Searching in Internet of Things using VCS

Mitul Shah
Dept. of Electronics and Computer Engineering
Indian Institute of Technology Roorkee
Roorkee, India
+91 – 7409234037
mitul_45@yahoo.co.in

Anjali Sardana
Dept. of Electronics and Computer Engineering
Indian Institute of Technology Rookee
Roorkee, India
+91-1332-285655
anjlsfec@iitr.ernet.in

ABSTRACT

Due to proliferation of WSN applications in real world, we are having smart devices everywhere. Our future will be full of these sensor objects. Internet of Things (IoT) is going to be future of today's Internet. Searching in such environment is necessary and challenging as user wants to operate an object locally or remotely. However, unlike web which is built of static documents, sensor reading possesses very short life span and real world entities are highly dynamic also. This implies traditional search techniques of web do not work in IoT. In this paper we present a novel idea of using Virtual Coordinate System (VCS) for searching in IoT. VCS approach has been used in Peer-to-Peer networks. The proposed mechanism is energy efficient and uses multiple features like memory, computation cost, etc. to find the optimal result as per user's requirement.

Keywords

Searching, Internet of Things (IoT), Virtual Coordinate Systems, Wireless Sensor Networks

1. INTRODUCTION

Today Internet connects thousands of people across the globe every day. Primarily Internet of Things is aimed at developing our daily life more sophisticated and flexible. IoT is going to be a network of *smarter objects* which includes people. These devices will have communication and computation capabilities [1]. Each object consists of a sensor of a particular type which will collect the data from its environment and it will either pass that data or do some processing on them. Some of the interesting applications of IoT are checking the water level of your plants remotely (www.botanicalls.com) or WEMO (www.belkin.com/wemo) is another product which allows us to control electronics appliances using a smart phone application. We will have hundreds of billions of RFID-tagged objects by 2015[2]. We can say future is not as far as we might think.

In case of IoT the search space is much larger than web because we have to search through sensor reading and other automated results while web consists of only user generated data. As an example user might have misplaced his key in the house. Now if he wants to search for it then some device discovery mechanism is needed. Searching in physical world is more difficult then searching web because of high mobility of nodes. One option is to equip each node with a GPS sensor, however this wireless sensors have low power supply and also they must be cheap in order to be used widely so this is not a feasible solution. We cannot even use the existing techniques like crawling and indexing as they inherently assumes that data updates are not frequent and data is static.

As of now there are some implementations for searching in real world, e.g. Snoogle [3], MAX [4], etc, but all of this have some drawbacks. Like Snoogle uses centralized approach which makes system not easily scalable. This leads us to design distributed system for searching process. And in case of MAX the queries are costly as it has to be sent to each tag.

VCS is used mainly in P2P architecture for finding the nearest node to get any service from peer node[5]. We suggest the use of Virtual Coordinate System for finding delays, latencies and services of the sensors in real world. Unlike earlier approach our approach is decentralized and takes various other parameters in to the account.

The rest of this paper is as follow Section 2 provides some information about the recent work in this area. Section 3 consists of our proposed approach. In section 4 we provide analytical analysis of proposed approach and finally we conclude the paper in Section 5.

2. RECENT WORK

In this section we present recent work done in this field.

2.1 Snoogle

Snoogle is an information retrieval system built on sensor networks for the physical world [3]. Each real-world object is associated with sensor node which carries the textual description of that object in form of keywords. Snoogle consists of three layers of components, object sensors, index points (IP), and a Key Index Point (KeyIP), which is a super node that manages all IPs. Overall architecture of this system is shown in Figure 1.

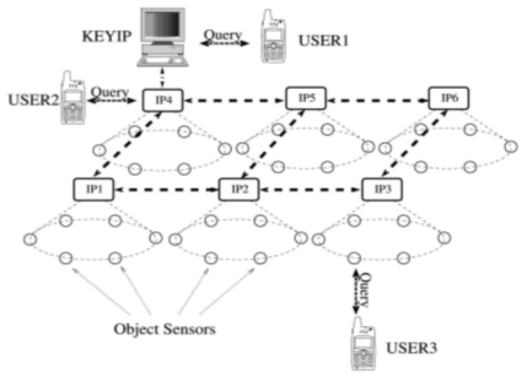

Figure 1. Snoogle – Architecture

Depending upon the mobility of real world object, the *object sensor* can be either static or mobile. Snoogle requires all object sensors to communicate using the same radio frequency.

An *Index Point (IP)* is a static sensor device that is associated with a physical location, like a cabin in an office building. These IPs are responsible for collecting and maintaining the data from the object sensors in their range. IPs are battery powered, they have microcontroller, radio module and large amount of flash memory.

The *Key Index Point (KeyIP)* does the job of collecting data from different IPs in the network. The KeyIP is assumed to have access to a constant power source, powerful processing capacity, and possess considerable storage.

In Snoogle user can query using some smart portable device such as Smartphone or a PDA. End user provides some keywords he/she is looking for. Based on these keywords Snoogle searches appropriate object sensors matching that keywords. User can perform two different kinds of query, local or distributed. A local query is to perform search under the region of some particular IP, when user knows the approximate location of physical object to be searched. Otherwise in distributed query an exhaustive search will be applied in which search will start form KeyIP. Snoogle also applies ranking algorithm and presents only top-k results to user.

The key limitations of this approach are as follows [6]. Firstly this supports searching for static data only. For each change in data of an object sensor the database at IP and KeyIP must be updated immediately. This questions the scalability of the approach. In a way this approach is centralized to KeyIP and in case of global search high number of small messages will be transmitted. Last limitation is due to use of bloom filters as results generated are approximate.

2.2 MAX

MAX [4] is the human-centric search engine for finding real-world objects. It provides location that is natural to humans rather than actual coordinates. This system inherently assumes that all physical objects like cloths, documents, keys are tagged with small devices which have limited processing and communication capabilities. For privacy concerns tags can be made public or private, where public tags are searchable by everyone and private tags can be searched by owner only. MAX uses hierarchical structure where objects in each level are more mobile than higher level. System architecture has been shown in Figure 2.

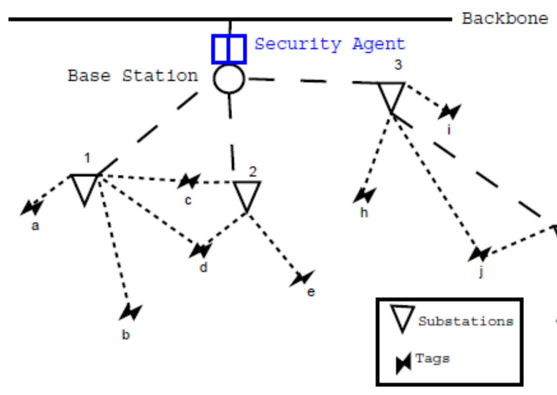

Figure 2. MAX – Architecture

Base-Station: The base-stations are at the highest level of the hierarchy, representing immovable localities such as room. It also stores descriptor of this locality (e.g. Guest Room). There could be one or more base-stations per locality if size of room is large enough. These base-stations work as mediator between wired backbone network and wireless tags. The base-station will have significant memory and processing capabilities.

Sub-station: Sub-stations are at the next level of the hierarchy and describes objects which are static and change positions occasionally (e.g., Study table). Sub-stations are powered by battery and have limited processing and communication capabilities.

Tags: Tags are at the lowest level of the hierarchy and attached to the objects which are mobile (e.g., cloths). It stores a descriptor of the object it is attached to. These tags have to extremely small and cheap. They either have limited or no constant power supply. The tags must be labeled sensibly is user wants the object to be found in search.

Security Agent: These are software agents who reside at the base-stations, as illustrated in Figure 2. It authenticates the user before they are allowed to query in the locality covered by the base station.

MAX uses pull based mechanism for query resolution. User can specify the base-station to query along with the keywords to be sought. Here once receiving the query base-station will broadcast the query to all its substations and sub-stations will further broadcast it. Finally tags will count how many keywords in the query match to its own textual description. This count will be pulled by sub-station and then it will forward it to base-station. And then base-station will return the descriptions of top *k* tags to MAX server and user.

The pull based approach is better in terms of supporting mobility but main drawback [6] is that queries are very costly. In MAX each query has to be sent to every single sub-station and tag. This makes MAX inappropriate for global networks.

2.3 OCH – Object Calling Home

With this system [7] user can query the current location of lost real-world objects. This system tags objects using identity unlike Snoogle and MAX in which keywords are stored. In a prototype, authors have used Bluetooth enabled mobile phones as object sensors and objects are tagged with small Bluetooth modules. The system provides approximate location of the lost object as a result. User can also include the time t and budget q with the query. A continuous query will end after t seconds and q indicates the total number of messages that the system may send for searching.

For scalability considerations queries are not sent to all object sensors. This system uses probabilistic measures instead of broadcasting. Each location has a given probability. Probabilities are assigned based on various heuristics like object is probably at a location that is near to location where it was seen last or the object might be at a location that is recently being visited by owner of that object or it might be at the work place of owner where he spends most of his time. End user can also associate priority with any of the above heuristics to be used.

In total query contains identity of the required object, the above priorities, timeout t, and budget q. Next query is sent to central mediator which creates a ranked list of locations with high probability. No object sensors have been communicated till now. Now central mediator contacts object sensors in order of ranked locations. Next mediator subscribes them in top to bottom order and stops when one of the sensor reports the required item or system reaches the budget limit specified or timeout occurs.

One of the advantages of the above approach is that it can be applied to very large networks as not all object sensors are queried and typically only a small proportion of all object sensors are contacted to resolve a query. However, if the budget limit is not set to infinite this approach does not guarantee that object will be found if it is actually detected by sensor. Furthermore, computing the models involves an overhead. Finally, this system supports and takes advantage of having mobile sensors, as mobility increases its coverage and hence the probability of finding a sought object.

Now we can compare this three existing approaches in Table 1. using some measures defined in [8].

Table 1. Comparisons of existing work

Dimensions	**Snoogle**	**MAX**	**OCH**
Aggregation type	Hybrid	Pull	Push
Query type	Ad hoc	Ad hoc	Continuous
Query mode	Keyword	Keyword	Keyword
Query scope	Local	Local	Global
Query time	Real-time	Real-time	Real-time
Query accuracy	Heuristic	Heuristic	Heuristic
Query content	Static	Static	Dynamic
Mobility	Yes	No	Yes

There are mainly four challenges [8] present for searching process in Internet of Things which are

Architecture design of search engines The goal of searching physical objects distinguishes that of commercial search engines is that users are willing to operate objects locally or remotely. Thus new techniques are required for crawling, indexing, storing and querying.

Search in locality Unlike the users of web services who requires get information from a distant server, in IoT in most cases user manually wants to operate or control the physical objects.

Scalability In case on IoT the network is saturated with huge number of sensors. These sensors and RFID keep capturing objects and generating extremely huge number of readings. Moreover these readings have usually limited timeliness. For example reading of a passive RFID tag will be expired in 2 seconds.

Real-time one of the most important function is real-time. As sensor readings can expire in small time such as 2 seconds.

According to another view [9] some challenges in Pervasive environments are as follows.

The future of internet will be made of highly dynamic and heterogeneous than current enterprise networks. This enterprise network services are operated within networks, protected by firewalls and managed by system administrators. Service discovery also differs in terms of locality. Like web services have no physical location limitations, whereas in IoT the physical location does matter a lot. The other main challenge is of privacy and security. As Ross Anderson pointed out that many security solutions will fail when applied to different environments [10]. Unlike the enterprise environment, where physical boundaries and firewalls can separate insiders and outsiders, users and service providers can coexist in a pervasive computing environment. Moreover the data content is also more important in case of IoT or Pervasive environments as it might actually have user's current location or his health status which are extremely private.

3. PROPOSED TECHNIQUE

In this section we propose a novel approach for service discovery problems in Internet of Things. Instead of using this centralized approach or IP based approach like Snoogle, we are using Virtual Coordinate System for discovery process. This VCS approach has been used in wired P2P networks for service discovery to find nearest node providing service [5]. We explore the light weighted VCS scheme for service discovery in IoT, where sensor networks are used. In our approach assign each node with several attributes like memory, energy, computation power, etc which are used in service discovery process. After having this features we do some calculation and present the list of available options ranked according to user's preference.

3.1 Parameters

The various parameters are defined as follow.

3.1.1 Shortest distance

This feature is used in the case user requires to find the physically nearest possible service provider in Internet of Things.

3.1.2 Energy level of a node

Because in case of wireless networks each node is equipped with small power supply energy level is an important parameter. Thus in the case when there are two possible service providers having same distance, memory, communication and computation cost prefer a node having higher energy level.

3.1.3 Communication cost

Consider a case when the service provider has sufficient memory, good energy level, and higher transmission capacity but is at far distance from the one which requires service then communication cost will be high and that will result in degradation of overall performance. Thus considering communication cost in discovery process improves overall performance.

3.1.4 Computation cost

Computation cost is calculated because there can be some service providers which have higher resources compared to others such that they can complete a complex task in lesser amount of time. So in presence of such node priority is given to such nodes to complete a task which requires higher computation.

3.1.5 Memory

As power supply, in general memory available to each node is also limited and so it is considered in the process.

3.1.6 Nodes status

This feature is used to indicate the current status of the servers so that load balancing approaches can be applied. In a case when one node A is serving 2-3 clients and other node B is currently free then we should choose B node as optimal even though its computational power, memory and other resources might be less than of A.

3.1.7 Bandwidth

As we know the client node needs to have communication with service providing node. In such case if connection between (i.e. transmission capacities) client – server is weak then communication delay will be higher.

3.1.8 Latency

Latency between two communicating node should be as low as possible because each task in Internet of Things will be performed in real time.

3.2 System Architecture

Proposed system architecture is shown in Figure 3. Here each node has seven different parameters and link between the nodes have two parameters (e.g. latency, bandwidth).

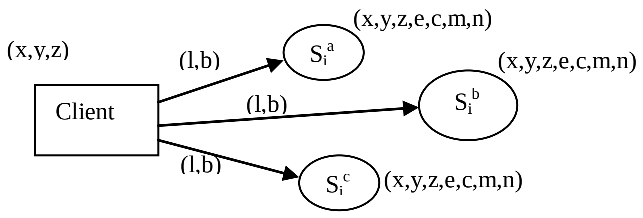

Figure 3. Proposed Architecture

The different parameters are defined as follow

x,y,z – the virtual location coordinates

e – energy level of a node

c – computation power of a node

m – available memory of a node

n – number of nodes being serviced by a server node

Each link between the nodes have two parameters

l – latency of a connection

b – transmission capacity of communicating nodes

In Virtual Coordinate System each node is assigned with a coordinate which represents it physical location relative to other nodes. It might not be real coordinates but in sensor networks virtual coordinates are enough for routing purpose. Now instead of focusing on only coordinates of the system we can also take the seven features listed above in to the account for finding optimal service provider. We are considering the n-dimension plane which consists of this seven feature as seven axes.

Unlike other centralized approaches, our proposed approach does not have any mediator. The client node acts as SRM (Service Request Manager) it contacts each node in its region. We use pull based mechanism such that whenever request arrives the respected parameters are calculated and decision is made.

User can also assign the weights to these measures like if user wants a service that requires higher computation then weightage to that particular measure (computation cost) will be high and ranking will be done accordingly. In case when user has not specified any parameters each attribute will be given equal priority.

Consider a case when a particular node wants some kind of service. The service providers are like servers which provide the services to SRMs. To maintain real time data about mobile nodes in the network two approaches exists [8].

First one is *push based* in which each servers keeps on periodically pushing list of services it provides. This solution guarantees up-to-date data but unnecessary updates are problem and so is not energy efficient solution.

The other approach is *pull based* in which each SRM keeps polling the probable servers in the region to know their current status. SRM can also keep up-to-date data about latency, bandwidth and other features in order to provide service to users as early as possible. This solution is more energy efficient than the previous one.

3.3 Calculations

The typical scenario for service discovery is shown in Figure 4. in which four node provides a service and SRM node is contacting each node and does the calculation.

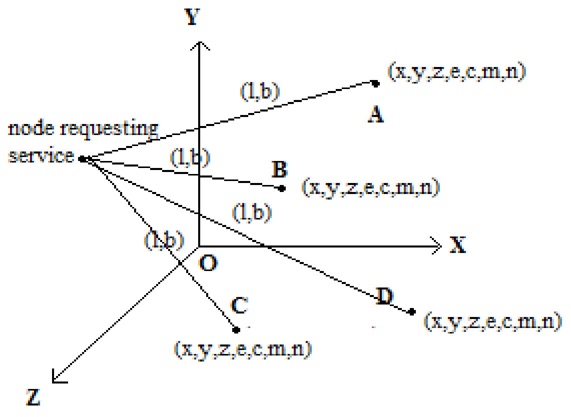

Figure 4. A Typical Scenario

Suppose there are three types of services S_i, S_j, S_k and client requires service of type S_i. We assume that client uses some king of broadcasting mechanism so it will first discover the nodes which provides service S_i.

Next SRM node calculates different attributes based on parameter assigned to server nodes and links. Here as shown in Figure 4 there are four possible service providers (A,B,C,D) and for each of them following parameters will be calculated and then decision will be made.

Least distance: This parameter is calculated using *x,y,z* attributes of two nodes this can be simple Euclidian distance.

Energy level: This parameter is specified by the server node. In a case when two nodes provides equal computation power and all other cost we should consider the node having higher energy level. This makes the node having less energy survive for some more time.

Communication cost: This parameter is calculated based on distance and the transmission capacities. Communication cost can be calculated in terms of messages passed per unit time or time required to send some fixed number of bytes to server node.

Computation cost: In our system each node is equipped with computation power of its own. Thus once getting the computation power of a node, SRM calculates approximate time it will take to complete user's task on that particular server node.

Memory: This parameter is provided by the server node. If a server node fulfills our memory requirement, then only this node is taken into consideration.

Node Status: This parameter is provided by server node. It indicates weather the server node is currently servicing any other nodes or not. Free nodes are given higher priority as compared to the nodes which are providing service to some other node.

Bandwidth: Bandwidth is parameter of communication link unlike others parameter it isn't specific to node. In wired networks bandwidth of a connection is static but in case of wireless networks, as nodes move around, some interference might affect the communication rate of a connection. This bandwidth between the nodes can be calculated periodically using some *hello packets* or it can be calculated on-demand.

Latency: Latency is also a parameter of a link between the nodes. For calculating latency of the link same approach is applied which is used to calculate bandwidth.

Having these all parameters we can calculate the mean (or weighted mean) or simple Euclidean distance can be used to decide the optimal service provider from different available options. In case if user has given priority to particular parameter like if he wants service by nearest node then *shortest distance* will have the highest weight among all others.

Our approach uses Euclidian distance for computing shortest distance (i.e. $(x^2+y^2+z^2)^{1/2}$). This calculation makes this algorithm lightweight and requires less computation power at client node. And all these features are calculated in real time.

In this way we can rank different service providers according to its capabilities like computation power, available memory, energy level and different network parameters to get the most optimal service provider.

4. ANALYSIS OF PROPOSED TECHNIQUE

This section provides analytical results to our proposed solution for discovery process. Table 2. includes one more column of our proposed technique to Table 1. And comparison is done using parameters defined in [8].

Table 2. Comparison

Dimensions	**Snoogle**	**MAX**	**OCH**	**Our Proposal**
Aggregation type	Hybrid	Pull	Push	Pull
Query type	Ad hoc	Ad hoc	Continuous	Continuous
Query mode	Keyword	Keyword	Keyword	Keyword
Query scope	Local	Local	Global	Local
Query time	Real-time	Real-time	Real-time	Real-time
Query accuracy	Heuristic	Heuristic	Heuristic	Exact
Query content	Static	Static	Dynamic	Dynamic
Mobility	Yes	No	Yes	Yes

As we can infer from the table, our proposed technique uses pull based approach to maintain real time data for energy reasons. There are two types of queries possible the ad hoc mode indicates one-time queries where result is returned immediately. While continuous types of queries are active for period of time and matches are returned while the query is active. Query mode is the manner that search engine supports, involving, keyword-based, and semantic-based and hybrid. Query scope can be local or global. Our proposed approach works on local networks only as there is no central authority. As searching is performed in smaller networks, we can perform search in a global network which is indeed network of smaller networks. Query time indicates the freshness of result, it can be real-time or historic. Query accuracy defines the closeness of the result with the real results. Query content defines the content user can provide while querying it can be static or dynamic. And last parameter mobility defines weather system supports mobile nodes or not.

The accuracy of the result is exact as we are calculating different parameter values considering attributes of server nodes. The query content is dynamic as user can specify his/her preferences. And finally our proposed approach does satisfy the mobility requirement as it uses mainly pull based approach for finding available servers.

5. CONCLUSIONS

In this paper we propose Virtual Coordinate System based searching technique for Internet of Things which considers more features like available memory, computation cost, energy level and other features to find the optimal service provider.

The future work mainly consists of implementing prototype of this technique and analyzing time required for total communication done during the whole process.

6. REFERENCE

1. Agrawal, S. and M.L. Das. *Internet of Things—A paradigm shift of future Internet applications*, Engineering (NUiCONE), Nirma University International Conferenceon, 2011.: IEEE : p. 1-7.
2. Das, R. and P. Harrop, *RFID Forecasts, Players and Opportunities 2011-2021, IDTechEx Inc. Report.* 2010.
3. Wang, H., C.C. Tan, and Q. Li, *Snoogle: A search engine for pervasive environments.* Parallel and Distributed Systems, IEEE Transactions on, 2010. **21**(8): p. 1188-1202.
4. Yap, K.K., V. Srinivasan, and M. Motani. *Max: human-centric search of the physical world.* SenSys '05 Proceedings of the 3rd international conference on Embedded networked sensor systems, ACM, 2005:p. 166-179.
5. Dabek, F., et al. *Vivaldi: A decentralized network coordinate system*. SIGCOMM '04 Proceedings of the 2004 conference on Applications, technologies,

architectures, and protocols for computer communications, ACM, 2004: p. 15-26.

6. Romer, K., et al., *Real-time search for real-world entities: A survey*. Proceedings of the IEEE, 2010. **98**(11): p. 1887-1902.

7. Frank, C., et al., *The sensor internet at work: Locating everyday items using mobile phones*. Pervasive and Mobile Computing, 2008. **4**(3): p. 421-447.

8. Zhang, D., L.T. Yang, and H. Huang. *Searching in Internet of Things: Vision and Challenges*., Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on, 2011:p. 201-206.

9. Zhu, F., M.W. Mutka, and L.M. Ni, *Service discovery in pervasive computing environments*. Pervasive Computing, IEEE, 2005. **4**(4): p. 81-90.

10. Anderson, R.J., *Security Engineering: A guide to building dependable distributed systems*. 2010: Wiley.