2013 IEEE 14th International Conference on Mobile Data Management

Distributed Complex Event Processing in Sensor Networks

Omran Saleh
Faculty of Computer Science and Automation
Ilmenau University of Technology
Ilmenau, Germany
Email: omran.saleh@tu-ilmenau.de

Advisor: Kai-Uwe Sattler
Faculty of Computer Science and Automation
Ilmenau University of Technology
Ilmenau, Germany
Email: kus@tu-ilmenau.de

***Abstract*—Mobile systems which include sensor networks, generate a continuous unmatched volume of primitive event streams with various properties where the interior semantic information of these events is generally very limited. Moreover, forwarding all these events to a central processing entity is not favored for limited-resource systems, it would deplete nodes' resources e.g., the energy.**

The goal of this work is to address these issues by extracting useful information locally, transmitting only this information and avoiding transmission of unnecessary low-level data between nodes. Therefore, an In-Network Distributed Complex Event Processing (INDCEP) based solution is proposed. INDCEP technology is used to perform the processing within the network by pushing complex event processing into network nodes. This research aims to develop a robust and high performance Distributed Complex Event Processing Engine for Mobile Systems (CEPEMS) based on aforementioned technology. It focuses on distributed event detection via disseminating distributed plans into a network of sensor nodes to perform complex event tasks. Furthermore, we present an optimization technique for the distributed queries and cost model to decide which query portions can be executed whether inside sensor nodes or at local CEP engine instance in a centralized point.

I. Introduction

Recently, mobile networks and especially sensor networks have become an important field of study. They have recently emerged as a substantial area of computing. These networks consist of heterogeneous distributed collections of devices which cooperatively operate to create an environment that has better characteristics than individual sensor devices and is appropriate for carrying out the required tasks.

The vast majority of mobile sensors network software solutions share the same functionality i.e., gathering of data regarded as events to be eventually delivered to a central processing entity. The large scale of sensor networks will generate continuous, massive amount of events. Transmitting all these events to this entity for complete processing makes the network overcrowded with superfluous data. Since the huge volume of transmissions among nodes would drain the nodes' resources e.g., the energy, and affect their performance, a central architectural approach is not a preferable solution. As the data volume grows rapidly, this approach for processing becomes more and more inapplicable.

Moreover, the events generated by sensor nodes are primitive. We can only obtain some naive information from these events. Most of real sensor network applications focus on complex information and the possibilities to find pattern matching over event sequences. Recognition of the event sequences provides us with such complex information by combining several primitive events together according to a set of rules.

One way to remedy the first problem is to utilize an in-network processing concept. Communication among the sensor devices and the central entity is expensive compared to local computation within the network which is much cheaper [1]. Thus, pushing more processing out into the network could significantly improve energy consumption. In-network processing aims to perform as much processing as possible within the network by enabling some operators, particularly filters, in the network nodes. Thus, the communication cost between neighboring nodes is reduced.

To handle the second problem, Complex Event Processing (CEP) has been proposed. In general, CEP is an emerging technology focusing on the correlation of simple events to generate complex events and detect the occurrence of event patterns on streaming data such as sensor network data stream. This technique is advisable for applications that should carry out processing on continuous event streams, and make low latency and real-time decisions. It allows applications to extract, understand and report valuable information by processing the stream instead of raw data.

Most of the conducted works in sensor network field avert to address the wider issues of how to integrate CEP with in-network processing. CEP has mostly been neglected to process inside the sensor nodes. In our research, an In-Network Distributed Complex Event Processing (INDCEP) based solution and its CEPEMS engine are proposed. We integrate the new engine with AnduIN [2] in order to receive and validate a user query, and generate CEP plans before disseminating them to the network.

The rest of the paper is organized as follows: related works are briefly reviewed in section II. Section III introduces our current approach and how it can be applied in sensor networks. AnduIN integration with INDCEP approach and CEPEMS is provided in section IV. In section V, our main contributions are listed. Finally, section VI presents our concluding remarks.

II. Related works

A large number of systems in the fields of complex event processing and in-network query processing which are related to our research have been proposed. There is also some research that employs the idea of CEP to process RFID or sensor networks streams as a central architectural approach. However, the vast majority of the early works take into account only one of these fields and disregard the other. In this section, we provide a brief overview about these systems.

CEP This concept has recently attracted attention by the community and has been studied widely in several systems. Many complex processing engines have recently been developed such as Cayuga [3] and SASE [4]. SASE deals with events as regular expressions. It detects events based on Non-deterministic Finite Automaton (NFA) and Active Instance Stack (AIS). It has a specific language to define events pattern and functionality. Cayuga is a high performance CEP engine, which is scalable to process events with high arrival rate as well as large number of queries by using custom heap management and indexing of operator predicates. The queries are expressed through Cayuga algebra. It uses finite state automata to implement the system. In addition to research prototype engines, several commercial CEP systems exist e.g., Esper [5].

Reference [6] describes a CEP system that can rewrite a query into a more powerful format and translate it to automata. To get more scalability, these automata can be distributed on a set of machines. Paper [7] demonstrates SASE through processing RFID stream data for a real-world retail management scenario. The authors of [8] and [9] used Timed Petri-Net (TPN) and workflow based methods, respectively, to model new complex event engines for RFID stream. Paper [10] proposes an application based on CEP for object detection tracking in sensor networks using Esper engine. In [11], a general software architecture is presented to process sensor networks data based on Event-Driven Architecture (EDA), CEP and ontologies concepts. All above-mentioned references used sensor nodes only for gathering the sensor readings.

The closer works in spirit to our work are presented in [12] and [13]. The latter proposes a framework to execute complex event processing tasks. There are three types of nodes in the framework: base station, proxy, and the ordinary nodes. The ordinary nodes detect primitive events and forward them to a proxy node where sub-events are detected. The proxy node later forwards sub-events to a base station in order to detect the complex event. The former uses the idea of query divider to split queries into server and node queries, where each query can be executed. The results from the two sides are combined by results merger. None of these approaches follows the typical approach to detect the complex event using finite automata, tree or graph. Furthermore, most of the processing load in these systems would be on a central server or proxy node.

In-Network Processing The idea of in-network processing is adopted by several systems such as TinyDB [14], Cougar [1], REED [15] and Frameworks in Java for Operators on Remote Data streams (Fjords) [16]. Fjords consists of two major components: adaptive query processing engine and sensor proxies. Adaptive query processing engine can combine streaming data with static data from traditional disk to process via pull and push mechanism. Sensor proxies are the bridge between sensor nodes and the query processing engine which performs simple local computation such as filtering and cashing on the delivered samples from sensor nodes.

Cougar is a system in which the sensor network is foreseen as a huge distributed database system. In Cougar, an additional query layer (query proxy) is placed in sensor node to perform in-network processing. Two query plans are generated, one is for the leader node to perform aggregation and transmit the data to the gateway node, the second plan is for non-leader nodes to measure and extract only relevant sensor data. TinyDB is a distributed query processor that runs on each node in a sensor network. It collects the data from motes, filters them, aggregates them together, and routes them out to a base station. It provides an SQL-like interface for querying the network. REED is an extension for TinyDB in which filter conditions are stored in tables, those tables are disseminated into multiple nodes. Each sensor reading is checked against all of these conditions. When this reading satisfies all conditions, it is outputted.

Most approaches in sensor networks have focused on data aggregation and routing algorithms to minimize the hop count between the nodes. In these approaches, several in-network aggregation techniques have been developed that significantly reduce the amount of data transmission and enhance the life time of sensor network by eliminating redundant data. Examples on data aggregation approaches are tree-based aggregation protocols i.e., TAG [17].

III. Applying INDCEP in sensor networks

Sensor networks are gaining a great relevance in many application domains. There is an infinite number of applications that can greatly take advantage of such systems including military, disaster detection, environmental and healthcare applications. When designing a sensor network system, it is important to recognize the circumstances under which it will operate and understand its characteristics such as limited computational power, memory and energy resources.

Many sensor network applications such as fire detection, storm detection, flood detection, tracking suspicious behavior and monitoring pollution levels require detection of higher-level composite events instead of raw events over high-volume event streams. These streams should be processed in real-time manner to respond quickly and take actions if necessary. For that purpose, we propose to shift complex event processing to the source of the stream, in our case to sensor devices. CEP can be considered as an essential part of in-network processing and vital process that can be pushed to nodes to perform as illustrated in Fig. 1. Sensor devices can utilize their highest possible level of processing capabilities to perform computations.

INDCEP is responsible for correlating primitive events from different nodes to identify higher level complex events and the event patterns of interest from a stream of events. Furthermore, it detects complex events and valuable information from a sequence of events with logical and temporal relationships by employing the idea of distributed computing. This concept authorizes our system to communicate only with neighboring nodes when it is certainly necessary or after detecting a complex event to reduce internal data traffic for sensor networks. Therefore, it avoids the energy consumption bottleneck by reducing the use of communication, especially when there are relatively few events satisfying complex event conditions.

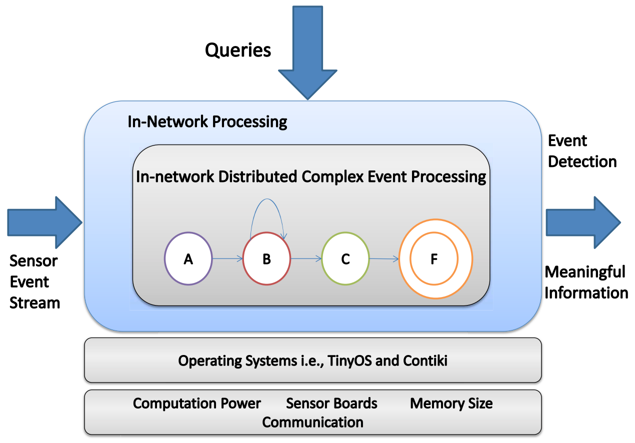

Fig. 1. In-Network Distributed Complex Event Processing for sensor node.

INDCEP should support several CEP operators such as disjunction, sequence, conjunction and some forms of negation. Disjunction of events is satisfied when one or more events out of a set of events would occur whereas conjunction of events is achieved when all the events take place. However, they can happen in any order. The sequence operator determines a specific order in which the events should occur sequentially. The negation operator is used to indicate the non-occurrence of particular events. Furthermore, a wide set of spatial and temporal operators can be provided. Some additional functions can be pushed into these devices besides CEP to help it perform extra tasks such as sampling, filtering and preprocessing the data, aggregation or even more complex analytical tasks like data mining. Our CEP function is based on Non-deterministic Finite state Automata (NFA). We have developed a query plan to implement the node process behavior. It corresponds to an acyclic directed graph of functions. These functions follow publish-subscribe mechanism to transfer sensing events from one function to another.

The main idea beyond INDCEP is that a large NFA expression can be built and then splitted into smaller NFAs (sub-expressions) according to some rules. Each sub-expression contains a query plan which has CEP functions and other functions. These plans can be matched independently on distributed nodes. In case of existence of plan dependencies between nodes, intermediate events can be sent between these nodes.

An illustrative example for this concept is to apply a sequence operator to different nodes which have multiple sensors. The user can issue a part of CEP query which contains the following pattern: **SEQ** (`A`, `B+`, (`C` **DIS** `D`), `F`) as in Fig. 2 where `A`, `B`, `C`, `D`, `F` are events, and **SEQ**, **DIS** are sequence and disjunction operators, respectively. `B` uses self-loop expression, denoted by "+", to represent one or more events of `B`. Suppose we possess three nodes, the first one can handle `A`, `B` , `F` events, whereas the second can handle only `C` event and the other only `D`. In our approach, each node has a different plan. Once the first node catches events `A` and `B` sequentially, it will try to communicate with the second and third node to examine whether a `C` or `D` event has been caught by them. If yes, the first node jumps to next state either S3 or S4. Otherwise, it stays in state S2. Reaching the accepting state (S5) indicates that the sequence is detected.

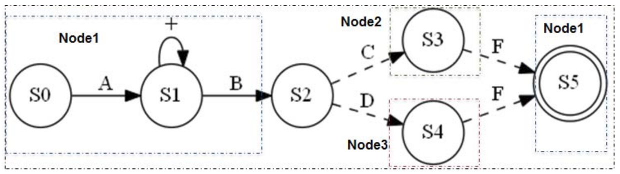

Fig. 2. Original automaton for pattern SEQ(A, B+, (C DIS D), F).

A real application for INDCEP is the fire detection system which has to be performed as close to real-time as possible. The detection of fire would be based on temperature, rain and smoke measurements. The application would trigger an alarm if the temperature was increasing, followed by smoke detection, without the occurrence of rain detection in between. We can write the above example as **SEQ**(`increasing temperature`, **NEG** `rain detection`, `smoke detection`) where **NEG** is the negation operator. In this case, the user would issue the aforementioned CEP query, and the underlying system would decide how the query would be splitted to plans (sub-expressions), how the plan would process the data and which plan would be executed in particular node.

IV. CEPEMS and AnduIN Integration

AnduIN [2] is a data stream engine to process and analyze dynamic data. It is developed by Ilmenau University of Technology and provides the standard query operators such as projection, filter, aggregation, and join. User can register the query which is expressed by using a simple SQL-like interface (CQL), in the system to process incoming data continuously.

It also combines sensor-local in-network processing and a data stream engine. Moreover, AnduIN decides which sensor network query portion can be carried out within the sensor network and which portion can be processed at the engine based on multi-dimensional cost model and taking energy consumption into account. We will add new functionalities to AnduIN for supporting internal, generic and scalable complex event processing. It would be provided by local powerful instance of CEP operators for identifying complex events and pattern matching.

In our research, we will develop a robust and high performance (CEPEMS) engine to apply INDCEP approach, this engine focuses on distributed event detection via disseminating distributed plans into a cluster of sensor nodes to perform complex event tasks. We will also add optimization techniques based on cost model and the information from *sensor catalog* to CEPEMS. The *sensor catalog* can give information about the properties of the nodes such as the computational power, memory size, communication cost and the available sensors of the node. After a query is validated, it is optimized to select the best query plans, that minimize the total cost of query execution, and reduce the size of code on sensor nodes. Query optimizers work by enumerating a set of possible plans, calculating and assigning a cost to each plan and choosing the lowest-cost plans. The optimization techniques also decide which operators can be executed in the local instance of CEP in AnduIN or which operators can be executed within the network.

CEPEMS is integrated with AnduIN engine and benefits from its feature. Other tasks can be added to AnduIN for supporting CEPEMS such as receiving and parsing queries to handle syntax errors, generating sensor network program from the optimized plans before deploying them into the network. The entire system can support various Operating Systems e.g., TinyOS [18], and various platforms e.g., TelosB and Mica2.

V. Contributions

Our main contribution is to develop a high-performance complex event detection system based on in-network processing and NFA, in which CEP can run locally inside the sensor nodes in distributed manner and can operate at tremendous streams of incoming data. In summary, the goals of this research are:

- Designing and implementing of a CEPEMS engine to employ INDCEP concept. This engine is responsible for generating distributed plans for CEP queries to perform filtering, projection, sampling, identifying the event patterns and detecting complex events based on NFA and finally transmitting events. These plans should process continuous event streams in real time manner and apply distributed processing strategies to work on events among several sensor nodes and among AnduIN and sensor nodes.
- Presenting an optimization technique to write CEP distributed queries into more efficient forms before disseminating them to the network. Moreover, we demonstrate a cost model based on some criteria such as energy consumption to decide the best distribution strategy and which subqueries can be executed inside sensor nodes and which portions can be executed at the local CEP engine instance in AnduIN.
- Applying INDCEP concept to various event detection and disaster relief applications for sensor networks including fire detection and tracking suspicious behavior such as tracking pollution levels.
- Tackling challenges such as extending the lifetime of a sensor network and obtaining useful information by distributed complex event detection. These challenges carry the most weight in terms of importance. The effectiveness of our approach such as energy conservation and system performance will be evaluated. The performance can be measured in terms of latency or throughput.

VI. Conclusion

This paper has briefly introduced in-network distributed complex event processing. We have described a novel approach to apply Complex Event Processing in sensor networks in order to meet the real-time requirements and network characteristics. We believe that INDCEP and its engine are of critical importance for detecting valuable information from sensor networks and transmitting them rather than exhausting the energy for transferring low-level sensor information to the destination. This leads to prolong nodes and networks lifetime. Finally, most of our proposed system still at an early stage and we still need a long research to accomplish a "complete" system that fully applies distributed complex event processing in sensor networks.

References

[1] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Rec.*, vol. 31, pp. 9–18, Sept. 2002.

[2] D. Klan, M. Karnstedt, K. Hose, L. Ribe-Baumann, and K. Sattler, "Stream engines meet wireless sensor networks: cost-based planning and processing of complex queries in AnduIN, distributed and parallel databases," *Distributed and Parallel Databases*, vol. 29, pp. 151–183, Jan. 2011.

[3] A. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. White, "Cayuga: a general purpose event monitoring system," in *Conference on Innovative Data Systems Research (CIDR)*, pp. 412–422, VLDB, 2007.

[4] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, (New York, NY, USA), pp. 407–418, ACM, 2006.

[5] EsperTech, "Event stream intelligence: Esper & NEsper." http://www.esper.codehaus.org/.

[6] N. P. Schultz-Møller, M. Migliavacca, and P. Pietzuch, "Distributed complex event processing with query rewriting," in *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, (New York, NY, USA), pp. 4:1–4:12, ACM, 2009.

[7] D. Gyllstrom, E. Wu, H. Chae, Y. Diao, P. Stahlberg, and G. Anderson, "SASE: complex event processing over streams (Demo)," in *Conference on Innovative Data Systems Research (CIDR)*, pp. 407–411, 2007.

[8] J. Xingyi, L. Xiaodong, K. Ning, and Y. Baoping, "Efficient complex event processing over RFID data stream," in *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pp. 75–81, May 2008.

[9] C. Zang and Y. Fan, "Complex event processing in enterprise information systems based on RFID," *Enterp. Inf. Syst.*, vol. 1, pp. 3–23, Feb. 2007.

[10] R. Bhargavi, V. Vaidehi, P. T. V. Bhuvaneswari, P. Balamuralidhar, and M. G. Chandra, "Complex event processing for object tracking and intrusion detection in wireless sensor networks," in *ICARCV*, pp. 848–853, IEEE, 2010.

[11] J. Dunkel, "On complex event processing for sensor networks," in *ISADS*, pp. 249–254, 2009.

[12] P. Li and W. Bingwen, "Design of complex event processing system for wireless sensor networks," in *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, vol. 1, pp. 354–357, Apr. 2010.

[13] Y. Lai, W. Zeng, Z. Lin, and G. Li, "LAMF: framework for complex event processing in wireless sensor networks," in *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, pp. 2155–2158, Dec. 2010.

[14] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, pp. 122–173, Mar. 2005.

[15] D. J. Abadi, S. Madden, and W. Lindner, "REED: robust, efficient filtering and event detection in sensor networks," in *Proceedings of the 31st International Conference on Very large Data Bases*, VLDB '05, pp. 769–780, VLDB Endowment, 2005.

[16] S. Madden and M. Franklin, "Fjording the stream: an architecture for queries over streaming sensor data," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 555–566, 2002.

[17] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 131–146, Dec. 2002.

[18] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGPLAN Not.*, vol. 35, pp. 93–104, Nov. 2000.