Elastic Streaming Semantic Engine for M2M Appliations

Xi Chen
Department of Computer Science
Zhejiang University
Hangzhou, China
xichen@zju.edu.cn

Huajun Chen
Department of Computer Science
Zhejiang University
Hangzhou, China
huajunsir@zju.edu.cn

Jue Huang
Department of Computer Science
Zhejiang University
Hangzhou, China
juehuang@zju.edu.cn

***Abstract*—With the rapid advances in Machine-to-Machine sensor data collection and communication, M2M applications become ubiquitous in various aspects of our daily life. Due to the highly distributed and heterogeneous characteristics of M2M data, it requires more and deeper support of semantic technologies. While most of the existing efforts are focused on the modeling, annotation, and representation of semantic sensor networks , there has been little work focusing on the background processing of large-scale real-time sensor data for M2M applications. In the paper, we present an elastic streaming semantic processing framework for M2M applications. The proposed framework efficiently capture and model different scenarios for all kinds of M2M applications. We have preliminarily implemented an engine based on popular large-scale distributed computing platform SPARK. Based on the engine, a typical use case on home environment monitoring is given to illustrate the efficiency of our engine. The results show that our system can scale for large number of sensor streams with different types of M2M applications.**

***Keywords*-M2M, Semantic Web, Streaming, Big Data, Spark.**

I. Introduction

Machine-to-Machine (M2M) applications are more and more popular due to the availability of smart M2M devices (sensors). M2M devices are used in a great deal of realms such as home monitoring, vehicular networks environmental monitoring (weather forecasting), health monitoring. Integrating and processing M2M data from heterogeneous domains is still faced with many challenges.

First, With the advances in M2M domain, more and more smart sensor devices are deployed in the M2M network.It is predicted that within the next decade billions of devices (Cisco predicts that the number of the Internet connected devices will be around 50 Billion by 2020) will generate myriad of real world data for many applications and services in a variety of areas such as smart grids, smart homes, e-health, automotive, transport, logistics and environmental monitoring. Such a stunning number of devices will generate massive data. Secondly, many M2M applications need to gather and combine multiple heterogeneous sensor data with different formats and measurements. Besides that, some background data also needs to be integrated. Thirdly, most of the M2M data is generated with spatial and temporal annotations, which is used to support all kinds of real-time and location-based applications. For example,

The Semantic Web technologies are viewed as a key for the development of M2M. Figure 1 shows the generic functional model of M2M for supporting semantics in the specification of oneM2M study on abstraction and semantics enablement. In specific, it serves as the following several purposes: First, the data access layer provides connections with a device and a gateway for accessing M2M data. Secondly, the abstraction and semantics layer provide us with a good way to resolve the problems of inter-operability and integration within this heterogeneous world of M2M devices by defining and reusing some standard semantic concepts. Besides, the Semantic Web provides a seamless interface to facilitate the interactions of M2M data and other existing non-M2M knowledge or services such as Linked Data, DBpedia, LinkedGeodata, various kinds of Web Services . At last, the service layer provides an interface for various M2M applications by semantic processing technologies, including semantic query, mash-up and so on.

Currently, much efforts have been made to model the M2M sensor networks by . For example, ontologies such as the W3Cs SSN ontology (Lefort et al., 2011; Compton et al., 2012) have been developed, which offers a number of constructs to formally describe not only the sensor resources but also the sensor observation and measurement data.

However, considering the another two characteristics of IOT data: dynamics and scale, current Semantic Web still exists many limitations:

II. RELATED WORK

Semantic Web is considered as one of the most fundamental and important technology in the M2M sensor data management and processing. Lots of efforts have been made by the academia and industry for the semantic support of M2M.

A. *Semantic Sensor Network and Ontology*

One key research topic in M2M semantic modeling is to represent the data observation and measurement from sensors. For example, Semantic Sensor Web (SSW) is a technology in which sensor data is semantic annotated for

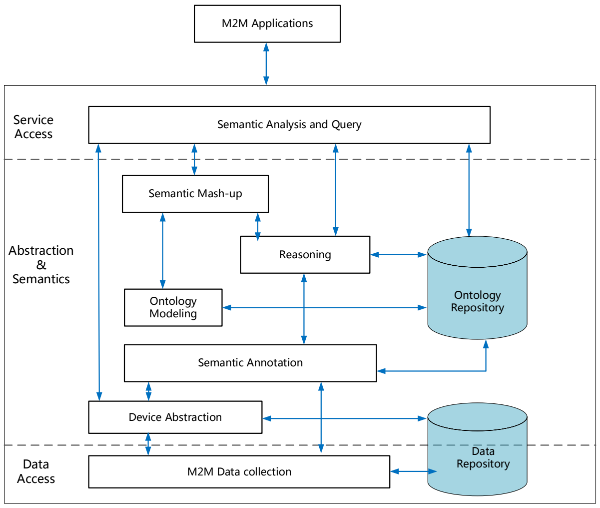

Figure 1. Generic functional model for supporting semantics

inter-operability and also provides contextual information for situational knowledge. Many works have proposed semantic model for representing sensors and data. Ontologies such as the W3C's SSN ontology have been developed. In particular, these ontologies provide metadata for numerical, spatial, temporal, and other semantic objects. Similar works for sensor metadata description also include Sensor Data Ontology (SDO) , Sensei O&M and SensorML. Meanwhile, there are ontologies like CSIRO Sensor Ontology and OntoSensor designed to describe semantic sensor specification and knowledge. In order to improve ontology reusability and inter-operability, some of them extend their ontology based on IEEE's Suggested Upper Merged Ontology (SUMO) as a root definition of general concepts and associations for semantic modeling.

These works mainly focus on semantic annotation for the inter-operability of sensors by defining an unified and standard ontology, paying no attention to the high-level semantic M2M applications.

B. Semantic M2M Applications

Gyrard proposes a semantic-based Machine-to-Machine Measurement approach (M3) to automatically combine, enrich and reason about M2M data to provide promising cross-domain M2M applications, such as naturopathy application based on multiple datasets. The approach also presents a hub for cross-domain ontologies and datasets. [20] and [21] apply the semantic IOT in the generic agriculture and healthcare context management. SSEO [22] is developed to enable semantic indexing, machine-processable event detection and data exchange for smart space modeling. Other applications include CONON [23], CoOL [24] and CoBrA [25].

All the works failed to provide a generic semantic M2M processing framework. And the applications also did not deal with some important challenges for M2M data, such as the real-time and scalability.

C. Semantic Web Technologies

III. Proposed Framework And Elastic Processing Engine

In this section, we propose a semantic processing framework for M2M applications and elaborate the Elastic Processing Engine to explain how it provides the capabilities for performing various M2M applications.

A. Framework

Figure 2 shows the architecture of our semantic processing framework for M2M applications. In general, it consists of five layers: Physical Entities Layer, Abstract Entities Layer, Window-Based Data Stream Layer, Virtual Entities Layer and Elastic Semantic Engine Layer.

Physical Entities Layer Physical entities layer is located in the lowest layer of the framework, which is responsible for generating M2M data in real-time. Every physical entity represents a tangible element that can be sensed by sensors that are deployed in the oneM2M Field Domain environment, and that is not specific to a particular M2M application in this environment. According to the oneM2M project standardization, every kind of sensors are be organized by logical entity (AE) and common services entity (CSE), which provide application logic and common services, respectively.

Abstract Entities Layer

Abstract Entities Layer is responsible for receiving and implementing the abstraction for the data from the physical devices by the semantic annotation of proxy software. The abstraction layer aims at hiding the complexity of variety of devices and environments by providing a single and standard format to represent devices. So from the view of upper layer, all the data can be .

Window-Based Data Stream Layer

Virtual Entities Layer Virtual Entities Layer aggregates the related data for every virtual entity. Virtual entity is a new resource created by multiple window data streams. For our latter use case, if user in a home requests the service for Discomfort Index(DI), a new virtual entity will be generated through aggregating corresponding home appliance sensors(such as temperature, heater, air cleaner sensors and so on).

Elastic Semantic Engine Layer

B. Elastic Processing Engine

1) M2M Data Model: All M2M data is modeled as the RDF Stream. Every virtual entity aggregates the RDF streams from corresponding several windows. A window extracts from its sensor stream the latest elements , which is considered by the virtual entity. Besides the streaming M2M data, some M2M applications need auxiliary background

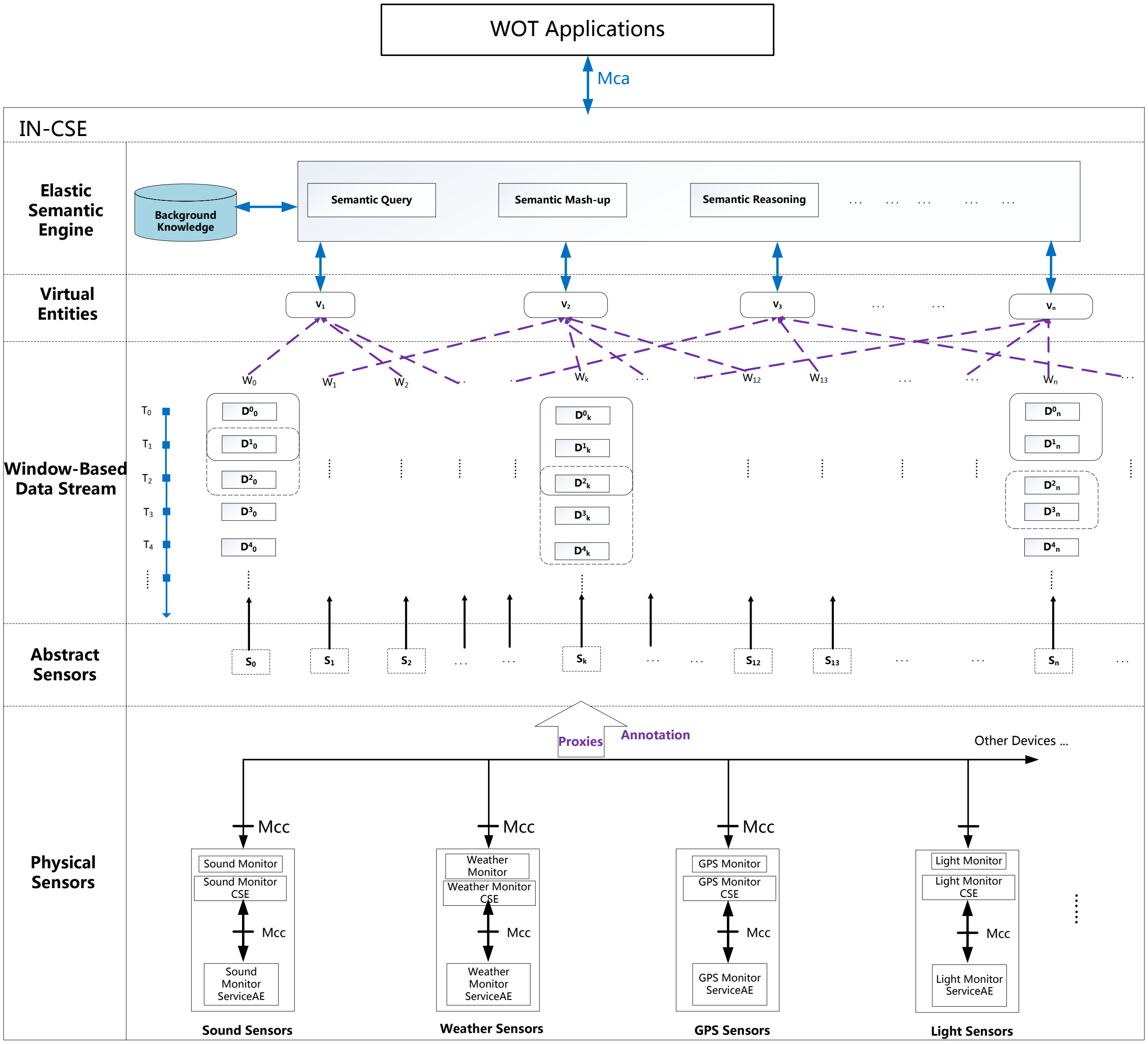

Figure 2. Semantic Processing Framework For M2M Applications

knowledge, such as Linked Open Data, DBpedia and so on. Related definitions are as follows:

Definition 1: **RDF Stream(S).** The basic data unit for RDF Stream is ($< s, p, o >$,timestamp). A RDF Stream is defined as an ordered sequence of pairs, where every pair is constituted by multiple basic data units, denoted as $D_i{}^j$, i represents the identification ID of certain sensor, j is the timestamp. For example, S_i={$D_i{}^0$, $D_i{}^1$, $D_i{}^2$,...,$i \in \mathbb{N}$} denotes the RDF stream data of the ith sensor.

Definition 2: **Window(W).** A window is a subset of the RDF Stream given a time range t. W_i(t)= {$D_i{}^0$, $D_i{}^1$, $D_i{}^2$,..., $D_i{}^(t-1)$}denotes the RDF stream data of the ith sensor within the latest t logical time unit(second, minute, hour, number...). Windows are sliding when they are progressively advanced of a given STEP. For example, the size of window and sliding window for W_0 in the figure 2 are 2 and 1, respectively.

Definition 3: **Virtual Window(VW).** Virtual window aggregates the related data needed by a virtual entity, which includes the corresponding window data and background knowledge $B = (s, p, o)$.

2) M2M Semantic Query: M2M Semantic Query is a basic function for M2M applications. It enhances the M2M discovery mechanism, to allow locating and linking resources or services based on their semantic information. For example, Get the temperature of the room 1", "Get the all rooms whose PM2.5 $\geq$ 80".

The query task is formalized via the concept of mapping. We denote as I, B, L, V respectively the domains of IRIs, blank nodes, literals, and variables which are all disjoint. We also define T = (I $\cup$ B $\cup$ L). A mapping μ is a partial function μ : V $\rightarrow$ T which gives the bindings for all the variables of a query. Evaluation occurs when a graph pattern (denoted as P) in the query is matched against a Virtual Window (VW). P is a set of triple patterns t = (s, p, o) such that s, p, o $\in$ (V $\cup$ T). We then define dom(μ) as the subset of V where μ is defined (i.e., the domain of μ and use the notation μ(x) to refer to the bindings of variable x in μ.

3) M2M Semantic Mash-up: Semantic mash-up provides functionalities to support new services through the creation of new virtual devices, which do not exist in physical world, by obtaining semantic information through semantic descriptions from existing M2M resources in the M2M System. For example, "compute the indoor air quality index of a room" is a typical Mash-up application, which needs to accomplish a task based on various window data.

4) M2M Semantic Reasoning: Reasoning is a mechanism to derive a new implicit knowledge from semantically annotated data and to answer complex user query. It can be implemented as a piece of software to be able to infer logical consequences from a set of asserted facts or axioms.

5) M2M Semantic Spatial-Temporal Processing:

IV. USE CASE: HOME ENVIRONMENT MONITORING

It is designed to facilitate the smart real-time monitoring to the home environment by providing three concrete application examples. We first give a brief overview of the use case, then

A. Overview

Nowadays, with the popularity of smart home appliances (e.g., heater, air conditioner, humidifier, air cleaner, etc.) equipped with environment sensors (e.g., sensors for temperature, humidity, CO1, CO2, VOC(Volatile Organic Compound) level, etc.), large volumes of data from all aspects of home environment is available, which makes it possible to implement various home monitoring applications including emergency detection(gas-leaking, fire accident), environmental index

The use case shows three representative scenarios of home environment monitoring: Indoor Sensor Discovery, Indoor Air Pollution Index (API), Human Comfort Index (I_{hc}). The three applications correspond to three kinds of M2M applications: M2M Semantic Query, M2M Semantic Mashup, M2M Semantic Reasoning.

As the paper mainly focuses on the background streaming data processing for M2M applications, we do not create a complex and comprehensive ontology to semantically annotate the all various M2M data. Conversely, we design a simple concept model for the home environment monitoring scenario (see Figure 3). The model captures three types of resources: Home, Room, Sensor.The label under the resource denotes the its URI. "p" is the namespace of the properties. Every sensor entity has three properties: type, value, time.

Figure 4 shows a simple snapshot data of the stream knowledge model based on the concept model. Every home contains multiple rooms(living room, bedroom, kitchen and so on). Every room is equipped with 15 kinds of sensors, including Temperature, Humility, Illumination, Volume, P-M10, PM2.5, O3, CO, SO2, NO2 and so on.

B. Scenarios

1) Semantic Query:S:

2) Semantic Mashup:AQI:

$$IAQI_p = \frac{IAQI_{Hi} - IAQI_{Lo}}{BP_{Hi} - BPLo}(C_p - BP_{Lo}) + IAQI_{Lo} \tag{1}$$

$$AQI = \max\{IAQI_1, IAQI_2, IAQI_3, ..., IAQI_n\} \tag{2}$$

3) Semantic Reasoning:I_{hc}:

$$I_{HC} = T - 0.55(1 - H_R)(T - 58) \tag{3}$$

V. MODEL IMPLEMENTATION AND EVALUATION

In this section, we briefly introduce the implementation of the M2M application framework. Then extended experiments is performed to evaluate the system from the functionality and scalability.

A. Model Implementation

Followed by proposed framework, we builded our elastic streaming processing engine based on the efficient inmemory cluster computing framework-SPARK, which provides us with rich data abstraction and operation abstraction to meet the needs of various M2M applications. We use the DStream(Discreted Stream) to model the window stream and implement related functions by translating the to the operators of SPARK including "filter" and "join" transformation .

Presently, we have preliminarily implemented the M2M query subsystem, mashup subsystem and reasoning subsystem. Every subsystem acts as a module of our elastic semantic processing engines. Once a M2M application requests service, corresponding subsystem will run a continues Spark job to return

B. Evaluation

1) Experiment Setup: **Configuration:** The experiment is implemented on a Spark cluster with three machines. Each node has 16 GB DDR3 RAM, 8-core Intel Xeon(R) E5606 CPUs at 2.13GHz, 1.5TB disk. The nodes are connected by the network with the bandwith of 1000M/s. All the nodes use CentOS6.4 with the softwares JDK-1.7.0, Scala-1.10.1 and Spark-0.9.0.

Data: The experimental data is generated by our stream data generator whose schema is based on the concept model in figure 3. The main parameters of the generator are R and T, denoting the number of home and sampling time, respectively. The number of home is in proportion to the number of sensor denoted by N_s(N_s=15*5*R, 15 and 5 represent the number of sensors in a room and rooms in a home). Sampling time stimulates the rate of sensor stream. The average data size generated by a sensor within a sampling time is 0.5(KB).

2) Functionality Evaluation:

3) Scalability Evaluation: During the process of spark streaming execution, we will write a total delay(TD) into the log file after the data in a time slice has been processed completely. The parameter records the total time from receiving window data to output final results. In our experiment, the time slice(D) is set as 5 seconds and we will run the program for 300 seconds. That is to say, 60 TD will be written into the log file.

Figure 4 and Figure 5 show the tread of the processing time(TD) in single node and cluster with varied sensors. For single node experiment, the number of sensors is varied from 15,000 to 150,000. For cluster experiment, the number of sensors is varied from 75,000 to 750,000. The two figures only show the processing time for parts of sensors so that we can recognize the broken line well. From both figures, in the beginning of executing a spark job, TD is not stable($0\tilde{5}0$). After a while, TD will stay in a comparatively stable level. Here we choose these TD in the last 250s and compute their average value denoted as TD_{SA}. To capture the fluctuation of the processing time, we compute the standard deviation σ of TD_{SA}.

Equation 4 computes the system's throughout(Q): 0.5 is the average data size generated by a sensor in a sampling time, N_S is the number of sensors, TD_{SA} is the processing time.

$$Q = \frac{1}{1024} \times \frac{0.5 N_s}{TD_{SA}} \tag{4}$$

$$Sizeup = \frac{computing \quad time \quad for \quad processing \quad m \times data}{computing \quad time \quad for \quad processing \quad data} \tag{5}$$

Table II and Table III show the execution results in single node and cluster. Figure 6 and Figure 7 show the throughput and processing time with increased sensors. We can conclude the following results from the :

Firstly, we can get the correlation among relevant variables: TD_SA, Q, N_s. From the graphs we can see that TD_SA and Q increase with the increasing numbers of sensors. But when the ratio of TD_SA to D reaches a certain value, the throughput will decrease rapidly. This is because the computing capability of the system will not be able to catch up with the velocity of the data stream. As a result, the delay caused by the last time slice will have an effect on the next process and the system will become more and more unstable. The σ will also increase evidently because the TD_SA has a larger fluctuation. The correlation analysis can help us control the rate of input stream and set proper time slice to accommodate the ability of the system.

Secondly, Table II and Table III show that our system achieves high throughputs: more than 53MB/s and 175 MB/s in single node and cluster. Benefiting from the system's elastic processing ability, it can concurrently process more than 300,000 sensor streams efficiently.

At last, the tables show that our system achieves excellent scalability. For both single and cluster configuration, the sizeup of m times input is much less than m. Especially for the cluster, when the input stream increases by 6 times($N_s = 262,500$), the processing time only increases by less 2(1.72) times. The results mean that the TD increases much more slowly than input data size and our system works better in processing larger input stream. At the same time, the tables also show that the processing capability of cluster is much better than single node. For example, when the number of input stream sensors is 150,000, the execution time in cluster is 0.575, compared to 2.440 in single node. The best throughput in cluster (175MB/s) is more than 3 times of the one in single node(53MB/s). It proves our system achieves good flexibility and elastic scalability: It can adapt to various different application scenarios and requirements by adding computing nodes.

To sum up, the results demonstrate excellent scalability regarding both the size of input stream and number of nodes.

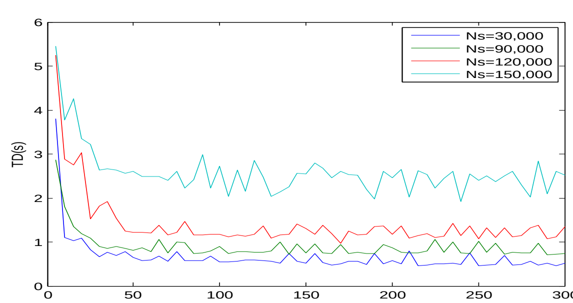

Figure 3. TD in single node

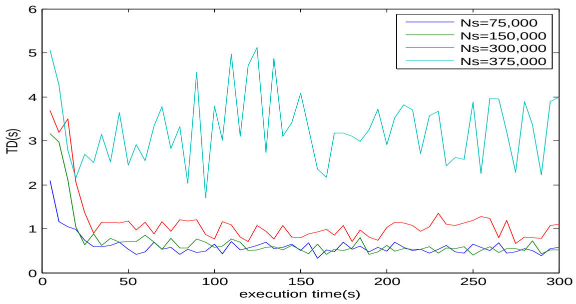

Figure 4. TD in cluster

Table I
THROUGHPUT IN SINGLE NODE

N_s	TD(s)	TD/D	σ	Q(MB/s)	Sizeup
15,000	0.308	6.17%	0.06584	23.759	1
30,000	0.566	11.32%	0.08999	25.879	1.836
45,000	0.635	12.70%	0.09619	34.605	2.06
60,000	0.727	14.54%	0.11524	40.305	2.36
75,000	0.751	15.02%	0.11416	48.769	2.44
90,000	0.821	16.42%	0.10602	53.518	2.66
105,000	0.956	19.12%	0.11979	53.635	3.10
120,000	1.218	24.37%	0.13368	48.092	3.95
135,000	1.785	35.70%	0.19057	36.930	5.79
150,000	2.440	48.81%	0.24599	30.012	7.92

Table II
THROUGHPUT IN CLUSTER

N_s(k)	TD(s)	TD/D	σ	Q(MB/s)	Sizeup
37,500	0.438	8.76%	0.08675	41.825	1
75,000	0.543	10.85%	0.08929	67.474	1.24
112,500	0.566	11.32%	0.10169	97.025	1.29
150,000	0.575	11.49%	0.10676	127.452	1.31
187,500	0.623	12.46%	0.13582	146.922	1.42
225,000	0.627	12.54%	0.13051	175.160	1.40
262,500	0.753	15.07%	0.14888	170.142	1.72
300,000	0.991	19.83%	0.17464	147.753	2.26
337,500	2.519	50.37%	0.56164	65.431	5.75
375,000	3.294	65.89%	0.78245	55.583	7.52

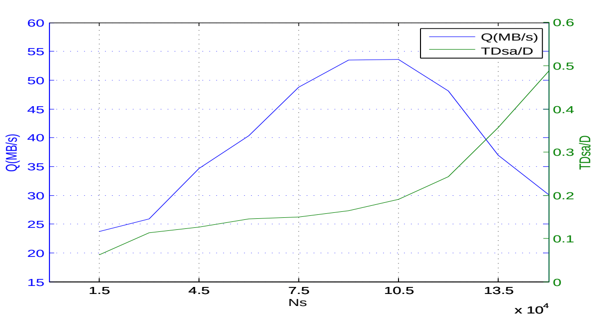

Figure 5. Throughput and TD with Increased Sensors in Single Node

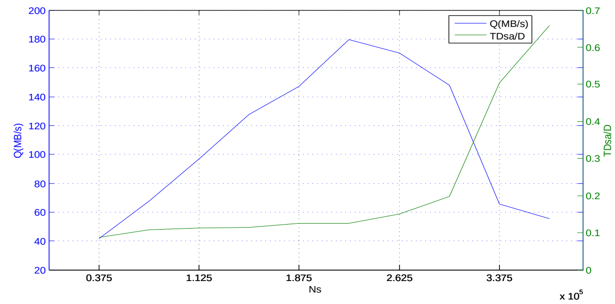

Figure 6. Throughput and TD with Increased Sensors in Cluster

VI. CONCLUSION