A Distributed Framework for Organizing an Internet of Things

Jamie Walters[1], Theo Kanter[1], and Enrico Savioli[2]

[1] Mid Sweden University - Sundsvall 85170, Sweden
{jamie.walters,theo.kanter}@miun.se
[2] University of Bologna - Bologna, Italy
enrico.savioli@studio.unibo.it

Abstract. Applications on a future *Internet of Things* require the provisioning of current, relevant and accurate context information to end-points. Context information existing globally require organization into object-oriented models available locally in APIs as current, relevant and accurate views. Moreover, such applications require support for the highly dynamic interactions influencing continual changes in global context information. Existing approaches, such as the web services, are unable to provide this support partly due to the presupposed existence of a network service brokering context information, relying on DNS; or adopting a presence model for context which does not adequately scale. To this end, we propose a distributed framework for the interconnection of end-points and co-located agent entities, whereby agents are provided with local views of a relevant subset of global context information. We show how to achieve relevant local current views of global context information via ranking in an object-oriented context model. The distributed approach realizes the provisioning of context information in real-time, i.e., with predictable time bounds. Finally, we demonstrate the feasibility of the approach in a prototype based on P-Grid.

Keywords: mediasense, dcxp, p-grid, context proximity, sensor ranking, context metrics, context distance, sensor ranking.

1 Introduction

The increasing interest in the provisioning of applications and services that deliver experiences based on context mandates the continual research into methodologies, architectures and support for delivering the context information required. Constraints on service delivery with respects to real-time availability underpins any such solution. A future connected things infrastructure with an installed device base exceeding billions [1], requires support for a wide range of context centric experiences ranging from personalized and seamless media access, to intelligent commuting and environmental monitoring. This incorporate devices such as mobile phones, personal computers or IPTV boxes; all merging towards the paradigm of *everywhere computing* [2]; the seamlessly connected new world. As users navigate a vast and seemingly endless connected things infrastructure, it

J. Del Ser et al. (Eds.): MOBILIGHT 2011, LNICST 81, pp. 231–247, 2012.

becomes increasingly important to be provisioned with the relevant subsets of information required in order to be enabled with an experience relative to the users' current situation.

1.1 Scenario

John is constantly on the move both for business or pleasure. Within a future cityscape, he encounters multiple information points which maybe used to inform him of the state of his surroundings. Embedded into a digital ecosystem, he is capable of deriving enough information in support of the services wishing to effect changes or deliver him a unique context-based user experience. Such information include temperature, humidity and location as well more complex sources such as audiovisual devices, network connections or traffic conditions. With his smartphone, John is able to connect to and derive representations of context from these points in order to support his applications.

1.2 Analysis

Applications and services wishing to respond relevant to John's current context require this information to be organized and made available in globally accessible end-points. Approaches such as IMS [3] or Senseweb [4] enables the required support, brokering context information via web service portals on the Internet They are, however dependent on DNS as a means of locating service portals, users and applications. Issues with DNS availability due to DoS attacks and configuration errors raises questions about its continued suitability and prompting research into Distributed Hash Table (DHT) based overlays such as Chord [5], Pastry [6] and Tapestry [7] as possible replacements [8].

To this end, early work surrounding the MediaSense architecture implemented the DCXP protocol [9], a Chord based approach capable of provisioning John's context information in support of his dependent applications and services. The DCXP approach produced the ranges in response times deemed adequate enough to support real-time context dependent services. Furthermore, it proved that distributed systems were more capable of achieving this than approaches building on mobile or web services. Other approaches such as [10] explore the option of building context provisioning solutions using DHTs. However, while a DHT provides for a more scalable and resilient approach, it relies on deterministic hashing algorithms for achieving the distribution, indexing and locating of information.

Consequently, this places a limit on their ability to utilize self-organization towards realising a more homogeneous distribution of information located on the overlay. With regards to the persisting of context information, an additional disadvantage of DHTs is their inability to support queries of a range of values, critical in scenarios where John might be trying to locate a service in some approximate area or over a series of context values. While solutions such as [11] have sought to address this problem, this is not done natively, mandating the implementation of additional layers of complexity on the existing overlay. DHT-based implementations such as Chord are limited to a searching complexity of

$O(logN)$ [12]. However solutions seeking to provision John with current and relevant information mandates investigations into alternatives capable of realising improved response times.

We are further mandated to organize John's context information in subsets representative of the dynamic state of the sensor information to which he has access. As John changes state in real-time, such as entering a building or vehicle, he will encounter new sensors or sensor information. This requires that a schema of available context information be maintained and kept current; an evolving meta model as suggested in [13].

In approaching such a possible solution, the use of distributed relational databases such as in [14] and making use of the advanced research in database distribution would not be applicable. This, as such a distribution assumes communication reliability in order to maintain database integrity across wide area networks which cannot be guaranteed in heterogeneous mobile scenarios [15], [16]. This is also undermined by the fact that relational databases are highly inefficient for supporting real-time data manipulation, evolution and querying.

Current metric approaches such as Internet search engines consider the theory of connected things, however relative to static document content. A document's connectivity determines its relevance. This concept of ranking has been explored and used both in a centralized [17] as well as distributed [18] solutions. However, centralized solutions such as Google index only a tiny portion, less than 10 billion of the estimated 550 billion pages, on the relatively static Internet [19]. Any attempt to apply such a centralized solution to the ranking of sensors in an *Internet of Things* would be undermined by its inability to scale well. Distributed solutions based on the PageRank [20] concept would not scale well to accommodate highly dynamic document sets. Current *real-time* searches are realized by targeting known content providers, an approach that could not scale to accommodate the vast and mostly ad-hoc nature of a connect things infrastructure.

In this paper, we present an alternative approach that permits users to browse and locate relevant information in a vast and dynamic *Internet of Things*. Solutions capable of providing broad access to context information and enabling the derivation of context-based metrics. Such metrics include a sensor ranking and context proximity metric detailed further in Section 3. We therefore revisit the MediaSense Framework in an attempt to provide the approach required to support such user activities within real-time. Key to this is our new approach to the overlay structure, substituting Chord with a more resilient and robust P-Grid overlay.

For the remainder of this paper, Section 2 details the revised architecture; Section 3 presents an overview of the metrics while Section 4 summarizes our conclusion and future work.

2 The MediaSense Framework

In response to the shortfalls discussed in Section 1, the MediaSense framework seeks to create a solution towards supporting an *Internet of Things*. A solution

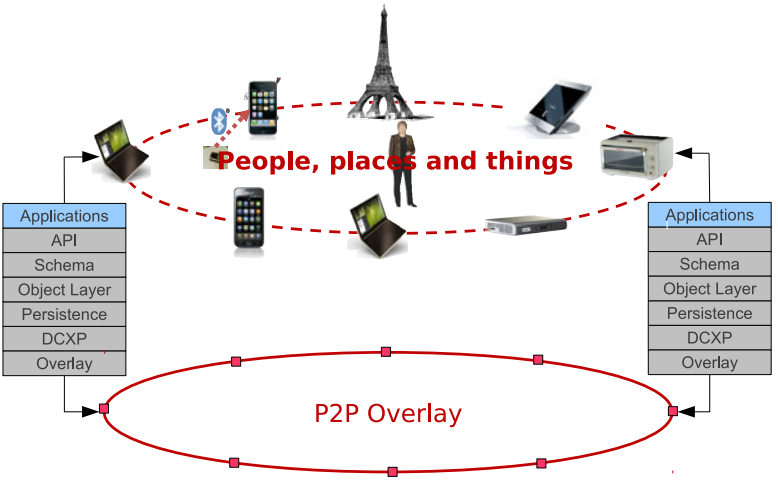

Fig. 1. The MediaSense Framework

in which presentities [21] are regarded as the focal point, enabling support for their dependent applications and services. This from the information gleaned from their interactions and associations within such a digital ecosystem.

Early work on the MediaSense framework realized an architecture for the distributed provisioning of user sensor information within real time constraints, providing the foundation for further work towards supporting the browsing of the dynamic data and interactions existing on an *Internet of Things*.

Our revised solution entails multiple layers of abstraction, enforcing layer logic independence. As a completely decentralized solution, nodes are permitted to freely participate, and realize the components required to supports its functions. Information Points, such as sensors, actuators or even an audio stream, can be registered by a node and be made available for usage at any layer across the solution. This is used to support an application layer exposed to applications and service providers for accessing the framework's functionalities. This masks the complexity of lower layers and their interactions, enabling users to focus on developing context objects, applications or services; having them transparently shared across the network with relative ease.

This components are illustrated in Figure 1 and are detailed in the remainder of this chapter.

2.1 The Overlay

The ability to provision context-centric user experiences from distributed information mandates an underpinning distributed overlay. Our previous work was supported by a Chord based [5] implementation used for maintaining the backbone communications as well as providing an indexing mechanism for information that must be persisted amongst participating nodes, also called

Context User Agents (CUAs). As with typical peer-to-peer protocol implementations, the nodes participating within the overlay act as entry points for applications and services wishing partake in the provisioning of sensor information across the overlay.

Citing issues with DHT based overlays as discussed in Section 1, we have substituted Chord with P-Grid [22] as the overlay of choice.

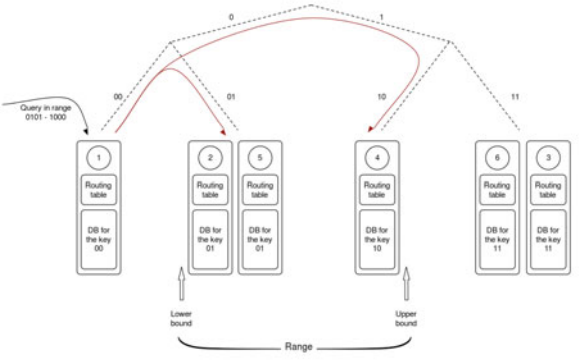

Fig. 2. The P-Grid Distributed Tree Structure

P-Grid. In an effort to increase the functionality of the architecture, we saw the need to move away from a DHT based implementation to overlay structures offering improved resilience, distribution and self organization. P-Grid, as the overlay of choice, shares a common behavior to DHT based implementation with respect to being able to index and locate information. P-Grid, however realizes a distributed binary tree, illustrated in Figure 2.

The key space is partitioned among all the nodes and organized into a tree structure with each node's location determined by the binary bit string representing the set of values for which the node is responsible. With this, it preserves the ordering on data and natively enables the resolution of both specific key, substring and range queries without any pre-existing knowledge. This is achievable with at most the same message complexity of most DHTs and has proven performance of $0.5logN$ versus $logN$ for a Chord based implementation.

The non-deterministic distribution of keys, offers improved resilience in the dynamic environments which are expected to exist in a future *Internet of Things* as it permits a more flexible self-organization in response to a very dynamic set of information. This is complemented with redundancy for fault tolerance; multiple nodes are assigned to the same key partition and nodes hold references to multiple partition holders. As a relatively future proof overlay, it readily permits future extensions and modifications.

While the overlay permits users to make sensor distributed information available his is not sufficient to support the interconnected and evolving *Internet of Things*. It is instead necessary to exploit the overlay as a building block and to rely on a protocol that focuses on data dissemination across interested nodes.

2.2 Distributed Content Exchange Protocol(DCXP) Layer

The DCXP layer primarily deals with the realization of such a protocol. One that permits users to publish and access context information in a structured manner, enabling the enforcement of some access controls. Residing immediately on top of the overlay, DCXP is an application-level protocol designed with the goal of enabling nodes to share and move context information between peers. As with the early implementations of the MediaSense architecture, this layer implements the core protocol employed in the provisioning of context information. These are summarized in Table 1.

One key departure is that the protocol is no longer used to maintain the overlay structure. With this task managed solely by the P-Grid overlay, network composition and state is abstracted from the protocol layer. The functionality of the protocol is now resigned to realising a distributed *publish/subscribe* interface to the resources available on the overlay. To this end, we introduce two new primitives: *TRANSFER* and *SET*.

The *TRANSFER* primitive provides for the ability to relocate context resources in support of applications and services.This is used by the object layer discussed later in Section 2.4. When a node requires a sensor resource that is not locally available, it makes a *TRANSFER* request, the object is then copied and used locally, reducing network messaging overhead, and the considerable demands that can be placed on nodes responsible for a context resource. Such an action could be achieved autonomously or in response to the application requirements.

The *SET* primitive enables interaction with actuators in end points completing the sensor/actuator pair, allowing applications to influence context in response to user preferences and context information.

The *publish/subscribe* functionality of the DCXP protocol is realized through a group of components namely, the Context User Agent, the Context Storage and the use of a Universal Context Identifier. These are discussed further in Section 2.2.

The resulting overlay, built using P-Grid, is used solely for maintaining the connection between nodes, and persisting the data registered by applications and services residing at the nodes. Such nodes, may not need to actively participate in the overlay, as would be the case with mobile devices over heterogeneous and sometimes unreliable connections.

Context User Agent - CUA. A computer wishing to participate within the context provisioning overlay is only required to implement an instance of the CUA. The CUA permits the seamless exchange of context information among sources and sinks, as well as the interaction with the sensors and actuators. Each CUA corresponds to a node on the virtual distributed tree described in section 2. Each CUA further contains some persistence in the form of object oriented databases (OODB) along with an API for creating applications and services consuming and responding to sensor information. It further provides the entry point for registering and resolving a UCI or a query across the CS.

Table 1. The Primitives of the Distributed eXchange Protocol

REGISTER_UCI	Registers a UCI along with the node which is responsible for it.
RESOLVE_UCI	Resolves a UCI to the node which is responsible for it.
GET	Fetches the current context value from the node responsible for a UCI. The reply is sent using a *NOTIFY*.
SET	Changes the current status of an actuator in an end point.
SUBSCRIBE	Makes a subscription request to the node responsible for a UCI, The node then sends a *NOTIFY* message containing the current context value, either at regular intervals or when the value changes.
NOTIFY	Notifies an interested node of the current context value associated with a specified UCI.
TRANSFER	Requests the manager of a resource to transfer responsibility to another node. This might be full responsibility or partial, where the requester re-creates a copy of the resource permitting improved real time performance.

Universal Context Identifier - UCI. All resources on the overlay is persisted using the universal context identifier (UCI) naming scheme. The UCI naming schema provides a URI inspired naming schema with the following syntax:

dcxp://user[:password]@domain[/path[?options]]

where *dcxp* is the new URI scheme name and *domain* is a Fully Qualified Domain Name (FQDN) relating to where the CI is located. The *user* and *password* arguments are optionally used as a means of authorization. The *path* adheres to the context information namespace hierarchy, permitting the organization and sorting of the items in a logical sense while *options* facilitates further modifiers in the form of *parameter=value* pairs.

An example of a fully qualified UCI adhering to this would be:

dcxp://andeen.mccarthy@miun.se/weather/temp?unit=celsius

Resources are registered with the Context Storage component residing in the overlay.

Context Storage - CS. Previous implementations, enabled a Context Storage mechanism residing on top of the overlay. With this approach, enabling more useful searches such as range queries involved additional layers of complexity as discussed in Section 1. In order to exploit these native characteristics of the overlay, the CS is now built into the overlay. The key role of the CS remains that of resolving UCIs to physical end point addresses of the responsible nodes, behaving similarly to a dynamic DNS service. Here the substring and range queries enable the locating of resources without needing to know the fully qualified UCI.

Additionally, the CS stores the current value associated with a resource. Where a resource consists of multiple dimensions such as GPS coordinates,

each attribute dimension is stored separately. We separate dimensions to further enable independent queries over any value constituting a context information source. An end point is then free to reconstitute and compare the entire n-dimensional value or any valid subset. We further benefit from the overlay's order preservation and range query properties, permitting the acquisition of useful data in support of context metrics or similarity functions.

2.3 Persistence

The CS along with the DCXP protocol enables distributed access to only current context information. Any new values reported in connection to a UCI or context dimension supersedes the current value. This prevents applications from using expired or stale data. Historical information was previously persisted to a centralized relational database by the nodes. The dynamic nature of context information mandates the storage of historical information both on values and the actual context objects. However storing versions of information as different entries on the overlay, would undermine its performance as a direct result of the vast number of attributes persisted at each node; this would be increased by a factor equal to the number of versions.

We therefore solve this problem by using an additional layer; the persistence layer. This consists of a collection of localized Object Oriented Databases residing at each CUA. Each node stores the set of objects that are created locally by the application and services co-located with the CUA. They further store a collection of objects that are being used by these applications and services but originate at a remote node within the overlay. We also persist all observed values for each context dimension permitting temporal views of data evolution, trend and pattern discovery.

Before an object is persisted locally, an attempt is made to persist it on the overlay. If this is successful, then the object being stored is guaranteed to be globally unique and is then stored locally with its UCI on the overlay. If this is not successful, the UCI is being stored already exists and the persistence operation fails. This ensures that locally persisted objects are always globally unique. All objects that are being used by applications and services local to the CUA are stored locally, these contribute to some local schema (Section 2.5) being used at the node.

2.4 Object Layer

The lower layers of the framework utilize only context attributes in the realization of the required functionalities. These are persisted as primitive values on the CS and made accessible through the publish/subscribe interface. Applications residing above the API, however require context objects with which to realize support for users and services.

The Object Layer resolves this gap by permitting the composition of context attributes into context objects. Such as a the latitude and longitude of a GPS sensor being exposed as a 2D location object attached to a presentity. There is no

requirement for an object to recompose all the underlying values of a sensor or for all the values to originate from the same sensor. Such composition is opened to implementation on the object layer. Application developers further work only with objects without needing to consider the UCI or location of the primitive sensor attributes and values.

An application requiring use of an object, makes a *TRANSFER* request. This retrieves a description of the object and constructs it locally and made available to the application or service. When an object is created or retrieved, the object layer constructs the object from the attributes it describes and realizes a subscription to all the sensors contributing to its composition. An attempt to retrieve a value would result in a request for the most up-to-date value stored in the CI, for both local or remote objects. An attempt to modify a value would be forwarded to the CI in order to have the value updated. It the object does not reside at the CUA attempting to modify its value, this fails as the modification must be done local to the owner of the object.

Additionally, an object initialized on a remote node, realizes a subscription to the object's home node. If the object is modifies, all nodes with an instance of the object are notified and they may update the object as required. This provides for an always up to date copy of a sensor object on the object layer.

Objects relevant to a presentity are grouped together and presented as a schema made available to applications through the API.

2.5 Schema Layer

In support of localized object views, we introduce the concept of a *Context Schema* [23], defined as:

The collection of information points associated with and contributing to a presentity's current context

where an *Information Point* is defined as:

Any source providing information about the context of an entity or any sink capable of accepting an input effecting changes to an entity's context

Within the schema layer, such a schema is attached to a presentity and encapsulates all the information points and the relationships related to a presentity. An application or service with a requirement to deliver some user-context centric experience subscribes to the current schema description; it realizes a collection of information objects underpinned by a *publish/subscribe* interface to the end points described by the schema. As a presentity traverses a connected things infrastructure it discovers new entities and consequently updates its schema to reflect this. As a result, all subscribing end points receive an updated schema and can adjust their services to accommodate this.

This addresses the scenario where a context network being highly dynamic, rapidly evolves, mandating the need for an historical representation of interactions. This permits us to examine the behavior of entities on an *Internet of Things*, deriving metrics representative of the interactions among these entities.

3 Context Metrics

The support of applications on an *Internet of Things*, mandates the provisioning of current, relevant and accurate context information to end-points. Such information must be derived and represented as a local subset of the global context information domain. With this, users and applications residing at endpoints are capable of having access to relevant information within some predictable window, explore and build dynamic context centric relationships in response to changes in state and context. In this chapter we discuss two context metrics with respects to the implementation detailed in Section 2. However, while these metrics may be implemented on any solutions capable of providing the required data, our architecture provides the most optimal support in a distributed environment.

Firstly, we need to identify similarities among entities providing a base for discovering new entities; and secondly, the need to be able to identify important and useful sources of context information, providing entities with the information needed to evaluate the reliability of context information sources as well as the resulting relationships established over this information.

3.1 Context Proximity

One metric we consider desirable in browsing a network of information, is a context proximity metric, a measure of the *distance* between presentities considering all expressions of context as illustrated in Figure 3. With this, we can create dynamic user-based context-centric clusters of information points and presentities that are capable of enabling applications and services to provide user experiences based on current context.

In Figure 3, P_1 while connected to S_2, derives an implicit but existing relation to P_2 via S_1. The implication being that their connection suggests that P_1 and P_2 share, to some extent a similar context. If S_1 and S_2 are expressing the same context indicator type, i.e. they are two information points of the same type such as a temperature sensor, then P_1 shares a context similar to that of P_2 by a function of the difference between S_1 and S_2; their sensor value proximity.

With this assumption, we explore our context architecture for context information sources that lay within X_S1; the context proximity limit of P_1.

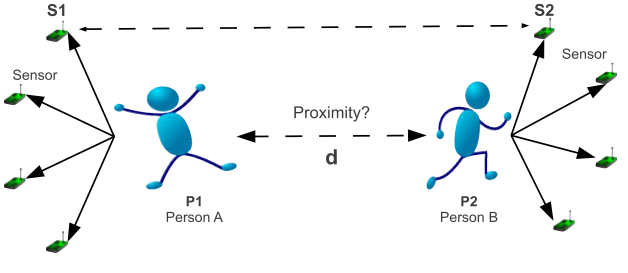

Fig. 3. Context Proximity

We envision that applications will be able to define limits of X_S1 such as: *find all people within 3km with a temperature less than* 5 °C *difference* permitting us to obtain parameters needed to derive the entity sets. Since our solution must remain fully distributed, we located initial nodes by issuing a search using the the range querying function of the underlying P-Grid overlay. This returns a list of entities with respect to the query and constructs a running query at each peer with the following constraints:

1. The peer is responsible for a sensor fitting the criteria of the search
2. The peer is responsible for a sensor S_i with a range such that the set of sensors fulfilling the query from S_1 would be a subset of a query from S_i.

Each peer is then required to:

1. Forward the sensors matching the standing query to S_1
2. Forward the query from S_1 to any node it encounters that matches 1 & 2 above.

This results in a group sensors being returned where for each group G:

$$G = \{S : S \in D : (|V_{S_1} - V_{S_i}| \leq X)\} \tag{1}$$

here, V_{S1} is the current value of S_1 and V_{Si} is the current value of S_i within a domain D.

This is a dynamic set of information points with respect to P_1 its context dimension S_1, that continually evolves to reflect the addition or removal of sensors with respect to their current values as nodes respond to the query.

We consider the fact that not all instances of S_i lie within the same proximity to S_1. This implies that S_1 shares a closer context with some members of G and subsequently those members must be given a higher preference with regards to any context dependent application or services wishing to find context information points in support of delivering some optimal user experience. Noting the varying scales for each sensor, we normalize the euclidean distances with respect to their scales and the distance from S_1 such that:

$$R_{S_i} = f(S_i) = (1 - |V_{S_i} - V_{S_1}| \cdot {X_{S_1}}^{-1}) \ \textbf{\textit{where}} \quad 0 \leq R_{S_i} \leq 1 \tag{2}$$

A value of 0 being at the edge and 1 being identical to X_{S_1}. This value is useful for applying weighting to the edges connecting S_1 to S_i and subsequently the edges connecting to P_1.

Figure 4 illustrates a possible resulting set of such implicit connections with some degree of context similarity owing to the fact that their underlying sensors are within close proximity. By deriving the degree of this closeness, we can obtain a set of presentities within proximity. Consider P_1 and P_2 connected to sets of sensors such that:

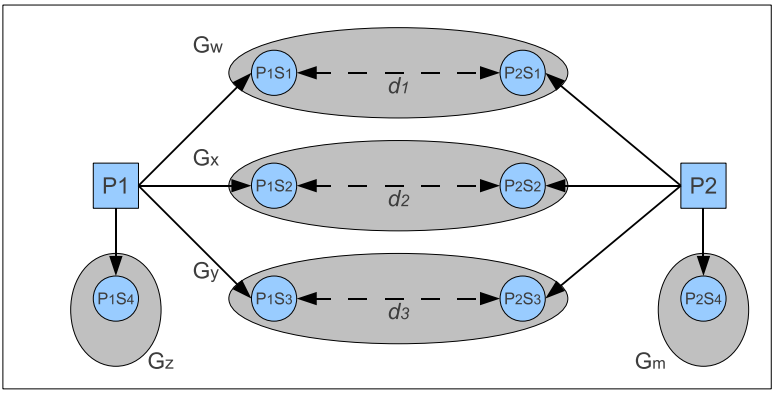

Fig. 4. Determining Presentity Proximity

$$P_1 = \{G_w, G_x, G_y, G_z\} \textbf{ and } P_2 = \{G_w, G_x, G_y, G_m\}$$

Based on this, we determine PS, the presentity similarity as Jaccard similarity of the set of sensors shared by P_1 and P_2:

$$PS(P_1, P_2) = \frac{|P_1 \cap P_2|}{|P_1 \cup P_2|} = \frac{|G_w, G_x, G_y|}{|G_w, G_x, G_y, G_m, G_z|} \tag{3}$$

This permits the comparison of values that cannot easily be measured discretely such as favorite color, mood, etc. In these expressions of context, we are unable to perform discrete distance measurements, however we can provide mechanisms for grouping together similar values which might equate to a user saying: *I like red, but pink, and purple are also acceptable alternatives.* Therefore if a presentity was comprised entirely of non-discrete values, we could still derive a measurement of distance based on the grouping of these values and finding the degree of similarity between the presentities. An end point could define the dimensions of context considered when calculating similarity, such that for an application only interested distance, temperature and humidity, where P_2 had only two dimensions:

$$PS(P_1, P_2) = \frac{|G_w, G_x|}{|G_w, G_x, G_y|} \tag{4}$$

Further to this, we consider the equation in 2 and adjust the value derived in equation 4 to reflect the distance between the underlying expressions of context supporting the presentities. This is adjusted by a factor of the average of the

rank of all the connections between P_1 and P_2. Therefore, we state the distance between two presentities, PR to be:

$$PR = \begin{cases} PS \cdot \dfrac{\sum_{n=0}^{k} R_{Sn}}{k} & \text{, i=0} \\ PS \cdot \dfrac{\sum_{n=0}^{k} R_{Sn} \cdot P_{Sn}}{\sum_{n=0}^{k} P_{Sn}} & \text{, i>0} \end{cases} \tag{5}$$

where i is the number of dimension restrictions, P, indicated by the application or service. When applying such restrictions, all dimensions must be accounted for. Each unaccounted for dimension will be ignored, effectively given a priority of 0. We provide for this as we consider that an application or service will be able to indicate context dimension priorities, eg. *find all persons within a context proximity of 0.7, prioritize by distance, then temperature* or *find all persons within 5km, prioritize by distance then temperature*. The resulting is a more relevant subset of presentities and information sources accumulated in an end point close to a presentity, application or service.

3.2 Sensor Ranking

While traversing an *Internet of Things*, users will be excepted to encounter masses of information sources such as sensors. A supporting solution should be able to provide application developers and users with as much information as possible in order to select the most relevant and recommended sources available.

Our ranking algorithm consists of two main components, illustrated in Figure 5. Firstly we need to determine the local ranking value for s with respects to P_i. We then need to aggregate the global ranking value for s.

Localized Ranking. In approaching this problem, we adapt a modified version of the *Inverse Document Frequency* algorithm [24]. This is shown in Equation (6) with a sensor s, and a presentity P. SR is the sensor ranking of the sensor s, R is the corpus, the total collection of schemata relative to presentity P with r being all the schemata relevant to P containing a reference to sensor s.

$$SR_s{}^{(P)} = log \frac{|R|}{\{r : s \in r\}} \tag{6}$$

This provides us with a representative metric as to the importance of sensor s relative to P. We consider further, that there exists scenarios where some presentities will be less dynamic or mobile with respect to s. Such an example might be a sensor located in a store; the employees working in the store will by default almost always utilize the sensors that are local to the store accounting for a disproportionately higher value for SR: In such scenarios, taking:

$$log \frac{|R|}{\{r : s_i \in r\}} \tag{7}$$

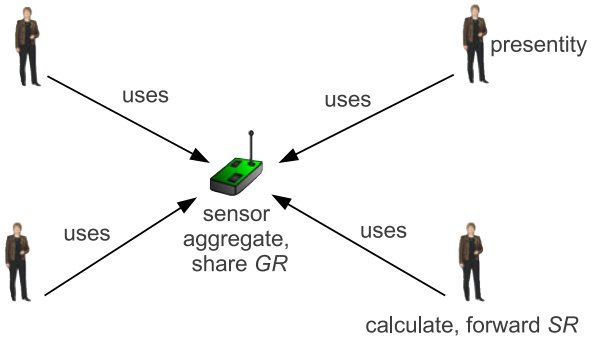

Fig. 5. Determining Sensor Ranking

considers more dynamic presentities traversing an *Internet of Things*. Such as a person that travels around the city interacting with more sensors and subsequently creating more context schemata in fulfillment of service delivery. This is represented by larger ratio of R to $\{r : s_i \in r\}$. Such presentities we argue, indicate a more accurate ranking for s, relative to the wider sensor ecosystem.

We could also calculate a ranking for sensors with respect to some time duration of interest, t, by limiting the schemata used in calculation to those created within t.

Global Aggregation. The second component of our approach is a global aggregation of all the local ranking values SR assigned to s. To achieve this, we calculate the Global Ranking GR by finding the sum of all SR of s such that:

$$GR_s = \sum_{k=0}^{n} SR_k^{(P)} \tag{8}$$

This value is continually calculated as new schemata referencing s are created. We further take into consideration the owner of s, the presentity or domain where it resides or to which it belongs. This we regard as the domain D and assign it a value equal to the average ranking of all the sensors belonging to D. This we call DR, the ranking of D. This value is important to us as it would permit us to identify more connected and important spaces such as domains, buildings or just a collection of deployed sensors. We calculate DR as:

$$DR = \frac{\sum_{k=0}^{n} SR_k^{(P)}}{k} \tag{9}$$

This ranking value can be made available on the overlay, supporting range queries across ranking values and readily accessible. No centralization is required for ranking to occur and new values are updated in real time. The resulting values can be used as indicators of relevancy or importance of sensors on an *Internet of Things*.

4 Conclusion

Users, applications and services on an *Internet of Things* demand sensor information in support of realising context-centric user experiences. Such information must be dynamically organized and provisioned as accurate and relevant subsets of global information in end-points. Additionally, such provisioning must be liberated from the assumption network services enabling the brokering of information or presence models that do no scale well.

In response to this, we presented further work on the MediaSense framework, detailing its re-implementation towards a more robust support for the dynamic properties inherent in provisioning context information among heterogeneous end-points. As the new overlay of choice, P-Grid realizes a more unstructured behavior over DHT solutions. It achieves this by implementing a non-deterministic key-space organized as a distributed binary tree. This, while preserving the ordering of values and allowing us the ability to perform more complex range queries than DHT based solutions.

The ability to perform such queries along with the self-organization behavior of the overlay permits us implement algorithms for deriving metrics ranking entities both personally and globally. Unlike cloud solutions such as Google [17] which are updated at regular intervals, this permit the ranking of resources as they become available. As nodes are added, they respond to any relevant standing queries created by the existing querying nodes. A node recalculates the proximity and ranking of sensors from this continually updated and relevant subset. A We further exploit this overlay to persist ranking values for entities providing a global access to an entity's reputation.

Future work on towards this includes deriving a large sample set capable of generating values for testing and bench marking the performance of the solution. The creation of a simulator for our solution is still required in order to verify scalablity and performance within a large scale deployment. The solution further requires the implementation of mobile nodes enabling performance measurements reflective of its usage in a real world scenario. This would include creating applications on mobile phones or computers. Other work includes the ability to conduct more knowledge discovery by exploiting the properties of the overlay and further work on extending and improving both algorithms.

References

1. Sundmaeker, H., Guillemin, P., Friess, P., Woelfflé, S.: Vision and Challenges for Realising the Internet of Things. In: Cluster of European Research Projects on the Internet of Things (CERP-IoT) (March 2010)
2. Lee, J., Song, J., Kim, H., Choi, J., Yun, M.: A User-Centered Approach for Ubiquitous Service Evaluation: An Evaluation Metrics Focused on Human-System Interaction Capability. In: Lee, S., Choo, H., Ha, S., Shin, I.C. (eds.) APCHI 2008. LNCS, vol. 5068, pp. 21–29. Springer, Heidelberg (2008)
3. Gonzalo, C.: 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular World, p. 381 (2005)

4. Kansal, A., Nath, S., Liu, J., Zhao, F.: SenseWeb: An Infrastructure for Shared Sensing. IEEE Multimedia 14(4), 8–13 (2007)
5. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, vol. 31, pp. 149–160. ACM (2001)
6. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. Design (2001)
7. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications 22(1), 41–53 (2004)
8. Pappas, V., Massey, D., Terzis, A.: A comparative study of the DNS design with DHT-based alternatives. The Proceedings of IEEE, 1–13 (April 2006)
9. Kanter, T., Pettersson, S., Forsstrom, S., Kardeby, V., Norling, R., Walters, J., Osterberg, P.: Distributed context support for ubiquitous mobile awareness services. In: 2009 Fourth International Conference on Communications and Networking in China, pp. 1–5. IEEE (August 2009)
10. Baloch, R.A., Crespi, N.: Addressing context dependency using profile context in overlay networks. In: 2010 7th IEEE Consumer Communications and Networking Conference (CCNC), pp. 1–5. IEEE (January 2010)
11. Ratnasamy, S., Hellerstein, J.M., Shenker, S.: Range Queries over DHTs. IRB-TR-03-009 Intel Corporation (2003)
12. Ding, G., Bhargava, B.: Peer-to-peer file-sharing over mobile ad hoc networks. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp. 104–108. IEEE (2004)
13. Kanter, T.G.: Going wireless, enabling an adaptive and extensible environment. Mobile Networks and Applications 8(1), 37 (2003)
14. Stonebraker, M., Aoki, P.M., Litwin, W., Pfeffer, A., Sah, A., Sidell, J., Staelin, C., Yu, A.: Mariposa: a wide-area distributed database system. The VLDB Journal The International Journal on Very Large Data Bases 5(1), 48–63 (1996)
15. Barbara, D.: Mobile computing and databases-a survey. IEEE Transactions on Knowledge and Data Engineering 11(1), 108–117 (1999)
16. Ulusoy, O.: Transaction processing in distributed active real-time database systems. Journal of Systems and Software 42(3), 247–262 (1998)
17. Google (2010), `http://www.google.com`
18. Zhu, Y., Ye, S., Li, X.: Distributed PageRank computation based on iterative aggregation-disaggregation methods. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 578–585. ACM, New York (2005)
19. Li, J., Loo, B., Hellerstein, J., Kaashoek, M., Karger, D., Morris, R.: On the Feasibility of Peer-to-Peer Web Indexing and Search. In: Kaashoek, M.F., Stoica, I. (eds.) IPTPS 2003. LNCS, vol. 2735, pp. 207–215. Springer, Heidelberg (2003)
20. Sankaralingam, K., Sethumadhavan, S., Browne, J.C.: Distributed pagerank for p2p systems. In: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, pp. 58–68. IEEE (2003)
21. Plaice, J., Kropf, P.G., Schulthess, P., Slonim, J.: DCW 2002. LNCS, vol. 2468, pp. 345–392. Springer, Heidelberg (2002)

22. Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R.: P-Grid. ACM SIGMOD Record 32(3), 29 (2003)
23. Walters, J., Kanter, T., Norling, R.: Distributed Context Models in Support of Ubiquitous Mobile Awareness Services. In: Par, G., Morrow, P. (eds.) S-CUBE 2010. LNICST, vol. 57, pp. 121–134. Springer, Heidelberg (2011)
24. Robertson, S.: Understanding inverse document frequency: on theoretical arguments for IDF. Journal of Documentation 60(5), 503–520 (2004)