two different vendor-specific devices. Clearly, a syntactic match will fail. For cases like this, we define semantic relationships to overcome these and other complications; examples of such relationships include synonyms, hyponyms, and meronyms, as well as “custom” relationships, such as various measures of semantic relatedness [12] and domain-specific relationships (e.g., “is similar to”, which is used to determine if two sets of commands will produce similar effects). These relationships are then used to construct a multi-graph by building a set of semantic edges that associate one or more nodes in a graph representing the DEN-ng model to one or more nodes in a graph representing our ontology. A semantic edge is an edge of a graph that is formed because of the semantic and/or linguistic relatedness between the nodes that it connects, such as two nodes that have similar definitions (e.g., “gateway” and “portal”), or realizing that both a router and a firewall can forward traffic.

Semantic relatedness enables entities to be semantically related using different types of lexical relationships. Currently, we use synonymy (e.g., “bank” and “lending institution”), antonymy (e.g., “accept” and “reject”), and meronymy (e.g., court is a part of government). There has been extensive research in the field of information retrieval on semantic relatedness between multiple objects [12] [13] [14]. Most methods have high computational processing demands; worse, most assume a generalized corpus. In our work, we are using a specialized domain (networking and network management), which means that many of the words that are used in network management do not appear. For example, when WordNet [15] is queried for “Internet gateway”, no matches are found; when it is queried for “gateway”, one match is returned: “an entrance that can be closed by a gate”. This definition is not relevant for network management. Therefore, we augment WordNet with our own customized vocabulary and use WordNet’s API to establish synonyms and other lexical relationships between WordNet’s words and our own customized set of words. For example, if we define “portal” as a synonym of “gateway”, WordNet offers three definitions for portal, with the second (“a site that the owner positions as an entrance to other sites on the internet”) being the most relevant. This enables us to use a standard tool and well-known approach instead of having to build our own linguistic tools in order to determine the semantic relatedness between two entities.

Given the above multi-graph, we initially used Tversky’s semantic relatedness measure [16]. The semantic relatedness of two concepts A and B is defined as:

\[
sim(A, B) = \frac{\alpha \cdot \text{sim}_\text{syn}(A, B) + \beta \cdot \text{sim}_\text{hyp}(A, B) + \gamma \cdot \text{sim}_\text{mer}(A, B)}{1 + \beta \cdot \text{sim}_\text{hyp}(A, B)}
\]

where “\( \cap \)” means “intersection”, “\( \backslash \)” means “set difference”, “\( \| \)” means “cardinality, and “\( \text{sim}_\text{syn}(A, B) \)” is a weighting factor; the notation “\( \text{sim}(A, B) \)” is used because semantic relatedness is not necessarily symmetric (the simplest case being that the similarity of a subclass to its superclass is not the same as the similarity of the superclass to a subclass, since the subclass inherits all of the properties of the superclass and can add its own properties).

This formula defines the semantic relatedness of A and B in terms of the semantics that are common to them (\( \text{sim}_\text{syn}(A, B) \)), the semantics that are particular to A (\( \text{sim}_\text{hyp}(A, B) \)), and the semantics that are particular to B (\( \text{sim}_\text{mer}(A, B) \)). We chose this formula because it takes into account what is common while also considering what is different.

Initial results were not as promising as we hoped; we hypothesized that this is because of the object-oriented nature of DEN-ng, along with it being based on formal classification theory [17]. This latter is important, because it connotes that the deeper an object occurs in the hierarchy, the stronger the asserted fact is. This is not taken into account in the above formula. Therefore, we modified (1) as follows:

\[
sim(A, B) = \beta_\text{syn} \cdot \text{sim}_\text{syn}(A, B) + \beta_\text{hyp} \cdot \text{sim}_\text{hyp}(A, B) + \beta_\text{mer} \cdot \text{sim}_\text{mer}(A, B)
\]

where \( \text{sim}_\text{syn}, \text{sim}_\text{hyp}, \) and \( \text{sim}_\text{mer} \) denote the similarity between the synonyms, hyponyms, and meronyms of A and B as defined in (1), and the parameters \( \beta_\text{syn}, \beta_\text{hyp}, \) and \( \beta_\text{mer} \) denote a weighted value for these three similarity measures, respectively; each weight is greater than or equal to zero, and the sum of the three weights must equal 1.

Initial results are much more promising using this modified approach. In particular, note the use of each of these relationships (which are unfortunately beyond the scope of this paper due to size constraints):

- Synonyms enable new possibilities to be tested
- Hypernyms find superclasses of concepts, and hence strengthen the semantic similarity
- Meronyms find whole-part relationships, which can be used to change the search to find new concepts as well as to reinforce semantic relatedness

We now need to apply this approach to address the needs of a ubiquitous communications environment. In general, there are two types of devices in such an environment, those that provide public resources and services, and those that are used for private purposes. This leads to the following problem. Traditionally, when two devices issue queries for the same data, two searches are performed independently, because the search result for one user device cannot be shared and reused by the other device. This is not just because of different source data. For example, each device could issue the same query in a different query language.

While metadata describe data semantics, queries state the desired properties of the requested information. However, we note that the query structure and the meaning of each element in this structure can themselves be used to provide insight into the data, via its metadata. Hence, we apply our knowledge representation to both data as well as queries, and track each as a tuple, in order to better understand the source of the data and how those data are being used. This leads to the conceptual architecture shown in Figure 5. Each node in our architecture implements the three sets of APIs shown in Figure 5; this enables communications to be defined in a
common manner. The Model-Based Transformation function, the Semantic Similarity function, and the three repositories shown, can either be implemented directly in the node, or external to the node, depending on the resource constraints of the node itself. The Model-Based Transformation function converts vendor-specific data into a normalized form, and the Similarity Computation function computes syntactic and semantic similarities of terms.

![Figure 5. Simplified Node Conceptual Architecture](image)

Currently, our knowledge representation uses graphs and custom data structures to combine knowledge extracted from models and ontologies. This enables existing tools to be used to manage and represent knowledge. Specifically, we use Rational Rose v7.0.0.0 for information modeling, and Protégé 3.4 beta for ontology design. By transforming existing knowledge from model elements and ontologies into a common format (graphs), it becomes possible to use established linguistic relationships to associate knowledge from model elements with knowledge from ontologies (and vice-versa). This approach lends itself to reusing existing languages, as well as developing new languages. Our algorithms execute in times ranging from less than a minute to less than ten minutes for small instantiations (tens to hundreds of network objects), depending on the amount of information contained in the classes (e.g., number of attributes of the class and relationships that the class has with other classes).

IV. INCENTIVE-BASED KNOWLEDGE SHARING

Ubiquitous communications requires infrastructural support. While not always possible, if an entity can contribute to the information access and processing needs of other entities, it stands to gain from like services performed for it. This enables entities to organize themselves in a way best to enable them to share knowledge and processing with other entities. This is a complex proposition that has many different possibilities, ranging from requiring entities that participate in such an environment to provide some minimal set of services and resources to providing incentives to entities that provide such services and resources to encourage them to continue to do so. Hence, we are designing several different incentive schemes for devices to participate in the activities of others as part of our future work.

For example, if a node participates in a particular process, it is eligible to receive data, metadata, and services from other nodes. Since metadata are associated with data describing an entity, they are classified as a tuple \((D_i, M_i, N_i)\) describing the data \(D_i\), its corresponding metadata \(M_i\), and the node \(N_i\) that contains the data. We also classify queries against data and metadata, along with some simple success metrics indicating whether the original query was directly satisfied or not using just the data or just the metadata. This is currently a set of simple Boolean attributes, and hence can be encoded as a bit vector for speed and processing efficiency. This provides feedback to the system concerning the utility of the metadata.

Given a query, we examine the terms in the query and compare them with terms in our metadata. If there is an exact match, and if the node that owns those metadata is available, then we use this match to forward the query to that node for processing. Otherwise, we compute the syntactic and/or semantic similarity of the terms in the query to determine the set of nodes that have the closest matching metadata to the query terms. Since the node that we want to forward the query to may not be directly connected to the query source, we record the path taken that connects the originating node to the node(s) holding the answers to the query, noting all intermediate nodes that help in the query. Nodes that help forward queries will obtain more knowledge about data and their locations in the network, thus enhancing their ability to serve future queries. When forwarding queries, nodes record the nodes that initiated the queries. When passing the query results to the initiating nodes, the nodes record the nodes providing the results. This way, nodes obtain knowledge about the data in the network, and knowledge of the actual data sources present. This knowledge can be useful in serving subsequent queries. The nodes that participate more in forwarding queries will have more knowledge, enabling them to build more sophisticated knowledge services.

Once the query executes, the results are sent back to the query source, and the corresponding metadata are sent to all intermediate devices that helped in performing the query. Similar queries made in the future will benefit from this initial transaction, since the system records data and queries, along with the success or failure of those queries, as tuples.

We can also use a modified version of this procedure to decide which node out of a set of nodes is best suited to answering a given query. For each metadata, we compute the semantic similarity between the query terms and the available metadata. The query is then forwarded to the node that has the most similar metadata.

V. RELATED WORK

Semantic searching is not a new idea. [18] defines a set of similarity measures for comparing ontology-based metadata. In particular, they discuss taxonomy, relation, and attribute similarity. This provides similarity according to three dimensions, based on their corresponding concepts and position within a taxonomy, their relations to other objects, and their attributes and attribute values. Our work is different, in that it uses a different measure of semantic similarity, called semantic relatedness. For example, in [18], similarity
based on relations assumes that if two instances have the same relation to a third instance, they are more likely similar to each other than if the two instance have relations to completely different instances. However, this does not measure how the two instances are semantically related to each other, whereas our method does. In [18], the arithmetic mean of these three measures is computed, which is different than the computation of semantic relatedness.

MoGATU [19] treats all devices as equal semi-autonomous peers that are guided in their interactions by profiles and context. Caching and replication are deployed at the devices, according to profile and context information. MoGATU is different from our work in the following respects. First, MoGATU is a peer-to-peer based system, whereas ours can use any set of communication mechanisms. Second, MoGATU implements a linear chained architecture, meaning that each node forms a direct chain in answering a query. Our approach is not constrained to use this mechanism.

VI. CONCLUSIONS AND FUTURE WORK

This paper has discussed an architecture that is based on using semantics and metadata to provide a distributed collective intelligence describing resources and services contained in the environment. It is a collaborative environment, predicated on the vast majority of its nodes being willing to provide resources and services for the needs of other nodes in exchange for like services rendered for it. Our architecture uses metadata to both locate information as well as to refine queries. In both cases, the objective is to find the set of nodes that contain the data being searched for; a consequence of our approach is that each participating node learns new information that enables it to provide new knowledge-based services.

Future work will proceed in several areas. First, we will continue to investigate different combinations of algorithms for constructing semantic similarity. Second, we will compare different caching strategies that can be used to make delivering previously found results more efficient. Third, we will build a large tested to measure how different algorithms perform against each other in retrieving semantic information. Finally, we will design and compare several different incentive schemes for devices to participate in the activities of others as part of our future work.

ACKNOWLEDGMENT

This work is sponsored in part by the WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0), the Irish High Education Authority (HEA) into the PRTLI Cycle 4 research program in the framework of the project “Serving Society: Management of Future Communications Networks and Services” and the Science Foundation Ireland under grant number and 08/SRC/11403 (project “FAME”).

REFERENCES


