Converting Raw Sensor Data to Semantic Web Triples: A Survey of Implementation Options

Nikolaos Konstantinou

National Documentation Centre / National Hellenic Research Foundation*
11635, Athens, Greece
Tel: +30 2107273963
nkons@ekt.gr

Abstract. In cases when the information being collected and processed originates by a multi-sensor system, one has to be aware of the impact of the approaches followed in designing the overall system behavior. Typically, there are many steps involved in processing sensory information, from the time a real-world event takes place until the information that describes it and captures its semantics is stored and is available for further exploitation. It this paper we analyze and discuss the options available in the design phase while we underline their respective strengths and weaknesses. The main contribution of the paper lies in the analysis of the factors that affect the behavior and performance of an information system responsible for fusing and integrating sensory information. The main components that constitute such systems are presented in depth, while every choice and the respective effects are discussed in an organized and complete manner.

Keywords: sensor data, semantic web, semantic integration, triples, sensor networks

1. Introduction

During the latest years, we have witnessed a rapid evolution in ubiquitous technologies, leaving back conventional Desktop computing and advancing to the era where pervasive computing is, no doubt, part of everyday experience. Computer-based devices can be seen supporting various activities, based either on user input or on information sensed by the environment. In this paper we discuss the latter case of information generation and the path it follows until it becomes suitable for human consumption. Information that will be contained in a repository originating from sensed data will be: generated by a sensor, communicated, aggregated, fused, integrated, annotated and reasoned about. Of course, not necessarily in this order and not necessarily following each and every step.

Because of rapid evolutions in the domain, various concepts have been developed and studied by researchers: ubiquitous, pervasive and context-aware systems in this paper refer more or less to the same property: The ability of a system of sensing its environment and including these measurements in its behavior.

Among the concepts related to information flow in these systems is the concept of *information fusion*: the study of techniques that combine and merge information and data residing at disparate sources, in order to achieve improved accuracies and more specific inferences than could be achieved by the use of a single data source alone [1][2]. Fusion implies a leverage of information's meaning while in the same time a partial loss of initial data may occur. This loss

* Large part of this research was conducted while the author was with the Athens Information Technology (AIT), 19002, Peania, Greece

does not necessarily degrade the value of the information conveyed. Contrarily, the resulting fused values will convey additional value compared to the initial ones.

Information integration is a concept fundamentally different than fusion. In information integration, the goal is to unify information originating from different sources in order to allow its processing as a whole. Obstacles in information integration can be heterogeneity in the source schemas and data, and differences at a syntactic and semantic level. In a system that offers information integration, a user can submit queries at the system and retrieve results from the source schemas without necessarily he or she being familiar –or even aware– of the heterogeneity in the sources.

It needs to be defined also that integration differs from *merging* because the latter implies unification of the information at the implementation level. Also, *interoperability* is a term often confused with integration while interoperability only refers to the successful information exchange between two systems. Finally, *semantic information integration* is the addition of its semantics in the resulting integration scheme. Compared to fusion, we could state that information fusion takes place in the processing steps while integration refers to the final step: the end user's gateway to (integrated) access to the information.

In this paper we analyze the factors that a system architect must take into account when designing a system that processes sensor information and the effect of the choices in the course of information. We follow the processing steps that sensory data is subject to until it is stored in a data repository/knowledge base and we discuss why there is no "right" or "wrong" decisions since there is no decision entailing only benefits. As analyzed throughout this paper, a system architect should be aware that a system that processes sensory data can be only optimized for a specific subset of the desirable properties.

This paper is structured as follows: Section 2 analyzes and discusses the blocks that compose the end-to-end architecture of a data fusion and integration system. Section 3 discusses the factors that affect the system's behavior and the respective trade-offs. Finally, Section 4 concludes the paper while providing a discussion over our main observations.

2. Fundamental Architectural Blocks

An information fusion and integration system typically consist of (a) sensors, burdened with the task of capturing real-world events, (b) perceptual components whose task is to process the raw sensory data and (c) the middleware system, responsible for administering and orchestrating the entire system. These components are able, with appropriate settings, to carry out the task of monitoring and administering information captured by the sensors.

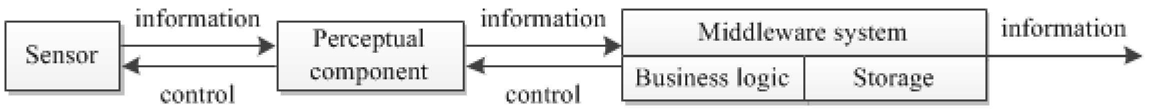

Fig. 1. Components of an information fusion system based on sensory data

We must note that in the cases when a system that deals with data fusion and integration needs to take decisions that affect the behavior and/or performance of its components, the incoming data flow can result in outgoing control commands. We can observe in other words a *duality* in data fusion and resource management, regarding the information that flows in such a system [3]. For instance, in the case when a camera has to be zoomed in or out depending on the state of the world as perceived, the system that supports its operation must allow for control to a certain level. This interaction (information and control) is depicted in Figure 1. Next, we present these fundamental architectural blocks (i.e. the sensors, the perceptual components and the middleware system) in greater detail.

2.1. Sensors

Sensors in general are devices responsible for converting real-world events into electromagnetic signals. Nowadays, a vast variety of sensors exists that can be roughly classified into three large categories. The first category refers to *wireless sensors* that can measure and monitor environmental metrics such as temperature, humidity, location, acceleration, pressure etc. This category includes sensors that are deployed over an area in order to sense and monitor its metrics in the scope of a specific application.

Among the main requirements for this category of sensors is energy efficiency (mostly due to the low capabilities in terms of hardware resources), unattended operation, resistance to adverse weather conditions and communication errors. These requirements have led researchers to propose numerous architectures regarding virtual and physical topology of the sensors, network congestion avoidance [4], routing protocols and routing schemes [5].

The second category refers to *smartphones*, PDAs and other hand-held or embedded devices with communication interfaces such as Bluetooth, GPS, and Wi-Fi. These devices can be considered as sensors, as they are able to communicate location-sensitive data to data aggregators. The difference between the first and second category lies mostly in the fact that the latter typically is more programmable, runs on better hardware and is therefore able to run more demanding applications written for instance in J2ME, Symbian, Android, etc.

The third category refers to *audiovisual* sensors. Cameras and/or microphones can be positioned in order to monitor areas of interest and forward collected data to an upper architectural layer.

We must emphasize that the pervasiveness of sensors nowadays categories above are not mutually exclusive. A temperature sensor can well be wired in a car while in the same time meeting the first category criteria. The lines between categories are even dimmer in smartphones since they can include features from all categories and be used in the same time for monitoring temperature, geo-location, and audiovisual content.

A sensor network can consist of many sensor types, according to the application needs. For instance, an area monitoring application can comprise a fixed-location camera network and mobile agents that can be considered parts of the system, as the case is in [6].

2.2. Perceptual components

After being produced by a sensor, data is subject to process by a *perceptual component*: software that can process a set of data in order to extract higher level information. Perceptual components play an important role in context-aware applications since they are responsible for information enrichment. With the use of such components, collected data can be filtered, annotated or abstracted in order to leverage its meaning and conveyed information. For instance, a perceptual component can be software that recognizes events in a sports game. The input can be in this case a video and the output a set of records such as `(id, event_type, timestamp)`. The information that a multimedia document conveys can be formalized, represented and analyzed with three levels of abstraction: subsymbolic, symbolic and logical [7].

- At the first level, raw data is targeted, and represented in well-known formats. A variety of formats exists at this level, for automatically recognizing and tracking specific features. Face detection [8], face recognition [9][10], object tracking algorithms [11] or speech recognition algorithms [12] operate at this level.

- The next level (symbolic) aims at providing a uniform representation in order to cover the needs for integration of the essentially heterogeneous information originating from various implementations. Standards such as MPEG-7, MPEG-21, NewsML, SensorML and so on, mainly operate at this level. The purpose is to provide homogeneity in annotations in order to combine the emerging results.
- Finally, the goal in the third level is to semantically annotate and enrich information. A variety of approaches that exist in the literature operate at this level [13][14][15].

Perceptual components' capabilities are used in a wide range of applications, since, in collaboration with respective sensors, they can filter and extract valuable information. They are able for instance to extract events of interest in an audio or video stream, measure speed and acceleration of a moving object, monitor environmental parameters such as temperature and humidity, even extract events from AutoID/NFC or RFID tag reads.

2.3. Middleware system

The next component responsible for information processing is the middleware system. As depicted in Figure 1, it can be regarded as a single component whose input is the information generated by the perceptual components and its output is information that combines the inputs of the trackers into a single output that consolidates the incoming information and exposes it for further third party consumption. Additionally, it can offer control over the sensors and perceptual components that are part of the information fusion and integration system. Internally, the middleware can perform a series of housekeeping actions such as to filter the events of interest, remove duplicates, resolve inconsistencies, and minimize ambiguity of information. Examples of available software solutions include GSN (http://sourceforge.net/apps/trac/gsn/) and IrisNet (http://www.intel-iris.net/). A middleware system can be logically regarded to as being consisted of two components, the *business logic* layer and the *storage* layer. We note that the ability of having multiple presentation channels, although technically interesting and challenging, it is out of the scope of this paper's focus.

1) *The business logic layer*: states how information is to be combined and reasoned upon. This layer defines how information is to be processed and what results are to be produced. In this layer, a series of options must be taken in order to tune the solution to the needs of the problem it is intended to solve. These options concern the fusion algorithm employed, the data transformations and they are analyzed in detail in Section 3.

2) *The storage layer*: refers to where the information is kept and maintained. Information storage is a problem especially present in information fusion and integration systems since the nature of the data is such that excess amounts can be collected in small time, depending on the amount of the sensors, the level of detail and the complexity of the observations. The problem that arises is where will all this information be stored.

Technically, storage can refer to middleware-specific solutions involving typically either a relational database or a semantically-enabled solution (triple store or in certain cases quad store). Relational databases constitute a mature, highly popular technology. Triplestores on the other hand are indeed not as mature as databases, but a well-established and widely used, promising technology with a vast potential, targeted at different kinds of application needs. A relational database constitutes an optimal choice when the schema is somewhat fixed and speed is an important factor. On the other hand, a semantically rich solution appears appealing in many cases. A triplestore (or quadstore) is an optimal choice when:

- The schema serves application-specific purposes.
- Collaboration with third parties is seeked/required.
- Inference is required to extract intelligence from the information stored.
- Export access to the data is needed

Additionally, the storage layer may be centralized or distributed. Distributed approaches include Google's BigTable approach [16], Amazon's Dynamo [17], a key-value store optimized for high availability or Apache's Cassandra (also used by Facebook). These approaches abide by Brewer's CAP theorem [18]. CAP theorem is formed by the initials Consistency, Availability, and Partition tolerance:

- A service that is *consistent* operates fully or not at all.
- *Availability* means that the system is available to operate (fully or not).
- *Partition tolerance* can be defined as: "No set of failures less than total network failure is allowed to cause the system to respond incorrectly" [19].

According to the theorem, it is not possible for a distributed system to present all three properties. Still, for transactional data, SQL is a prospective option, since distributed approaches may not opt for consistency. A combined solution would also present powerful results: a cautious approach may be to use a distributed/NoSQL approach for the "extra data" but SQL for the real mission-critical data.

In a way analogous to the duality of data fusion and resource management, we consider the storage layer as layer that exports two APIs: A *query* API through which it can accumulate information and an *export* API that can offer the ability to a third party to export information from the data repository. These API's can be implemented in a number of technologies such as:

- Web services (can be based on SOAP and WSDL or RESTful).
- Sparql endpoints. Using Semantic Web technologies, through these endpoints a client can access the data and its structure.
- Legacy interfaces such as ODBC/JDBC. The reason these interfaces are not preferred over the other methods is because they are costly in terms of time and resources, needed to open a secure connection and transfer the information required.

Middleware system solutions that can be found either closed source or open source include Virtuoso (http://virtuoso.openlinksw.com) and the Talis platform (http://www.talis.com).

3. Factors that affect behavior and performance

The strategy that will be followed in the various steps involved in the information flow affects the overall system response. The main factors where respective decisions have to be taken regard the data fusion algorithm, windowing, context modeling, data transformation and semantic representation, real-time approach, search accuracy sought and fault-tolerance.

3.1. Data fusion algorithm

The most important decision about the fusion algorithm concerns the level at which it operates, regarding which, several levels of fusion can be distinguished. The JDL model designed to describe data fusion offers a generally accepted categorization of the levels at which fusion can take place. However, since this categorization is not concrete (the JDL model appears in the bibliography as having four or five levels according to the context), in this paper we choose to refer to data fusion regarding four levels:

First is the *signal level*. When signals are received simultaneously by a number of sensors, the fusion of these signals may lead to a signal with a better signal-to-noise ratio.

Similarly, fusion can take place at the *feature level*. In this case, a perceptual component must first extract the desired low-level features from each modality and represent them typically in a multidimensional vector. Such examples are presented in [20] and [21] where normalization and transformation matrices are used in order to derive a single ”multimodal” feature vector that will be subsequently used during analysis. However, the abstraction at the feature level may lead to loss of possibly supporting information since the perception of the real world cannot usually be complete with the use of a single medium.

Next, fusion can be realized at the *analysis level*. Fusion at this level is based on acquiring knowledge from several media. Each media has its own “knowledge” of the world but only by fusing at this level we will be able to extract knowledge that would be impossible to identify using a single medium.

Ultimately, fusion at the *decision level* involves combining information from multiple algorithms in order to yield a final fused decision. Fusion at this level may be defined by specific decision rules.

Another important property concerns the algorithm perception of the world. Consider a fusion system consisted of a sensor network that collects information. Using an online (or distributed) algorithm, each node can take decisions based only on its perception of the world. In this case, each algorithm execution is based on the knowledge of only a local node or a cluster of nodes. Contrarily, in offline (or centralized) algorithms there is a need of a central entity maintaining system-wide information.

The choice of a *sensor network topology* needs to be taken in parallel with the choice of the level at which fusion will take place. Regarding the topology, we have in general three categories, namely the serial, star, tree, and mesh topology as depicted in Figure 2. Gray nodes depict sink nodes; nodes with more enhanced capabilities in terms of processing power, storage, bandwidth and autonomy.

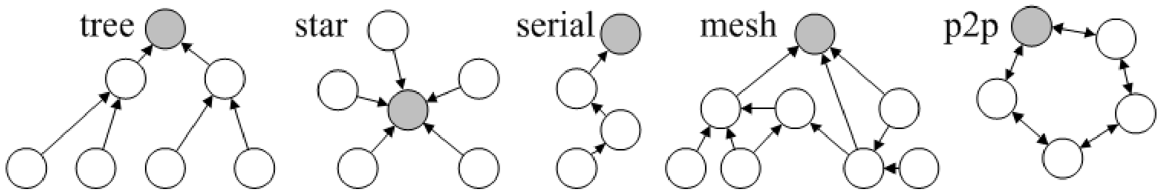

Fig. 2. Sensor network topologies

Synchronization is also an important decision that needs to be taken when designing time management. The approach followed in timestamping the generated messages and events may be not as trivial as it seems. A system for fusing information might have rules of the form:

if *event*$_1$ occurred before *event*$_2$ **then**…

In such cases, deviations from a common clock may lead to irregularities. If we consider a fusion node in a fusion graph, the events that are generated can have two kinds of timestamps:

- The time the event was recognized: the local time in the node that measured the property. In this case, the event is timestamped close to its actual creation, which is ideal since it reflects the real world conditions. The problem that arises in this case is how to synchronize the sensor network to a common clock.
- The time the event arrived in the fusion node (such as a gray node in Figure 2). This revokes the need of a common clock, since the fusion node will timestamp events upon their arrival. However, this approach should be followed when there are no great delays in communicating messages between nodes.

In the same sense, the fusion algorithm can operate either locally at each node or remotely.
A fusion node can:

- Act as a server, waiting to be contacted. In this case, the information flow is asynchronous. The system can be considered as operating in *push* mode in the sense that information is *pushed* from the sensors to the fusion node. This operation mode is important when the results need to be communicated to the user as soon as possible.
- Harvest information that is collected in the distributed sensors. In this case, the information flow is synchronous. This mode of operation is referred to as *pull* mode as opposed to the *push* mode.

3.2. Windowing

In cases when streaming data is generated at high rates, a mechanism needs to be developed to assure its continuous process. For the newly generated information, it needs to be assured that in order to process it properly, the system will not have to take into account all existing information. In other words, it is important that the system maintains a working memory window since, by definition, streams are unbounded and cannot fit into memory in order to be processed as a whole. Regarding windowing, there are three important decisions that have to be taken:

1) *The measurement unit.* The window is defined in certain units that can be for instance tuples in a relational database or triples in a semantic web triplestore. Units can be physical, as the aforementioned or logical, defined in terms of time (e.g. milliseconds) [22]. Time-based windowing, however, tends to produce more fluctuation in the response times [23].
2) *The size.* The window size increases proportionally the system latency. Therefore, a large window size in a fusion node, although it will reflect a more complete perception of the world, it may reduce its performance. Therefore, the window size has to be balanced according to the accepted latency. On one hand, having all the data available would be ideal but on the other hand, resources in terms of processing capabilities and storage are limited and latency should always be kept to an acceptable level. This could be remedied with the adoption of hierarchical or incremental processing approaches [24] which enable taking into account the whole window but regard only the latest points.
3) *The window behavior.* Memory windows, depending on how they are updated, they can be categorized as follows:
   - Sliding windows have a fixed length and both their endpoints are moving at a predefined interval, replacing old stream elements with new ones.
   - Tumbling windows can be considered as a special case of sliding windows, where the window is moving at a fixed interval of length equal to the window size. Thus, tumbling windows are pairwise-disjoint and do not share any stream elements. This behavior, although convenient for a large number of cases, it is not always optimal since window moving operations could take place at inconvenient times (e.g. during the occurrence of an event of interest) leading to incorrect results [23].

- Landmark windows maintain a fixed beginning endpoint while the other one is moving, resulting in a variable window size [25].
- Partitioned windows dictate that windows are formed according to attributes of interest [26].
- Predicate windows, a model introduced in [27], limit the stream tuples to the focus of a continuous query, in a manner similar to the partitioned windows.

Regardless to the windowing behavior, older elements that are dropped out of the window are discarded, leading to an obvious loss of information. This imposes restrictions in the expressiveness of the rules that we are able to define and which shape the system's behavior, i.e. the rules that process newly generated information. Because at all times only the latest information is available, we cannot declare rules that have to take into account older –discarded or archived– information. The rules applied to real-time are restricted to the current information window. Specifically, in order to state a rule that needs to take into account archived information, each rule execution would impose extra computational burden. Therefore, among the (necessary!) drawbacks of maintaining a window of information we have to include the reduction in the rule expressiveness in the window. Of course, this does not restrict the expressiveness of the rules that can process archived information.

3.3. Modeling context

Modeling context, a system's perception of the world, is hardly a trivial task. Context refers to "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [28]. The challenge in modeling context is in the complexity of capturing, representing and processing the concepts that comprise it. The description of the world must be expressive enough in order to enable specific behaviors but not as complex as to render the collected information unmanageable.

Consecutively, one of the most challenging issues of context-aware applications is the inclusion of intelligence while processing the incoming information and deducting meaning. Ontology-based descriptions help toward this direction since they offer unambiguous definitions of the concepts and their relationships.

The authors in [29] follow an approach based on describing the knowledge base model using *resource*, *actor*, *environment* as its base concepts. As the authors describe, information stored in a context model can be logically organized in:

- parts that describe the state of the resources (the resource part)
- the entities operating in the resources (the actor part), and
- the surrounding environment conditions (the environment part).

Regarding the vocabulary used in order to provide a uniform ontological representation, numerous ontologies have been proposed in the literature such as OpenCyc (http://www.opencyc.org), the Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA), Situation Awareness Ontology (SAW) [30], or the one proposed in [31], where the authors claim that among the benefits of using this ontology is the maximization of the precision of searching sensor data by utilizing semantic information.

In the case when contextual information needs to be transferred among applications, a common representation format should be followed. In this case, a semantic web compliant ontology-based approach can ensure interoperability –at syntactic and semantic level– while enabling integration with third party data sources. Uniform context representation and processing at the infrastructure level enables better reuse of derived context by multiple data producers and consumers. In addition, such approaches allow further exploitation of the knowledge base through the possibility of submitting and evaluating higher level intelligent semantic queries.

As far as Semantic Sensor Networks are concerned, several ontological approaches have been proposed in the literature, aiming at covering descriptions of the most widespread concepts in the domain. The SSN ontology [32] suggested by the W3C Semantic Sensor Network Incubator Group is highly relevant since it attempts to become a standardized approach for describing sensor data in the Linked Sensor Data context, bridging the Internet of Things and the Internet of Services.

Context modeling in sensor data fusion and integration systems can be considered as a special case of the data integration problem. In the data integration problem, we have the source schemas (the schema that describes the data of a node) and the target schema (or global schema, the schema that describes the overall system's data). In practice, schemas typically refer to relational database schemas, semi-structured information or ontologies. According to the data integration problem, we can consider two distinct types of approaches: the global-as-view (GAV) and the local-as-view (LAV) approach, while the idea is to provide a uniform query interface over the data. In the GAV setting, the global schema is modeled as a set of views over the schemas of the sources. The major drawback of the GAV approach is that it is necessary to redefine the view of the global schema every time a new source is integrated and therefore it is an optimal choice when the source schemas are not subject to frequent changes. Reversely, in the LAV setting, the source database is modeled as a set of views on the global schema.

In the case of a sensor fusion system where new types of sensors need to be integrated and there is a relatively high turbulence in its structure, a LAV setting is considered optimal. In other words, there should be a global schema to which every sensor node should conform. This approach is preferred to translating user queries from the global schema to each one of the local ones in order to retrieve, aggregate and return results to the user.

3.4. Transformations between technical spaces

Information that is collected by sensors will undergo a number of transformations and be subject to processing before it will finally lie in the system's repository. Each step in its flow needs to have well defined semantics regarding its representation and its transformations that can take place.

In order to specify the context encompassing specific distinct approaches, the term of *technical spaces* has been introduced: "A technical space is a working context with a set of associated concepts, body of language, tools, required skills and possibilities" [33]. Although it is difficult to formally define technical spaces, we can easily recognize them: XML-based languages (XML TS), Data Base Management Systems (DBMS TS), Ontology TS (OTS) are some of the technical spaces mostly in use.

Distinct phases in data transformation and process can use different technical spaces. For instance, in Figure 1, the distinct components that comprise the system can employ different technical spaces.

First, the sensor feeds the perceptual component with raw, sensor data streams. The perceptual component is responsible for processing and fusing the data in order to extract meaningful information, i.e. leverage its representation to (semi)structured information. This information will be processed by the middleware solution according to its business model in order finally to occupy space in the storage repository.

In each process step, the form of the language and the respective technical space may vary in its appearance but it can be defined by conventional grammars, by DTDs or XML schemas, by ontologies, or by metamodels in general.

The Open Geospatial Consortium (OGC) members deal with specifications regarding interoperability interfaces and metadata encodings that enable transformations and transitions between technical spaces. For this reason, the consortium has developed and tested among others specifications like (a) the Observations & Measurements (O&M), a model and XML schema for encoding observations and measurements from a sensor, both archived and real-time and (b) the Sensor Model Language (SensorML), a model and XML schema for describing sensor systems and processes associated with sensor observations. These processes concern discovery of sensors, location of sensor observations and processing of low-level sensor observations. Regarding sensor data representation using Semantic Web technologies, the authors in [34] propose an approach to annotating sensor data with spatial, temporal, and thematic semantic metadata.

In order to further clarify technical space concerns, let us consider an example of a sensor that produces raw data using e.g. a struct in C language. In order to store the measurements, one approach can be to store the measurements in XML files and from there to the relational database. But, avoiding the XML step would improve system speed so this step should be only included when these XML files serve other purposes as well besides simple transformations between the distinct technical spaces. The same applies to the ontology space as well. In conclusion, inclusion of various technical spaces in the information flow should be justified and, of course, avoided if not completely necessary.

3.5. Real-time vs. near-real-time

A real-time system is one that must satisfy explicit bounded response time constraints to avoid failure and present consistency regarding the results and the process time needed to produce them. The fundamental difference between a real-time and a non real-time system is the emphasis in predicting the response time and the effort in reducing it. In [35], a system's *response time* is defined as "the time between the presentation of a set of inputs and the appearance of all the associated outputs". Therefore, in a sensor fusion system that, for instance, produces certain alerts, the notion of real-time dictates that these alerts (outputs) are connected to the appearance of certain inputs (events).

Systems that schedule their operations at fixed time intervals cannot be considered realtime. "Near real-time" is a characterization that can be applied if these intervals are frequent enough. For a system that employs semantic web technologies, this means that unless the entire Knowledge Base is reloaded and reasoning procedures are performed on it, we cannot consider that all the associated outputs are produced and the system cannot be classified as real-time.

More specifically, when new facts are appearing in the system, one can choose whether to just store them and process them at a later time, asynchronously, or to calculate the changes reflected at the entire Knowledge Base at once. The former approach is not real-time, although it can produce near real-time results if the calculations are performed frequently enough. The latter is the real-time approach and would perform optimally at low data incoming rates or small computational overhead.

3.6. Precision vs. recall

In a search system, *precision* is defined as the number of items correctly returned over the number of the returned items. *Recall* is defined as the ratio of the items correctly returned over the number of the items that should be returned. These two parameters are by definition reversely proportional. In the design phase of a system, therefore, a choice must be made between precision and recall.

Web search engines (e.g. Google, Yahoo!, Bing) where speed is an important factor are optimized for high precision. The end users to whom the applications address to are not willing to wait. Therefore, in the small time between the query and its results, the system must be precise, i.e. relevant to what the user wants to find.

Reversely, when a query is evaluated, for instance in an audiovisual archive looking for events of interest, the nature of the application most probably allows higher response times in the sense that its end users are willing to wait more. This allows a system architect to optimize for high recall or, in other words, assure that a query returns all relevant results, while sacrificing speed. However, under all circumstances and especially in systems with large collections of data, imperfect annotation may be preferable over no annotation at all [36].

3.7. Dealing with node misbehavior and network problems

An important task when designing a sensor information fusion system is to assure the system's unhindered operation. The system's robustness can be affected by two kinds of failures: failure of certain sensors/perceptual components or failure of the underlying protocols for various reasons.

Failures by the trackers can be caused in two cases. First, a perceptual component may fail to capture a real world event. This case, although it does not affect the system's performance, it can be dealt with by applying rules at a higher fusion level. Especially when the properties monitored must present continuity in their values, such as the trajectory of a moving object, disappearance and appearance at a later stage can be assured at the semantic level. In the same sense, measurements that are highly deviated from a flow of values can be discarded at a higher level. In the same manner, the system can treat false positives possibly generated by perceptual components. Of course, this consistency assurance is an additional computational burden to the system.

Second, a misbehaving component may generate an overwhelming number of messages and flood the system. The countermeasure that can be taken in this case is the creation of buffering or caching mechanisms. In cases when part of the information available is sufficient for fully perceiving the world's state, sampling could be applied.

Failures of the underlying protocols may first be caused by network errors. Especially in the case of wireless sensor nodes, the operation of the communication protocol (ZigBee, Wi-Fi) can impose a serious impedance in the system's operation. Network problems can cause skewing and multiple delays that will affect performance. In this case, message synchronization is important in order to assure information integrity. Under all circumstances, the above-mentioned are problems that to a certain extent will happen and therefore respective countermeasures have to be devised.

4. Discussion and Conclusions

As defined throughout the paper, there are many steps and many components involved in processing the information generated in the system. Additionally, there is no straightforward approach in configuring every component. Therefore, there can be no deterministic manner in setting the priorities in the resulting system's behavior. However, our main observations are as follows:

- *Scalability* is an important system property in terms of (a) the volume of information and (b) the number of sensors/perceptual components. Unless properly handled, information generated in such systems can quickly rise to difficultly manageable volumes, especially when there are audiovisual sensors involved. In order to assure scalability it is necessary to configure the sensors to report at long time intervals, as long enough as to assure meaningfulness. Next, fusion at low level can trim information not worth being stored. Next, the time window of the perceptual components can be limited to a level low enough to assure adequate results. Finally, near-real-time approaches are more promising since following this approach as explained in Section 3.5 every single measurement does not affect the whole system state but part of it.

- In cases when the goal is to perform analytical reasoning and intelligent analysis (such as data mining, pattern extraction) over the collected data, adoption of semantic web technologies is recommended, since it will enable inference that will deduce implicit information. Moreover, in cases when *integration* with third party software is important, semantic web technology adoption will assure unambiguous definition of the information and the semantics it conveys. While being aware that semantic web technologies do reduce energy-efficiency and autonomy because of the extra computational burden they constitute, once the system's data repository is turned into a knowledge base, the possibilities for further exploitation are virtually endless.
- Assurance of high *availability* in the system entails that we will have to choose between information consistency and partition-tolerance. In order to assure that the system is up and running at all times, the knowledge base generated will either have to be partitioned or not searched completely. An approach can be in this case to partition the system and allow information to be asynchronously disseminated in the system (eventually consistent).
- Accuracy in the *search mechanism* first, depends on the time users are willing to wait for the results. As analyzed in Section 3.6, longer times will allow for greater recall in the results while in cases similar to Web search engines, smaller waiting times entail high precision and low recall. Second, in cases when storage is distributed, the CAP theorem (see Section 2.3) dictates that consistency can only be combined with either availability or partition-tolerance. Also, precision and recall depend on the windowing approach in the fusion algorithm. Smaller windows will decrease precision but keep response times lower since the amount of information that will be searched will be reduced.
- Regarding the *technical spaces* employed, the need for scalability dictates that closer to the sensors the approach should be based on more lightweight technologies. This approach also favors energy-efficiency when this requirement is present. Under all circumstances, transformations between technical spaces and ways to deal with node misbehaviors are an extra non-negligible burden. The same holds for largely descriptive context models.

To sum up, in order to deploy successful solutions, one must be very careful in choosing the building blocks. Sensors, perceptual components and a middleware system have to be orchestrated in order to make the most out of every bit in the information flow. However, the choices in designing every component are so many, that the system architect should evaluate and order the system's requirements according to their importance before designing. At all times, it should be kept in mind that there is no single approach that can offer at the same time scalability, energy efficiency, and an accurate search mechanism.

References

[1] Sheth, A, Larson, J. Federated database systems for managing distributed, heterogeneous, and autonomous databases. In: ACM Computing Surveys, 1990, 22 (3): 183–236.
[2] Llinas, J, Waltz, E. Multisensory Data Fusion. Artech House, Inc., 1990.
[3] Bowman, C. The data fusion tree paradigm and its dual. Paper presented at the 7th National Symposium on Sensor Fusion, March 1994.
[4] Ahmad, M, Turgut, D. Congestion avoidance and fairness in wireless sensor networks. In: IEEE Global Telecommunications Conference (GLOBECOM 2008), December 2008, Pp. 1–6.
[5] Shah, R, Rabaey, J. Energy aware routing for low energy ad hoc sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC2002), 2002, Pp. 350–355.
[6] Doulaverakis, C, Konstantinou, N, Knape, T, Kompatsiaris I, Soldatos, J. An approach to Intelligent Information Fusion in Sensor Saturated Urban Environments. In: European Intelligence and Security Informatics Conference (IEEE EISIC 2011), Athens, Greece, September 2011.
[7] Stamou, G, van Ossenbruggen, J, Pan, J, Schreiber, G, Smith, J. Multimedia Annotations on the Semantic Web. In: IEEE Multimedia, 2006, 13 (1): 86–90.
[8] Viola, P, Jones, M. Rapid Object Detection using a Boosted Cascade of Simple Features. In: IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, December 2001, pp. 511–518.
[9] Turk, M, Pentland, A. Eigenfaces for recognition. In: Journal of Cognitive Neuroscience, March 1991, 3 (1): 71–86.
[10] Bartlett, M, Movellan, J, Sejnowski, T. Face Recognition by Independent Component Analysis. In: IEEE Transactions on neural networks, November 2002, 13 (6): 1450–1464.
[11] Allen, B, Bishop, G, Welch, G. Tracking: Beyond 15 minutes of thought. In: SIGGRAPH Course Pack, 2001.
[12] Sohn, J, Kim, N, Sung, W. A Statistical Model-based Voice Activity Detection. In: IEEE Signal Processing Letters, 1999, 6 (1): 1–3.
[13] Schroeter, R, Hunter, J, Guerin, J, Khan, I, Henderson, M. A Synchronous Multimedia Annotation System for Secure Collaboratories. In: second IEEE International Conference on e-Science and Grid Computing (E-SCIENCE’06). Washington, DC, USA: IEEE Computer Society, December 2006, Pp. 41.
[14] Petridis, K, Anastasopoulos, D, Saathoff, C, Timmermann, N, Kompatsiaris, I, Staab, S. M-OntoMat-Annotizer: Image Annotation. Linking Ontologies and Multimedia Low-Level Features. In: Engineered Applications of Semantic Web Session (SWEA) at the 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES’06), Bournemouth, UK, October 2006.
[15] Chakravarthy, A, Ciravegna, F, Lanfranchi, V. Cross-media document annotation and enrichment. In: first Semantic Web Authoring and Annotation Workshop (SAAW2006), 2006.
[16] Chang, F, Dean, J, Ghemawat, S, Hsieh, W, Wallach, D, Burrows, M, Chandra, T, Fikes, A, Gruber, R. Bigtable: A distributed storage system for structured data. In: Proceedings of the 7th Conference on Usenix Symposium on Operating Systems Design and Implementation, 2006, 7: 205–218.
[17] DeCandia, G, Hastorun, D, Jampani, M, Kakulapati, G, Lakshman, A, Pilchin, A, Sivasubramanian, S, Vosshall, P, Vogels, W. Dynamo: Amazons Highly Available Key-Value Store. In: 21st ACM Symposium on Operating Systems Principles (SOSP 2007), Stevenson, Washington, USA, October 2007, Pp. 205–220.
[18] Brewer, E. Towards robust distributed systems. In: Proceedings of the 19th annual ACM symposium on Principles of distributed computing. Portland, Oregon, United States: ACM, 2000, Pp. 7–10.

[19] Gilbert S, Lynch, N. Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services. In: SIGACT News, June 2002, 33(2).
[20] Li, D, Dimitrova, N, Li, M, Sethi, I. Multimedia content processing through cross-modal association. In: ACM International Conference on Multimedia, Berkeley, California, USA, November 2003.
[21] Wu, Y, Chang, E, Chang, K, Smith, J. Optimal multimodal fusion for multimedia data analysis. In: ACM International Conference on Multimedia, New York, USA, October 2004.
[22] Patroumpas, K, Sellis, T. Window Specification over Data Streams. In: International Conference on Semantics of a Networked World: Semantics of Sequence and Time Dependent Data (ICSNW’06). Springer, 2006, Pp. 445–464.
[23] Konstantinou, N, Solidakis, E, Zafeiropoulos, A, Stathopoulos, P, Mitrou, N. A Context-Aware Middleware for Real-Time Semantic Enrichment of Distributed Multimedia Metadata. In: Multimedia Tools and Applications, 2010, 46 (2-3): 425–461.
[24] Barbieri, D, Braga, D, Ceri, S, Della Valle, E, Grossniklaus, M. Incremental Reasoning on Streams and Rich Background Knowledge, In: The Semantic Web: Research and Applications, Lecture Notes in Computer Science, 2010, 6088: 1–15.
[25] Arasu, A, Babu, S, Widom, J. The CQL Continuous Query Language: Semantic Foundations and Query Execution. In: The VLDB Journal, 2006, 15 (2): 121–142.
[26] Li, J, Maier, D, Tufte, K, Papadimos, V, Tucker, P. Semantics and Evaluation Techniques for Window Aggregates in Data Streams. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD’05), 2005, Pp. 311–322.
[27] Ghanem, T, Aref, W, Elmagarmid, A. Exploiting Predicate-Window Semantics over Data Streams. In: ACM SIGMOD Record, 2006, 35 (1): 3–8.
[28] Dey, A. Understanding and Using Context. In: Personal Ubiquitous Computing, 2001, 5(1): 4–7.
[29] Toninelli, A, Montanari, R, Kagal, L, Lassila, O. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In: The Semantic Web – ISWC 2006, Lecture Notes in Computer Science. Springer, November 2006, Pp. 473–486.
[30] Matheus, C, Kokar, M, Baclawski, K. A core ontology for situation awareness. In Proceedings of the Sixth International Conference on Information Fusion, 2003, Pp. 545–552.
[31] Eid, M, Liscano, R, Saddik, A. A universal ontology for sensor networks data. In: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA’07), June 2007, Pp. 59–62.
[32] Lefort, L, Henson, C, Taylor, K (Eds), Incubator report, available online at http://www.w3.org/2005/Incubator/ssn/wiki/Incubator_Report , 2011.
[33] Kurtev, I, Bézivin, J, Aksit, M. Technological Spaces: an Initial Appraisal. In tenth International Conference on Cooperative Information Systems (CoopIS 2002), International Symposium on Distributed Objects and Applications (DOA 2002), Federated Conferences, 2002.
[34] Sheth, A, Henson, C, Sahoo, S. Semantic sensor web. In: IEEE Internet Computing, 2008, 12(4): 78–83.
[35] Dougherty, E, Laplante, P. Introduction to Real-Time Imaging. Wiley-IEEE Press, chapter What is Real-Time Processing?, October 1995, Pp. 1–9.
[36] Dou, D, Pan, J, Qin, H, LePendu, P. Towards Populating and Querying the Semantic Web. In: Proceedings of 2nd International workshop on Scalable Semantic Web Knowledge Base Systems (SSWS’06), co-located with ISWC 2006, November 2006, Pp. 129–142.