ICIC Express Letters
Part B: Applications
Volume **8**, Number **11**, November **2017**

 ISSN 2185-2766
pp. **1529–1536**

AN ESPER-BASED FILTERING SYSTEM FOR REAL-TIME DATA STREAMS

Sebin Park, Sanghun Lee, Myeong-Seon Gil and Yang-Sae Moon

Department of Computer Science
Kangwon National University
1 Kangwondaehak-gil, Chuncheon-si, Gangwon-do 24341, Korea
{ sebinpark; sanghun; gils; ysmoon }@kangwon.ac.kr

Received May 2017; accepted August 2017

Abstract. *In this paper, we deal with the filtering problem of data streams. The data stream is continuously generated, and its size is huge. In order to process and analyze the data stream in real time, we need to sufficiently remove unnecessary data through filtering. However, existing filtering algorithms can be applied to a single data format only, and it is very difficult to apply them to a variety and complex stream environments. To solve this problem, we propose a filtering system that can choose various filtering algorithms according to the stream format. The proposed system is based on Esper, which is a representative open source data stream management system (DSMS) for real-time filtering support. Using Esper we can filter data streams in real time anywhere, anytime, based on a Web-based client-server model. Our system supports real-time stream and/or bulk stream as the input data. In addition, we implement typical filtering algorithms including query filtering, Bloom filtering, and Bayesian filtering to operate in real time. Through the real implementation of the proposed filtering system, we show that the user can extract only meaningful data more accurately and efficiently by exploiting various filtering algorithms.*
Keywords: Data stream processing, Esper, DSMS, Real-time purification, Filtering

1. **Introduction.** In this paper, we deal with the real-time filtering problem of data streams. Filtering is a way of removing unwanted data, such as spam mails. Since the data stream is generated in real time, we need to sufficiently remove unnecessary data for analyzing it very fast. However, most existing filtering algorithms work well for a specific type of data [11,12]. Therefore, it is hard to accurately filter unnecessary data using a single filtering algorithm for various types of data stream or mixed stream environments [1]. To solve this problem, we propose a filtering system that can choose a filtering algorithm based on the data stream format and the analysis purpose.

Data processing is divided into batch processing and real-time processing. Batch processing is a method of analyzing already stored data, and real-time processing is a method of real-time analyzing input data in main memory. In general, the data stream is generated in real time and is large in size, and thus, it is difficult to store them in a database for the analytical purpose. To analyze the data stream in real time, we can use an in-memory based data stream management system (DSMS) [2]. Esper [3] is a representative open source DSMS for real-time processing of data streams. It supports event processing language (EPL) [4], which is similar to SQL and capable of complex stream processing. However, Esper only provides EPL-based query filtering for stream filtering, and it does not support various existing filtering algorithms. Therefore, in this paper, we propose an Esper-based real-time filtering system applying various filtering algorithms to Esper for efficient processing of real-time data streams.

The proposed system is designed and implemented based on the client-server model. The operation procedures are as follows. The client passes the user selected data stream,

filtering algorithm, and filter conditions to the server. Then it displays real-time filtering results from the server. The server constructs a filter model and a filtering algorithm based on the information received from the client and applies it to Esper to filtering the data stream in real time. The server then returns the filtering results to the client in real time.

2. **Related Work.** Filtering [5] is a purification method that removes unwanted data from the input data for the purpose of improving the quality and reliability of the data. Filtering algorithms can be divided into supervised learning-based filtering and unsupervised learning-based filtering. Supervised learning-based filtering is an intelligent method of purifying the input data by learning the filter model before filtering. Representative algorithms for this method are Bayesian filtering [6,13], content-based filtering [1,14], and Kalman filtering [6,15]. On the other hand, unsupervised learning-based filtering is a purification method that does not require an additional learning procedure for the filtering. Typical algorithms for this method include hash filtering [1,16], query filtering [7,17], and Bloom filtering [1,18]. In this paper, we implement the proposed system using three filtering algorithms: Bayesian filtering with supervised learning-based filtering, Bloom filtering and query filtering with unsupervised learning-based filtering. We choose these filtering algorithms since they are easily applicable to the data stream environment and are most commonly used in many fields such as classification and detection search.

Esper is a representative open source DSMS for the analysis of complex event processing (CEP) [8] and event series. CEP is an in-memory technology for real-time processing of a large-capacity stream generated from various data sources. Figure 1 shows the detailed structure of Esper. As shown in the figure, Esper consists of input adapters, a CEP engine, and output adapters. For the detailed description of Esper, readers are referred to [3].

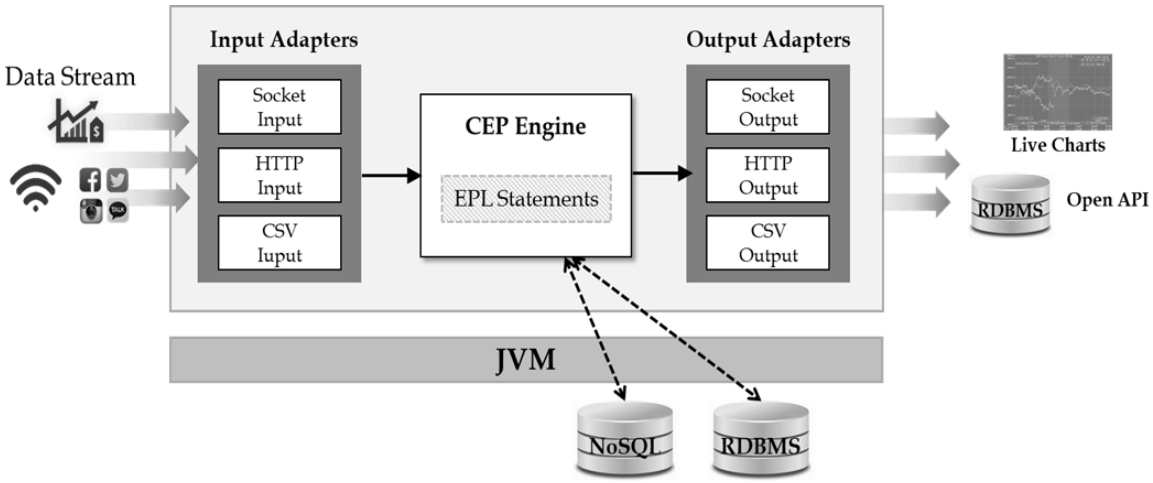


FIGURE 1. Overall working structure of Esper

As a specific language to the stream domain, Esper supports EPL for query processing and execution. EPL is a query language like SQL that has SELECT, FROM, WHERE, and HAVING clauses. Likewise, Esper is not only capable of real-time processing of the data stream, but also easy for the user to use. Therefore, in this paper, we propose a useful system that applies various filtering algorithms to Esper and selectively provides filtering algorithms according to data characteristics and analytical purpose.

3. **Esper-based Real-time Filtering System.** The Esper-based filtering system proposed in this paper is based on a client-server model as shown in Figure 2. The client-server

model is a network-based architecture widely used by many analysis and management systems. Also, it is easy to maintain and allows multiple users to use the same system at the same time. In this paper, we implement the client as a Web-based system so that users can exploit the filtering system anytime and anywhere through the Web browser. The working procedure of the client and the server is as follows. First, the client chooses the data stream from the user and selects a filtering algorithm suitable for the data format and analytical purpose. Then, the client receives the filter conditions to be used for filtering and sends it to the server (①). Next, the server generates the data stream, the filter model, and the filtering algorithm using the received filter conditions and applies it to Esper (②). After then, the server filters the data stream in real time (③). Finally, the server sends the filtered result back to the client in real time, and the client visualizes the filtering result to the user in real time and stores them as a file (④).

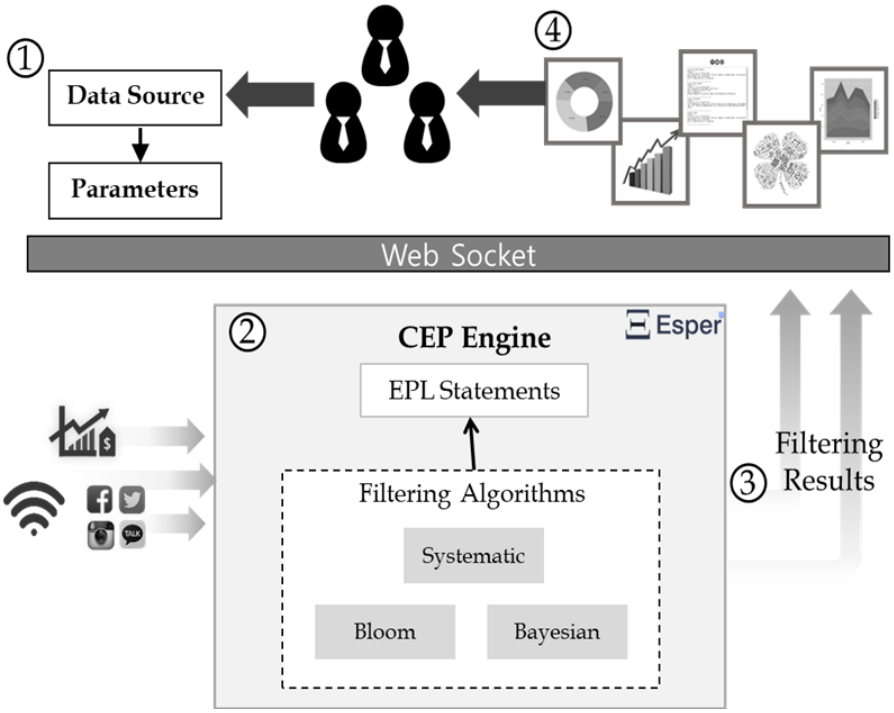


FIGURE 2. The overall architecture of the Esper-based real-time filtering system

Figure 3 shows the detailed working procedure of the client-side in our filtering system. As shown in the figure, the client consists of five modules: data source input, algorithm selection, condition input, communication, and output modules. The detailed functions of each module are as follows. **Data source input module** receives the data stream to be filtered, where the proposed system supports real-time and bulk data streams. The real-time data stream inputs source information in which data is generated, and the bulk data stream directly inputs a file stored in the disk. The input data is filtered by the algorithm selected by the user through the **algorithm selection module**. The algorithms supported by the proposed system are query filtering, Bloom filtering, and Bayesian filtering. **Filling condition input module** receives the filter condition for generating the filter model. If the user uses query filtering, it enters the condition to the EPL WHERE clause, and if the user uses Bloom filtering, it registers the condition for the Bloom filter. In the case of Bayesian filtering, unlike the above two filters, the learning data must be used to generate the filter model. Therefore, the user inputs conditions for generation of learning data in Bayesian filtering. **Communication module** serves to exchange data with the server. In particular, it uses HTTP and Web socket communication [9] to transmit data streams, filtering algorithms, and filter conditions from the user to the server,

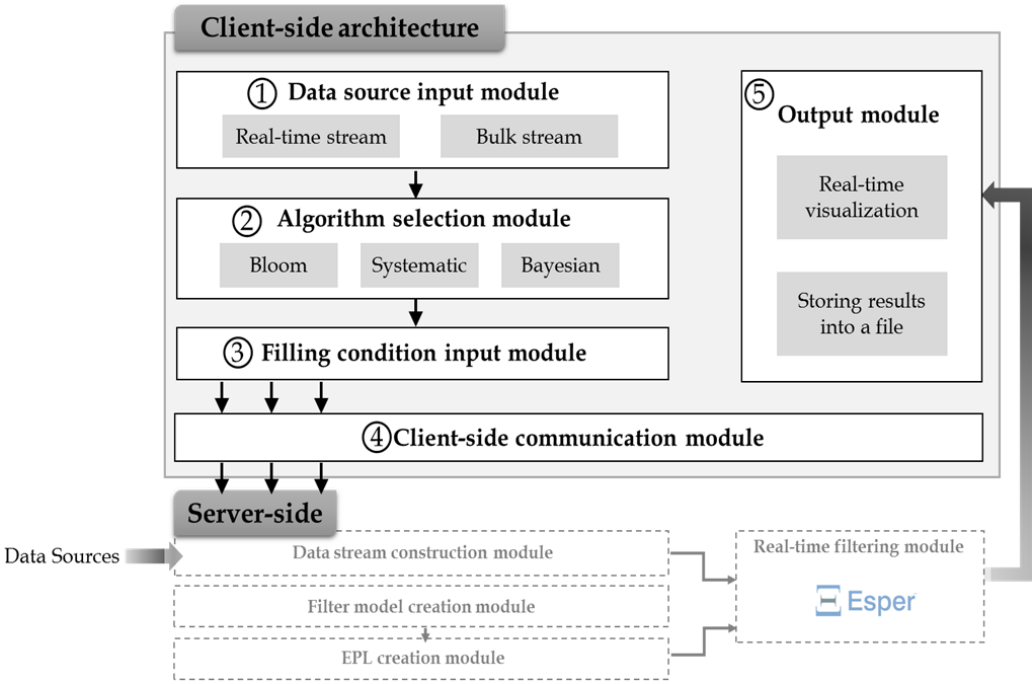


FIGURE 3. The client-side working procedure of the proposed system

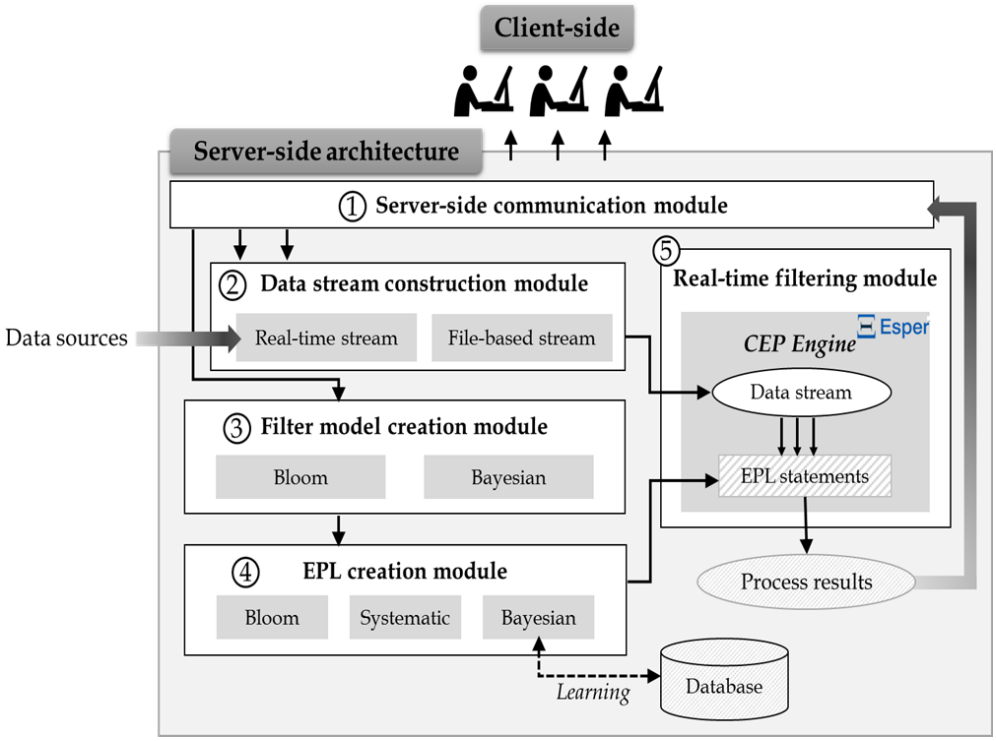


FIGURE 4. The server-side working procedure of the proposed system

and receives the filtering results from the server in real time. **Output module** visualizes the filtering results delivered from the server in real time to the user. If the user requests to save the result, this module stores the filtering results as a file and provides it to the users.

Figure 4 shows the detailed working procedure of the server in our filtering system. In this figure, the server consists of five modules: communication, data stream construction, filter model creation, EPL creation, and real-time filtering modules. The detailed functions of each module are as follows. **Communication module** exchanges data with

the client and is the same as the communication module of the client. **Data stream construction module** constructs a data stream from a data source or a file. When the data stream received from the client is a data source, the data stream is constructed from the source in real time. If the data stream received from the client is a file, this module converts the file directly into a data stream. **Filter model creation module** builds a learning model for Bloom and Bayesian filters. This module receives the filtering conditions from the client and creates a model for each filtering algorithm based on the conditions. The results of each learning model are reflected in the conditions of Bloom and Bayesian filters. **EPL creation module** generates each algorithm in an EPL form based on user input conditions and learning results of the filter model. The generated EPL type algorithms filter the input data stream in real time on the Esper's CEP engine-based **real-time filtering module**. The result of filtering is delivered to the client in real time via the communication module.

4. **System Implementation and Evaluation.** The implementation environment of the proposed Esper-based real-time filtering system is as follows. To implement both client and server, we adopt Java and Apache Tomcat 7.0 using the Eclipse Java EE IDE for Web Developers on the Windows 8 operating system. As the real-time input stream, we use Twitter data called "tweet" from the Twitter API [10]. Bulk stream uses 30,000 pre-collected tweets. Figure 5 shows an example of the tweets used in the experiment. You can see that a tweet consists of five schemes: user name, user ID, creation date, language, and content.

Figure 6 shows the main screenshot of the proposed real-time filtering system. As shown in the figure, the part ⓐ is a data stream selection button, which is to select a real-time stream or a bulk stream. The part ⓑ is for selecting an algorithm to be used for filtering, which is to select one of Bloom, query, and Bayesian filtering. The part ⓒ is to input filtering conditions.

Figures 7(a) and 7(b) show the results of query filtering and Bloom filtering, respectively, using a real-time tweet stream. In these figures, the part ⓒ confirms the filtering result in real time, and the part ⓐ stores the filtered result into a file. Also, the part ⓑ

{ "UserName": "knu", "UserID": 3157128440, "CreatedAt": "Wed Feb 10 15:50:45 KST 2017", "Lang": "ko", "Text": "@Doc_Ruby_bot Hello, this is Kangwon National University." }

FIGURE 5. An example of the collected tweets

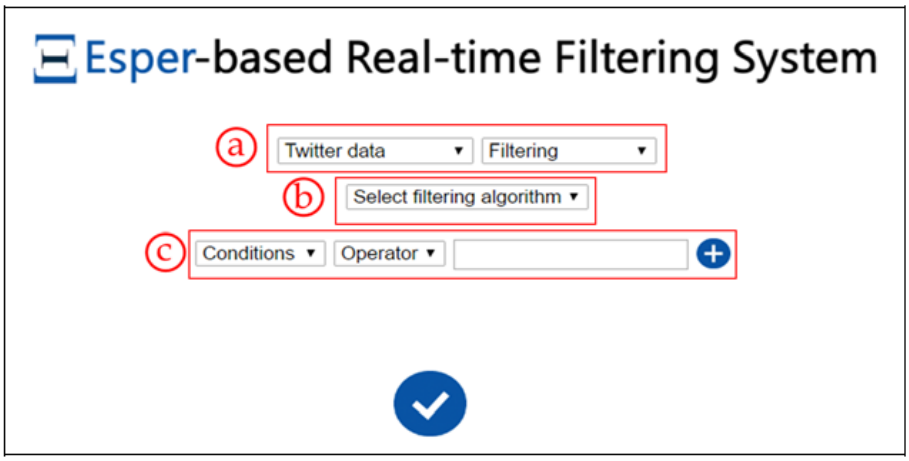
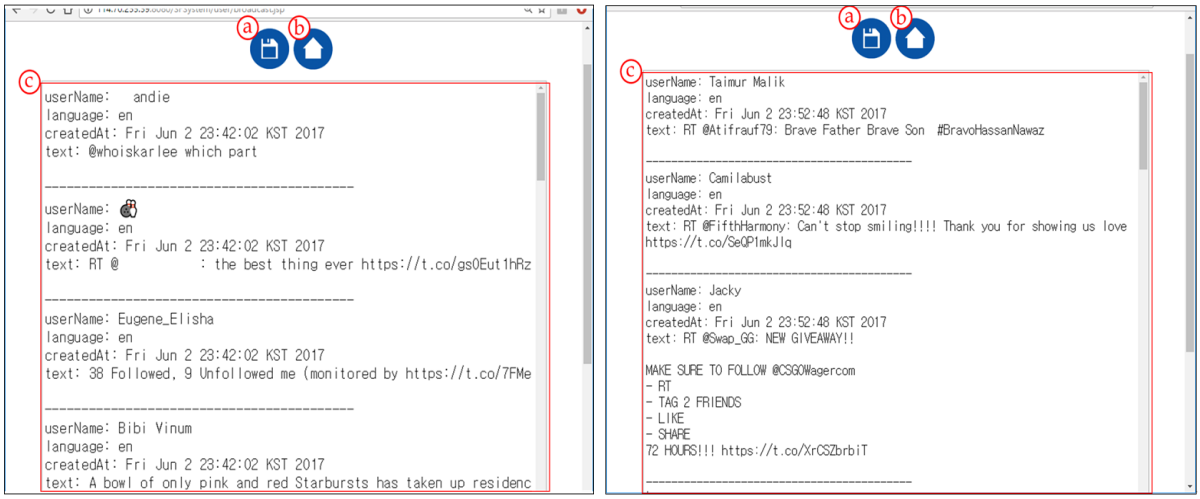
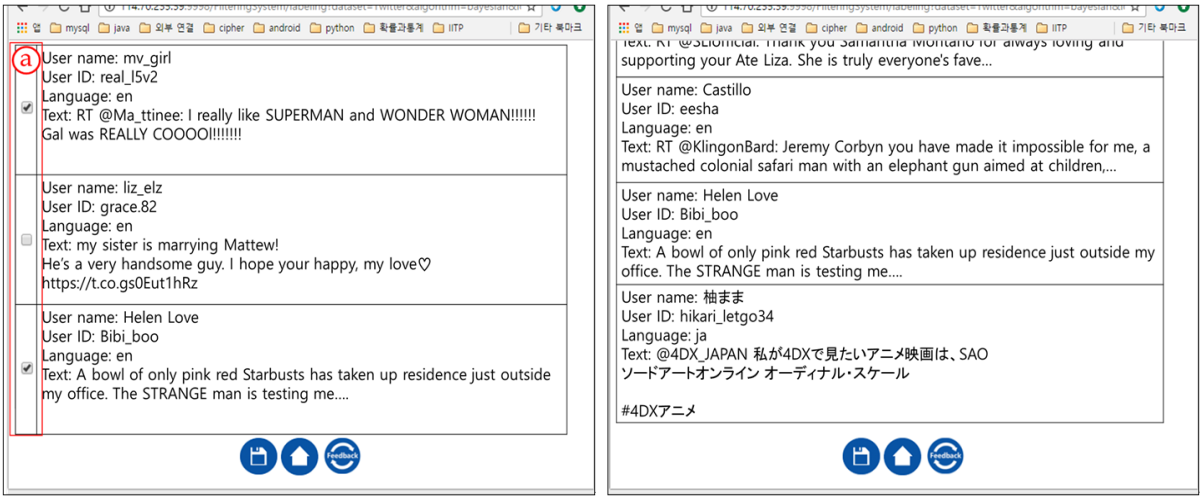


FIGURE 6. The main screenshot of the proposed system



(a) Results of query filtering (b) Results of Bloom filtering

FIGURE 7. The results of supervised filtering algorithms



(a) Screenshot of selecting learning data (b) Results of Bayesian filtering

FIGURE 8. Bayesian filtering's learning step and filtering results

returns to the initial screen of the client. In query filtering of Figure 7(a), we use "language != en" as the filtering condition. In other words, we filter all non-English tweets. The results of Figure 7 show that the language of the filtered tweets is all only English.

Figure 7(b) shows the Bloom filtering results using a real-time tweet stream. In the Bloom filter experiment, the filtering condition is "**language = en**". In addition, contrary to query filtering, it uses filtered data as results. This means that we do not want to filter tweets in English but extract it as a result. As shown in Figure 7(b), we can see that only the English tweets appears in the same way as the query filtering results in Figure 7(a).

Figure 8(a) shows a screenshot of selecting learning data in Bayesian filtering. In the Bayesian filtering experiment, we use bulk tweets and use "**language = en**" as a condition for selection of learning data. As you can see from the figure, we use that only English tweets are shown as training data according to the condition. The user classifies these learning data into the data to be filtered and to be extracted. That is, the user selects the data to be filtered in the check box of the part ⓐ, which will be used as the learning data, and the client transmits the selected data to the server. The server learns the filter model from the received learning data. In the experiment, we check the tweets with 'man' as

filtering data as shown in Figure 8(a). Figure 8(b) shows the results of Bayesian filtering with the filter model generated from Figure 8(a). However, the figure shows that both the tweets with the 'man' and the tweets without it are filtered out. This is because the performance of the Bayesian filter model is greatly influenced by the number of learning data. In this experiment, only a small amount of learning data was used, and the filtering was not performed effectively. For the more accurate filtering, we need to use a large amount of learning data.

5. **Conclusions.** In this paper, we proposed and implemented a real-time filtering system for data streams based on Esper. Since the data stream is generated in real time, we need to filter out unnecessary data very fast and very efficiently. However, existing filtering algorithms can filter only a specific type of data, and it is difficult to process various complicated data streams. To solve this problem, we propose a new filtering system which can select a suitable filtering algorithm for the stream format. We also implement the system based on Esper which can process the data stream in real time. Through the real implementation of the proposed filtering system we show that the user can extract only meaningful data more accurately and efficiently by exploiting various filtering algorithms. As the future work, we will apply the proposed real-time filtering system to Storm for the distributed processing of filtering functions.

Acknowledgment. This work was partly supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. R7117-17-0214, Development of an Intelligent Sampling and Filtering Techniques for Purifying Data Streams) and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B4008991).

REFERENCES

[1] J. Leskovec, A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*, Cambridge University Press, 2015.

[2] D. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, C. Erwin, E. Galvez, M. Hatoun, A. Maskey, A. Rasin, A. Singer, M. Stonebraker, N. Tatbul, Y. Xing, R. Yan and S. Zdonik, Aurora: A data stream management system, *Proc. of the Int'l Conf. on Management of Data*, San Diego, CA, USA, pp.1-18, 2003.

[3] *Esper*, http://www.espertech.com.

[4] H. Li, Y. Zhang and Y. Chen, PSTER: A novel probabilistic event processing language for uncertain spatio-temporal event streams of Internet of vehicles, *Proc. of the 2015 IEEE Int'l Conf. on Software Quality, Reliability and Security-Companion*, Vancouver, Canada, pp.161-168, 2015.

[5] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Dover Publications Inc., 2012.

[6] E. Alpaydin, *Introduction to Machine Learning*, The MIT Press, 2010.

[7] B. Chazelle, Filtering search: A new approach to query-answering, *SIAM Journal on Computing*, vol.15, no.3, pp.703-724, 1986.

[8] E. Wu, Y. Diao and S. Rizvi, High-performance complex event processing over streams, *Proc. of the Int'l Conf. on Management of Data*, Chicago, IL, USA, pp.407-418, 2006.

[9] M. R. Rahman and S. Akhter, Real-time bi-directional traffic management support system with GPS and WebSocket, *Proc. of the Int'l Conf. on Computer and Information Technology*, Ankra, Turkey, pp.959-964, 2015.

[10] *Twitter API*, http://dev.twitter.com.

[11] A. K. Uysal and S. Gunal, The impact of preprocessing on text classification, *Information Processing & Management*, vol.50, no.1, pp.104-112, 2014.

[12] K. Manandhar, X. Cao, F. Hu and Y. Liu, Combating false data injection attacks in Smart Grid using Kalman filter, *Proc. of the 2014 IEEE Int'l Conf. on Computing, Networking and Communications (ICNC)*, Honolulu, HI, USA, pp.16-20, 2014.

[13] N. Jatana and K. Sharma, Bayesian spam classification: Time efficient radix encoded fragmented database approach, *Proc. of the 2014 IEEE Int'l Conf. on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, pp.939-942, 2014.

[14] R. Ronen, N. Koenigstein, E. Ziklik and N. Nice, Selecting content-based features for collaborative filtering recommenders, *Proc. of the 7th ACM Conf. on Recommender Systems*, Hong Kong, China, pp.407-410, 2013.
[15] G. Ligorio and A. M. Sabatini, A novel Kalman filter for human motion tracking with an inertial-based dynamic inclinometer, *IEEE Trans. Biomedical Engineering*, vol.62, no.8, pp.2033-2043, 2015.
[16] T. Dong, J. Shi, J. Fan and L. Zhang, An improved rete algorithm based on double hash filter and node indexing for distributed rule engine, *IEICE Trans. Information and Systems*, vol.96, no.12, pp.2635-2644, 2013.
[17] J. Shin, S. Eom and K. H. Lee, Q-ASSF: Query-adaptive semantic stream filtering, *Proc. of the 2015 IEEE Int'l Conf. on Semantic Computing (ICSC)*, Anaheim, CA, USA, pp.101-108, 2015.
[18] H. M. Ju and H. Lim, New approach for efficient IP address lookup using a bloom filter in trie-based algorithms, *IEEE Trans. Computers*, vol.65, no.5, pp.1558-1565, 2016.