A CONTEXT INTERPRETATION FRAMEWORK FOR COGNITIVE NETWORK DEVICES

Hrishikesh Sharma(Tata Consultancy Services Ltd., Bangalore, Karnataka, India, hrishikesh.sharma@tcs.com);
Balamuralidhar P(Tata Consultancy Services Ltd., Bangalore, Karnataka, India, balamurali.p@tcs.com)

ABSTRACT

Application of Cognitive techniques is an upcoming paradigm in telecommunications, in which either a network or a radio can change its configurations and techniques, to communicate and manage efficiently in changing environment. Application of knowledge-based approach to embed computational intelligence in these network devices to achieve this object is of exploration by many researchers.

In this paper we propose an architecture for reasoning and interpretation over information gathered in form of context data. A prototype was developed that uses certain knowledge management tools as its components. We present an overview of the prototype and further discuss some of the results. Certain deployment strategies and open issues are also discussed in this paper.

1. INTRODUCTION

In telecommunications, especially wireless communications, there is a thrust towards usage of cognition in next-generation networks and devices. The requirement here is that based on user needs and the situation of network, provision of radio resources and wireless services most appropriate to those needs is sought. Such requirement can, in general be satisfied by making the network devices automated as well as intelligent. Current research trends for next generation technology bring out strong relevance of embedded intelligence in various network functions.

Research towards provision of better end-to-end performance has led to conception of cognitive networks. A cognitive network has a cognitive process that can perceive current network conditions, and then plan, decide and act on those conditions. The network can learn from these adaptations and use them to make future decisions[1].

The context data represents the dynamic state of various components within a cognitive network. The data is raw data, gathered with help of various sensors and monitors embedded or placed at various places within the network. A cognitive device can intelligently act only once it has correctly gauged the current 'situation'. A 'situation' is in practice a compound state, based on various other context data. Inferring such meaningful situation out of raw data is the responsibility of a context interpretation function.

There are several middleware frameworks that are available to design systems that work with context data[2]. For example, SOCAM and CASS both provide holistic architecture to implement a context management subsystem. Especially within wireless networking domain, specialized frameworks for context management such as ContextWare[3] have started emerging. However, most of these architectures incorporate the interpretation function in partial way. That is, they do not use all possible sub-functions involved in a joint, coordinated and synergistic fashion, but some. To have a framework that is stable, practical and is a superclass of all these partial architectures, will allow even more kind of intelligence requirements within a cognitive device/network to be implemented. Hence the focus of our work has been to strengthen the reasoning framework i.e. the interpretation function.

1.1. Need within Reconfigurable Networks

From an economic point of view[4], the path ahead lies in convergence and interworking of existing, emerging and future radio systems and mobile networks. Hence next generation networks are going to be highly heterogeneous. Reconfigurability will then be the *enabling technique* that aims to differentiate user perception in volatile radio conditions while optimizing the use of network resources. The goal of end-to-end reconfigurability project, E2R, in whose purview this work was done, is to address design evolutions for such networking in the context of telecommunication infrastructures.

Reconfiguration of various network components can only be done if the corresponding 'situation' requiring the reconfiguration action has been sensed a-priori. This necessitates the requirement of a context interpretation function. In the kind of network scenario discussed above, compounding raw sensor data coming from various sources and networks, of various types, at various rates and mechanisms is never an easy task. Hence a structured approach to context interpretation is always favorable.

Further, one of the aims is to make our solution amenable to open-adaptivity. Open-adaptive implies that new adaptations can be discovered and it should be possible to upgrade the system at runtime for such further adaptations. It is an important requirement because while communication related adaptation space such as RAT switch etc. might be limited, the adaptation space related to the mobile device user can grow unlimited. Handling user needs is becoming increasingly important with the shift of service design from technology-oriented to user-centric.

Not much literature can be found which deals with a structured way for context interpretation. Also, architectures for conventional radio elements are not sufficient to handle such advanced capabilities. To have open-adaptive system, a modular and extensible architecture is required. This entails a formal fine-grained structure for context interpretation. Also, such structure will be flexible towards future advances in context-awareness and reasoning. Finally, such architecture can also be used to cater to providing context information at multiple aggregation levels.

The rest of the paper is organized as follows. Section 2 places context interpretation within the bigger structure of autonomic computing model for reconfigurable systems. Section 3 lays the foundation of the architecture. Detailed architecture is presented in section 4. Implementation details and results are discussed in section 5 and 6 respectively. Section 7 presents some deployment options, while section 8 concludes with discussing some open issues.

2. AUTONOMIC COMPUTING, RECONFIGURABILITY AND INTERPRETATION

The administration and management of complex information and communication systems in the future networking scenario comprises a significant part of overall operational expenditure that has raised the need for self-management. Autonomic computing emerges as a new paradigm[4] for managing such complex tasks at various levels without human intervention. The corresponding adaptivity also leads to increase in usability and effectiveness by taking system's environmental context and self-information into account.

Following [4], there is a third level of evolution in design sophistication of networking systems, in which the set of multiple contexts of deployment is not known a-priori. One may note that engineered systems are generally designed with a known and specific context of deployment. In such cases, hence, there is a need for mechanism in the architecture to infer contexts and learn the pattern of context changes in addition to mechanisms of self-reconfiguration, that is, to realize autonomic capabilities.

The reconfigurability required to enable such autonomic capabilities *is guided by* the Cognitive networks paradigm defined earlier. It is an extension of cognitive radio paradigm, which in turn is an enabler of SDR, with introduction of learning capabilities for user, network context etc *while focusing on* end-to-end goals. Hence one can refer to the six-stage OOLPDA model[1] proposed by Mitola. This model is a direct extension the four-stage MAPE model referred by E2R[4].

2.1. Interpretation function within OOLPDA

In order to be context aware, the network elements must interact with the outside world. This is accomplished via the cognition cycle, OOLPDA. The cycle is a model to a process that can perceive current network conditions, and then plan, decide and act on those conditions. Also, learning can happen from sensing effects of various adaptation acts, and be used to make future decisions.

The cognition cycle starts with observing the environment as well as self-conditions. This is done by collecting raw data from various sensors and monitors. The data is according to need to react to or act upon, according to systems' management tasks and end-to-end goals. The model is mum on how the data is processed next, before the cycle moves into orientation phase. Here one can go back to MAPE model to understand the steps. The collected raw data is elaborated and organized through events. A situation is physically an event, and it indicates a change of state of a resource. To gauge occurrence of a potentially interesting(for example, dangerous, valuable, or important) 'situation' needing reconfiguration in the managed system using these events, correlation with other events is required. The main goal of correlation is to enrich the meaning of the events by condensing the received events into a single event. Such task hence falls in the purview of 'analysis' stage.

Such condensed, observed (event) data is the input to next stage, where the cycle 'orients' itself by determining the priority associated with the input. As can be seen now, the context interpretation function falls under the purview of 'analysis' stage of MAPE, or 'observe' stage of OOLPDA.

3. CONTEXT MODELING AND INTERPRETATION

A requirement of autonomic communications is that the state description of 'communication world', i.e. the telecom infrastructure as well as the users, be explicitly represented in machine-processable form. This description is essentially the context information, which by definition, surrounds, or gives meaning to, an entity. The complexity of reconfiguration planning and action mentioned before can be resolved only by exploiting the interdependencies among components of this 'communication world', systematically.

A context model is needed to define and store context data in the required machine-processable form. A survey by Strang et al[5] classified the most relevant context modeling approaches. The conclusions of their evaluation show that ontologism is the most expressive model and fulfills most of their six requirements for context-aware systems.

3.1. Ontology-based Modeling

In context of this paper, ontology provides a set of definitions and relations between concepts relevant for networking, device, services, users etc. The introduction of ontology-based explicit models of resources within a network enables the network to deal with policy-based goals on a higher abstraction level. Ontologies are key requirements because [6] 1) a common ontology enables knowledge sharing in an open and dynamic distributed systems such as the vast networking scenario mentioned before, 2) ontologies enable machine-processing, and 3) ontologies allow devices and agents not expressly designed to work together to interoperate. This requirement is especially meaningful in a heterogeneous network.

3.1.1. Ontology and Logic

For machine-processability, it is required to access *explicitly represented* knowledge and to process it. Ontologies allow machine-processing of context data by enabling formal reasoning over them. It is reasoning that lends the system capability of intelligence. Reasoning as a function has foundations in formal logic. A logic allows the axiomatization of the domain information, and the drawing of conclusions from that information. Hence ontologies are represented using various languages, whose semantics is based on various logics.

Logics are generally organized in terms of their expressivity[7] which represents the ability to say things more precisely. Unfortunately, along with expressivity, comes the difficulty of reasoning with the logic. Some logic can be so expressive that reasoning can go into infinite loop. There are also computational complexity issues with these logics. For that matter, in popular ontology languages, finite expressive logic such as description logic is most used.

3.1.2. Ontology Languages

There are wide varieties of languages for explicit specification of concepts and relations for context information. The graphical notations include UML and RDF while the (description) logic-based notations include DAML+OIL, OWL etc. The actual instantiated-from-concepts context data is specified using notations such as RDF. Typical features of such instances can be found in [2]. OWL Language is an extension of RDF; hence it is possible to associate an instance pool with an OWL ontology. All information is organized in so-called OI-models(ontology-instance models), containing ontology entities (concepts and properties) and their instances. This allows grouping of concepts with their instances into self-contained units. Such a grouping is also referred to as *Knowledge Base*.

3.2. Ontology-based Interpretation

While the context information can be stored and represented using ontologies, the "intelligence" is implemented by context interpretation engine, which reasons over the context information. Classically, ontology based methodologies have been well explored for performing interpretation in systems predominantly at application level. But the approach could be extended further to cover all other entities in a cognitive network by integrating information about radio resources, protocols and user actions. Suitable mechanisms for context interpretation are required here to process the raw context data to bring out meaningful higher level abstractions. These mechanisms are generally provided in form of various ontology *rules*.

Ontology rules provide a way to define behaviour in relation to a system model. There are generally two types of rules[8]. *Correlation rules* correlate various context event data mainly to enrich the meaning of events, and to reduce the number of events. *Action rules* enable taking action over these enriched context events via various system policies.

One can see that the context interpretation function is limited to using correlation rules in the above context. Popular schemes for specifying correlation rules include RuleML, and recently, SWRL. SWRL is an extension of OWL DL. Hence specifying rules in conjunction with ontological terms and data becomes an easier task. SWRL extends the expressivity of OWL at the expense of the decidability of query answering operations. In order to cope with this problem, several decidable subsets of SWRL have been identified and investigated, including DL-safe rules.

4. SYSTEM ARCHITECTURE

The detailed architecture for implementing context interpretation function into cognitive radios and networks is described in the following sections. The blocks in grey relate to context provisioning issue, and are discussed briefly only to provide completeness to the interpretation architecture.

4.1. Syntax Conversion

Context data is generated mainly by sensors(exception: profile data), and they are not in general owned by the network owner/service provider. Since they are pervasive and hence vast in numbers, there is a high chance that their outputs are in different formats. Thus, for transportation purposes, a wrapper mechanism is required for context data to arrive at the interpretation point in single format. XDI is a promising technology in that aspect[9].

Because we used ontologies to represent context information, we chose RDF as the suitable candidate for storing context data. Hence it makes sense to convert the data into RDF format at early stage itself.

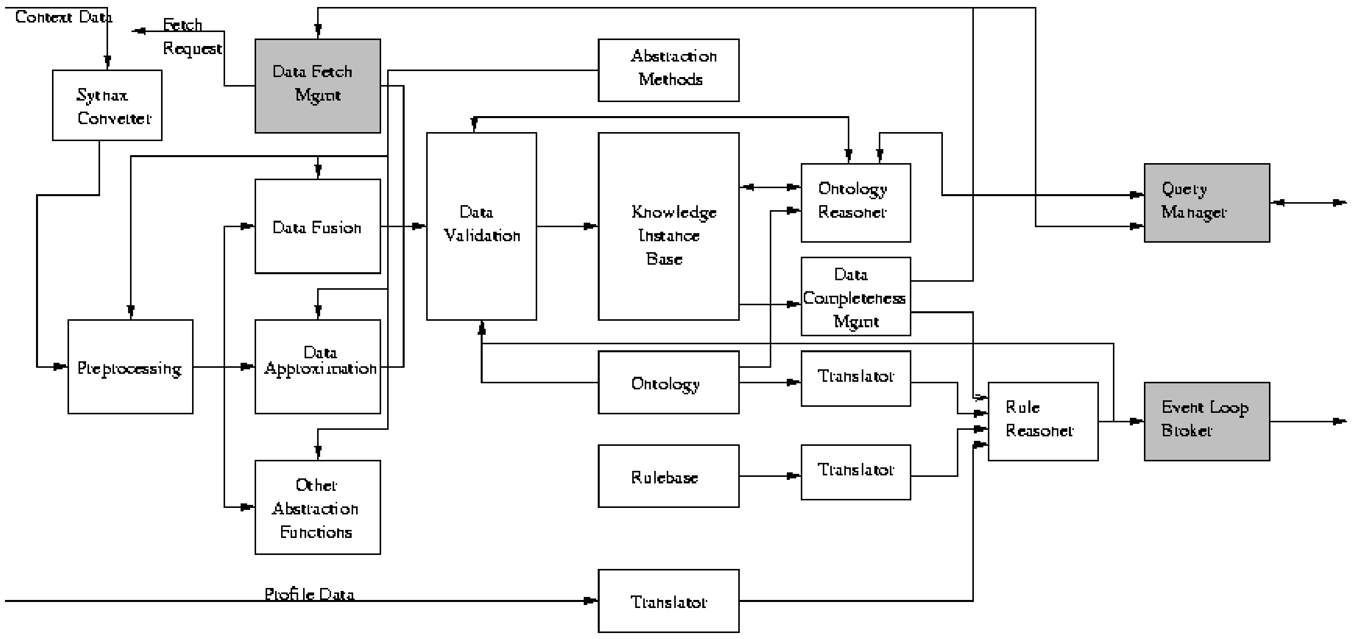

Figure 1: Architecture for Context Interpretation

4.2 Context Abstraction

Analyzing context data is difficult if the data is not normalized into a common, complete, and consistent model. *Preparing* context data for analysis entails not only reformatting the data for better processing and readability, but also moulding it into its most cohesive pieces. Further, it includes filtering out unwanted information to reduce analytical errors or misrepresentations. The capabilities employed for this purpose are discussed as follows.

4.2.1 Initial Preprocessing

The main function here is data reformatting. The requested data may be retrieved from several context sources. Each of these context sources may use different interfaces, protocols and data formats to exchange information. Hence this information is aggregated into one common data model, before further usage. For example, in a usecase, we implemented a method to change data timestamp format.
To aggregate into one common format, mappings need to be defined between the data formats used by each of the context sources and the common data format. Content MathML is one of the languages fitting this requirement.

4.2.2 Data Approximation

Sometimes incompleteness of some data can be worked out by approximating, rather than re-asking for its more precise sensing. This can include, for example, resampling, interpolation/extrapolation, statistical parameter estimation etc. Generally, a user-provided library of methods is used, For example, remaining battery life is generally an approximation.

4.2.3 Data Fusion

Data fusion is the combining of context data from multiple disparate sources such that the resulting information is in some sense *better* than would be possible when these sources were used individually. The term *better* is obviously desirable, and can mean more accurate, more reliable, more meaningful, more complete, or refer to the result of an emerging view of same data object, e.g. stereoscopic vision.

At times, user's behavior pattern also needs to be recognized for prediction and usage. Feature extraction from a window of data, for this purpose, logically becomes part of data fusion.

The data sources may be similar, such as antennas, or dissimilar, such as electro-optic or acoustic sensor. A key issue is to deal with conflicting data, producing interim results that can be revised as more data becomes available.

4.3 Data Validation

Context data, before being stored for usage, needs to be filtered. Filtering generally involves eliminating incorrect, invalid or unknown data. For example, reasoning could determine that two context sensors have provided really different location information, which is an inconsistency. At times, given ontological restrictions, there may outliers of the raw context data, which need to be removed.

Further, this module must be used to check the consistency/integrity of the knowledge base, at the juncture when another piece of information is about to be added. This requires entailment reasoning, and hence an ontology reasoner's support is required for data validation.

4.4 Knowledge Database

A knowledge base is a special kind of database for knowledge management. It provides the means for the mechanized, computer-readable collection, organization, and retrieval of knowledge in form of (context) data. Due to machine-readability, automated deductive reasoning can be applied to them. The bases contain a set of data, often in the form of facts that describe the knowledge in a logically consistent manner. An OWL ontology directly maps to a DL Knowledge Base K = ⟨T, A⟩, where T is set of TBox and A is set of ABox axioms[7]. Our work required this kind of knowledge base to be used, based on description logic.

Physically, the knowledge base stores the context data in RDF syntax in the form of C(x) or R(x,y). A snapshot of OI model in OWL, from knowledge base is in the figure below. For example, it tells that there is an employee instance, 'Ryan', who is linked to a BatOpTerminal(battery-operated terminal) instance 'Laptop' via hasTerminal property.

```
<Employee rdf:ID="Ryan">
  <hasTerminal>
   <BatOpTerminal rdf:ID="Laptop">
    <hasProfile>
     <TermProfile rdf:ID="Profile1"/>
    </hasProfile>
  </hasTerminal>
</Employee>
```

Figure 2: Snapshot of Knowledge Base

4.4.1 Other Components

Classically, a knowledge base only consists of Abox and Tbox rules. However, storage is also needed for various rules as well as abstraction methods/formulas. In certain implementations, storage for these can be clubbed with ontological data storage.

4.5 Context Data Reasoning

As we saw earlier, data validation requires the service of an ontology Reasoner. Checking integrity of data in a knowledge base, detection of redundancies, and property refinement are some of the used functions of a DL reasoner.

4.6 Rule-based Reasoning

In OWL, it is not straightforward to join two property instances in a <subject predicate object> form. For example, it is not possible to deduce, "I am in a room with no coverage". Similarly, it is not possible to make temporal statements in OWL, something which is a must for dynamic domain such as wireless networks. A lot many times, reasoning over context history is needed to detect an event.

Also, in OWL it is difficult to reason on a subset of a given ontology, since the OWL reasoner takes all available knowledge as input and there is no standard way of only providing a part of this knowledge based on certain criteria. To improve performance, decreasing the amount of information that is used to perform the reasoning is needed.

One of the better options to handle these problems is to use more expressive languages than OWL. SWRL, introduced before, is an example, which allows to reason on composite relations using rules. Data reasoning rules run on basis of *symbolic fusion* in order to generate higher-order, *implicit* context. The correlation rules could either be *stateless rules*, considering events in isolation, or be *state-based rules*, analyzing events gathered over time. The new generated data needs to be stored back into knowledge base to support querying.

For example, in a particular case, the only available information on the location of a user could be that he is in a specific meeting. However, this might not be the required information since the requestor expected geographical location information with a specific quality of context. Based on other information available – like the building the meeting is taken place, other attendees that might be present, etc. – this information could still be derived. An example rule from one of our usecases is as follows. It says that if an employee has a terminal that has a battery which will last till time x, and the employee is found to be traveling via a travel booking from his daily work plan, and if time x is found to be less than the end time of employee's travel, then the battery needs charging(new information).

```
Employee(?a) ∧ hasTerminal(?a, ?b) ∧ hasBattery(?b, ?c) ∧
willLastTill(?c, ?d) ∧ hasWorkPlan(?a, ?e) ∧
hasTravelBooking(?e, true) ∧ hasBookingDetails(?a, ?f) ∧
travelReqd(?f, true) ∧ bookingReqd(?f, true) ∧
hasApproxEnd(?f, ?g) ∧ swrlb:greaterThan(?g, ?d)
 → needsCharging(?c, true)
```

Figure 3: Example SWRL Rule

The rule-based reasoning may require to be done iteratively till some point, because the new information may further correlate with other present information.

4.6.1 Data Completeness Management

For state-based correlation, one needs to wait for data to arrive before a set of rules can be applied. This can be achieved by implementing various FSMs. One cannot assume that all sensors always active. Hence at times, when a decision needs to be taken without waiting long, this function can trigger a data fetch as well.

4.6.2 Data Fetch Management

This function manages local loops of various fetch requests placed by various modules.

4.6.3 Profile Data and Rule Reasoning

User preferences, generally captured using various profiles, can influence interpretation. For example, a reduction in user memory partition can lead to blockage of a software download. Thus, sensed data, and shared profile data are both considered during reasoning.

It is expected that profile data is in a different format, and hence it needs to be translated before used.

4.7 Event Triggering

To implement a push, the output of context interpretation function is fed into various other stages of the cognitive cycle. The output can be transferred to corresponding modules using events representing an implicit context.

A way to fetch such events includes the decision module etc. registering themselves with the context broker module, to be notified of particular events. In turn, the broker needs to monitor any context information that might be relevant for each of the modules that have registered. As soon as a particular type of context information changes, this is noticed by the context mediator, which notifies the applications that have registered for this event.

4.8 Information Querying

To implement a pull, i.e. the other way to know that a situation has occurred, is by querying the knowledge base. In this process, the modules retrieve current data from the base, and at times, further use the rule reasoner to fuse and to figure out whether a situation has arisen.

Multiple queries may happen concurrently in practice. Hence a query management module is required. In case queried information is unavailable, this module also needs to commission data sources. Reasoning capabilities might be useful to a certain extent to classify context sources and sinks that receive and provide particular context information, and thus to help in matching.

5. SYSTEM IMPLEMENTATION

An initial proof-of-concept system has been implemented and tested using use-cases that illustrated feasibility of such an architecture. The system was implemented as a tool-chain using tools from semantic web technology area. OWL was chosen as the ontology language, RDF as the context data language, while SWRL was chosen as the rule language. Being extension of OWL, it is easy to store SWRL rules with OWL, RDF information. The popular query language to RDF database is SPARQL, and we chose this language for our use.

Protégé-OWL plugin was used as the basic framework, because it *solved most of the integration problems*.

1. Protégé is based on Java, as well as most other tools.
2. Protégé supports exactly the same fragment required for our work – OWL-DL.
3. Protégé has a built-in knowledge base container.
4. Protégé allows editing and storage of SWRL rules in the knowledge base.
5. Protégé supports rule engine integration, especially with JESS.
6. Protégé also attaches to various OWL reasoners easily, such as Pellet or RacerPro.

We chose Pellet as ontology reasoner for ease of integration as well as license issues. Similarly, JESS was chosen as the rule reasoner. To support SPARQL query, we used JENA framework. The entire integration work was carried out in Java language.

Work is in progress to design and integrate generic modules for various context abstraction functions such as data fusion. It is envisaged that a lot of formula-based processing will be involved. Possibilities of integrating a lightweight Content MathML interpreter are being explored. By generally extending definition of a knowledge base, and by using annotations, one can incorporate logical responsibility for being able to store MathML formulas in the database. Alternatively, especially for approximation, use of built-ins of SWRL is also being considered.

6. OBSERVATIONS

The prototype was tested by us for some usecases for functional as well as performance studies. The first performance bottleneck was found in full validation of knowledge base, when some new information was added. We changed a runtime parameter of Pellet to make it do quick, *incremental* reasoning. We experimented with iterations over rule-based reasoning, but could not gain clarity over when to stop.

As the size and complexity of ontology model using OWL DL increased, we found it very time-consuming to fix semantic defects, even after enough experience with the language. We found the debugging support in Protégé is not strong. Especially in scenarios, where the ontology may get updated after learning, it makes sense to experiment with ontology repair systems.

For validation, we had tried using DIG socket-based tunneling mechanism. However, as many people have faced, we too faced problems in working with some OWL constructs via DIG version 1.0, which is current but incomplete. Hence we switched over to direct, API-based integration. The time-performance for our medium-sized ontologies drastically increased. The Protégé-JESS bridge exports only those facts that are relevant to the set of rules being executed, and hence time-performance was satisfactory. Overall, the time taken at various steps for a database of 23 concepts, 39 properties and 149 concept/property instances was found to be of order of 100 ms on a Pentium-IV processor-based desktop.

7. DEPLOYMENT STRATEGIES

For cognitive networks, the context server approach to architecture deployment[2] seems to us the most suitable. This approach permits multiple client access to remote data sources. The server deals with both the context provisioning as well as the interpretation aspects. For provisioning, an access managing and a gathering component are envisioned. Since end cognitive devices such as mobile handsets have resource limitations in computation power, disk space etc., this approach relieves such clients of resource intensive operations, which are performed on the server. Doing it centrally also increases re-usability of cached results by multiple clients, and hence increases net network performance. On the flip side, appropriate protocols and QoS issues for client-server architecture need to be worked out. Since historical context data is also required by various clients, and that the maintenance of context history is mainly a memory concern, the server approach fits well in this requirement as well. Most of the context management frameworks surveyed exhibit strict division of the context data acquisition and use. Thus context sources become reusable and are able to serve a multitude of context clients.

To avoid different applications trying to contact the server for recent information, *local caching* of results is suggested. This caching may be distributed e.g. over multiple base stations in order to minimize retrieval time. Further, an extension in form of *distributed, layered approach* is also possible. It is expected that not all interpretation sub-functions will be required in various deployment scenarios. Certain *required* context abstraction functions, which do not require major resources, can be part of a lower layer doing part of interpretation locally, before handing over control flow to upper layer in the server. Certain interpreted information such as spectrum availability may be broadcast using E2R concepts such as cognitive pilot channel and resource awareness channels[10].

8. OPEN ISSUES

One of the open issues is the footprint requirement of such an architecture. In E2R, it is being debated whether the interpretation function will be pushed into a cognitive device. In such a case, number of tools need to be cut down. Evolution of Pellet holds much promise, because it specifies support for better querying and rule execution integration in future.

Another problem in the presented approach is the variety of used context encodings found in practice. Every system and framework uses its own format to describe context and its own communications mechanisms. Standardization of formats such as XDI/XRI and protocols holds key to resolving this issue.

The interpretation function for cognitive networks, that's driven by OOLPDA model, can't be complete without a learning element. Machine learning at top of Bayesian Networks' based inference can be explored in this aspect.

9. CONCLUSION

We have described and validated an architecture for context interpretation that is based on ontologies. The usage of rules, facts etc has been demonstrated within the prototype for this architecture. Further evolution of architecture in terms of design and performance is expected, as more and more open issues get addressed.

10. REFERENCES

[1] Thomas, R.W. DaSilva, L.A. MacKenzie, A.B., “Cognitive Networks”, First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, pp. 363-360, November 2005.

[2] M. Baldauf et al, “A survey on Context-aware Systems”, International Journal on Ad-hoc and Ubiquitous Computing, Vol.2, No. 4, 2007(forthcoming).

[3] Alex Galis et al, “Contextualisation of Management Overlays in Ambient Networks”, International Multi-Conference on Computing in the Global Information Technology, 2006

[4] G. Dimitrakopoulos et al, “Adaptive Resource Management Platform for Reconfigurable Networks”, ACM Journal of Mobile Networks and Applications, Vol. 11, No. 6, pp 799-811, 2006.

[5] T. Strang and C. Linnhoff-Popien, “A Context Modeling Survey”, Sixth International Workshop on Advanced Context Modelling, Reasoning and Management, September 2004.

[6] M. Laukkanen, “Semantic Web Technologies in Context-aware Systems”, Seminar Report, University of Helsinki, March 2004

[7] E. Franconi, *Lecture Notes on Description Logic*, http://www.inf.unibz.it/~franconi/dl/course

[8] Henry Chang et al, “Complex Event Processing using Simple Rule-based Event Correlation Engines for Business Performance Management”, The 8th IEEE International Conference on E-Commerce Technology, 2006.

[9] Giovanni Bartolomeo et al, “Design and XDI/XRI based implementation of a profile management architecture for next generation networks”, 16th IST Summit, July 2007.

[10] Oliver Holland et al, “Stepping Stones to the Realization of Cognitive Radio”, 13th International Conference on Telecommunications, May 2006.