Journal of Information Processing Vol.23 No.4 430–440 (July 2015)

[DOI: 10.2197/ipsjjip.23.430]

Regular Paper

Development and Evaluation of Congestion Detection System Using Complex Event Processing for Mobile Networks

Hiroshi Yamamoto[1,a] Tatsuya Takahashi[1] Norihiro Fukumoto[2] Shigehiro Ano[2] Katsuyuki Yamazaki[1]

Received: September 3, 2014, Accepted: March 4, 2015

Abstract: A congestion detection on mobile networks becomes the main challenge of cellular carriers and mobile network providers because the mobile network quality easily degrades when many users concentrate on a limited place. Especially when a large-scale event is held, a heavy network congestion interferes with the communication of the participants and local residents. Therefore, the congestion detection process has been performed by several network providers, but has been executed on a high-performance computing resource in a centralized manner, which markedly increases the computing cost. On the other hand, with the wide spread of a large-scale distributed computing environment (e.g., cloud computing), a Complex Event Processing (CEP) system has recently been made available for several purposes. The CEP is a distributed computing system which can identify meaningful events by analyzing a large amount of data stream (e.g., sensor data) in real time. Here, the congestion detection can be considered as a suitable application for the CEP system, where a large amount of traffic logs (i.e., data streams) should rapidly be analyzed in order to detect network congestions (i.e., meaningful events). Therefore, in this study, we propose a new system structure of the CEP-based congestion detection system using distributed computing resources. In the proposed system, processing components are deployed on multiple resources, and execute independent tasks that are carefully extracted from a system procedure of the congestion detection. Through experimental evaluation using computing resources on a popular cloud service (Amazon EC2), it is disclosed that the CEP-based system contributes to achieve the real time detection of congestions on the mobile networks.

Keywords: congestion detection, mobile network, Complex Event Processing, cloud computing

1. Introduction

With the wide spread of smartphones and tablets, the mobile network quality easily degrades when many users concentrate on a specific and limited place. In particular, when a large-scale event is held, the total amount of network traffic becomes much larger than usual, and heavy network congestion interferes with a network communication of participants and local residents. Therefore, congestion detection on mobile networks becomes the main challenge of cellular carriers and mobile network providers.

In the congestion detection system, several measurement points are prepared on the border of access networks (e.g., access points of a mobile network) of the provider, and continuously capture data traffics that are transmitted to/from user terminals. A large amount of traffic log is collected by a centralized management server, and is analyzed for locating an access network where the network congestion occurs. The congestion detection process has mainly been executed on a high-performance computing resource in a centralized manner. However, the centralized processing requires much computing cost so that the analysis of the traffic log can be completed in real-time.

1 Nagaoka University of Technology, Nagaoka, Niigata 940–2188, Japan
2 KDDI R&D Laboratories Inc., Fujimino, Saitama 356–8502, Japan
a) hiroyama@nagaokaut.ac.jp

On the other hand, with the wide spread of a large-scale distributed computing environment (e.g., cloud computing), a Complex Event Processing (CEP) [1] system has recently been made available for several purposes. The CEP is a distributed computing system which can identify meaningful events by analyzing a large amount of streaming data (e.g., sensor data) in real time. The congestion detection can be considered as a suitable application for the CEP system, because the traffic log is the typical streaming data including multiple events, and should rapidly be analyzed in order to detect meaningful events (i.e., occurrence of network congestion).

Therefore, in this study, we propose a new system structure of the CEP-based congestion detection system using distributed computing resources. The proposed system derives representative network performance metrics (e.g., Round-Trip Time) by analyzing a large amount of traffic logs that is captured on packet capture points, and identifies the occurrence of the network congestion when the metrics change differently than usual. In the proposed system, processing components are deployed on multiple resources on the distributed computing system, and execute independent tasks that are carefully extracted from the system procedure of the congestion detection. Here, in order to decide appropriate network metrics and a functional structure, we analyze a large amount of traffic logs which is captured on a mobile network when a large-scale event, the Nagaoka Fireworks Festival, is held [2], [3]. Furthermore, through experimental evaluation using computing resources on a popular cloud service (Amazon EC2), it is disclosed that the CEP-based system contributes to achieve real time detection of congestions on the mobile networks.

The rest of this paper is organized as follows. In Section 2, existing studies related with network congestion detection and a distributed computing system are introduced. Section 3 introduces an overview of our proposed CEP-based congestion detection system. Sections 4 and 5 decide appropriate network performance metrics by analyzing an actual traffic log captured in a large-scale event, and construct a functional structure which is suitable for congestion detection based on the metrics. In Section 6, the practicality of the proposed system is revealed through experimental evaluation using a cloud computing service. Finally, the conclusion is presented in Section 7.

2. Related Works

A detection method of network failure has been proposed in order to provide high-quality and high-reliability services [4], [5]. The network failure indicates a condition where a communication is stopped due to failure or configuration errors of network equipment. However, even if the network failure does not exist, network congestion occurs when a large amount of traffic concentrates at a specific and limited area. Therefore, a network congestion detection method should be considered in a different manner from the network failure. However, the congestion detection can also be applied for avoiding the network failure because the network failure is sometimes caused by a heavy network congestion [6], [7].

The congestion detection system requires high-performance computing resources in order to derive network performance metrics (e.g., Round-Trip Time) related with the network congestion by analyzing a large amount of traffic logs (e.g., packet capture data). In addition, the heavy computation process should be quickly completed because the real-time information is necessary to prepare for the occurrence of the network congestion. Therefore, the traffic log has mainly been executed on a high-performance computing resource in a centralized manner, but the centralized processing requires much computing cost so that the analysis of the traffic log can be completed in real-time.

On the other hand, with the wide spread of a large-scale distributed computing environment (e.g., cloud computing), a batch processing and a stream processing techniques have recently been made available for analyzing a large amount data (i.e., big data) in a distributed manner. As the distributed batch processing technique, MapReduce [22] and Hadoop (as the implementation) [23] are attracting attention. The MapReduce is a distributed processing framework of the big data on clusters of commodity hardware, which splits a large set of data files into blocks, and distributes them among the computing nodes in the cluster. Therefore, the MapReduce-based processing system is not suitable for the real-time processing of the continuously generated data, but is appropriate for processing a large amount of dataset at a time (e.g., once a day).

On the contrary, the stream processing technique is composed of a set of processing components where the type of input/output stream is defined and the processing details between them are designed. Each component is deployed on the computing nodes, and can be linked with others in the cluster for constructing a large-scale pipelining system through the Internet. The Internet-scale pipelining system is suitable for processing the continuously generated streaming data. The streaming data processing system has widely been studied and has already been implemented by several organizations [8], [9]. The implemented framework was mainly utilized to analyze web access [10], automotive driving [11], sensor data [12]. The processing technique can also be applied to the detection of network congestion, because a traffic log is typical time series data generated continuously from measurement equipment (e.g., routers, switches). In particular, we focus on a CEP which is a suitable technique for quickly analyzing a large amount of data [13], [14]. By analyzing a combination of multiple data streams or complex events with distributed computing resources, the CEP can extract a more effective event from them in real time. The CEP-based system can minimize the idle time of the computing resource, because each function of the CEP-based system can start to process next part of the streaming data without being blocked by the system just after transferring the processing results to the other functions. The network traffic is typical streaming data consisting of multiple events, and should be analyzed for detecting a sign of the network congestion. Therefore, by using the CEP, the network management system can rapidly and precisely detect the occurrence of the network congestion. In this research, we utilize an Oracle CEP [15] for building our proposed system, because the Oracle CEP is free for prototyping, and has been applied to various commercial systems.

Here, in the existing study [24], the CEP-based system for detecting traffic congestion, i.e., traffic jam, in the transport network has been proposed. The existing system has analyzed the location information, i.e., GPS, of each vehicle on the road, but the location of each packet cannot directly be captured on the Internet because the capture points of the packets can be deployed on the limited number of places. Therefore, the procedure of the existing system cannot be easily applied to the network congestion detection on the Internet, and a different approach should be selected for building the detection system.

Furthermore, in recent years, an web service that provides a virtual cloud computing environment such as Amazon EC2 (Amazon Elastic Compute Cloud) [16] provided by AWS (Amazon Web Services) [17] is attracting attention. By utilizing the Amazon EC2, computing resources (i.e., virtual machines) can be prepared on-demand for constructing a high performance distributed processing environment [18]. Considering these situations, we propose to use the Oracle CEP on the virtual machines of the Amazon EC2 so as to detect a sign of the network congestion in real-time.

3. Overview of Proposed CEP-based Congestion Detection System

Figure 1 shows an overview of our proposed CEP-based con-

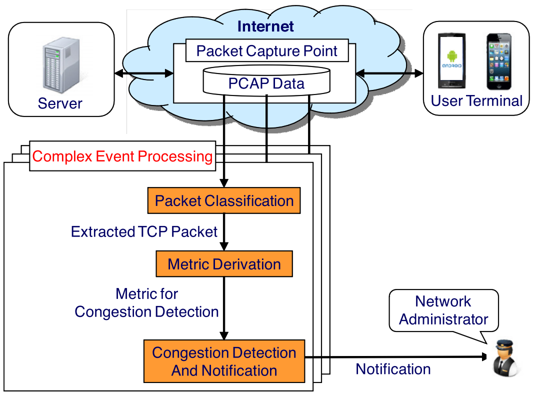


Fig. 1 Overview of proposed congestion detection system.

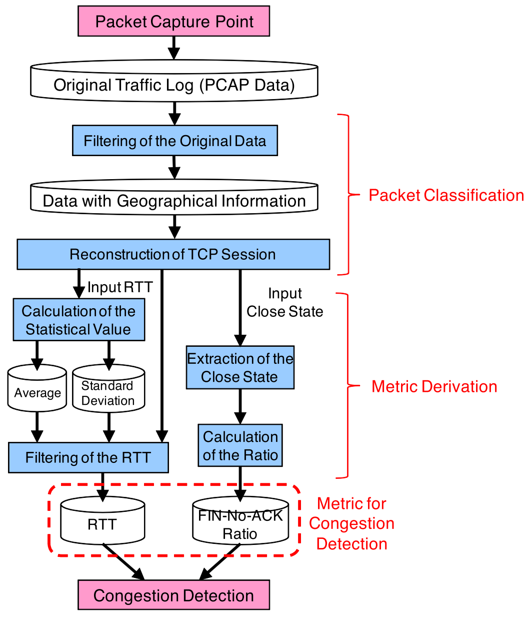

Packet Capture Point
Original Traffic Log (PCAP Data)
Filtering of the Original Data
Data with Geographical Information
Reconstruction of TCP Session
Packet Classification
Input RTT
Input Close State
Calculation of the Statistical Value
Average
Standard Deviation
Extraction of the Close State
Calculation of the Ratio
Filtering of the RTT
Metric Derivation
RTT
FIN-No-ACK Ratio
Metric for Congestion Detection
Congestion Detection

Fig. 2 Functional structure.

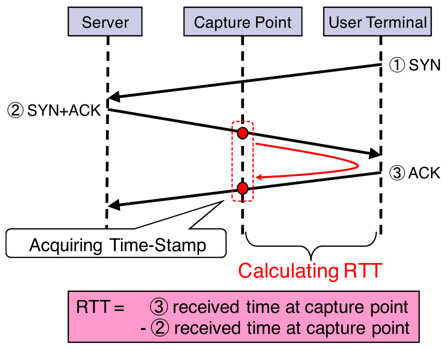


Fig. 3 Metric-1: RTT.

gestion detection system. As shown in this figure, the proposed system is composed of four functions.

The first function, Packet Capture Point, is usually deployed on a mobile access network, and continuously capture data traffics that are transmitted through a communication path between user terminals and a server on the Internet. The capture points that are prepared on the Internet send the traffic log to a cloud computing environment so that a large amount of traffic data can be processed by distributed servers using the CEP.

In the second function, Packet Classification, of the CEP-based system, the traffic log is analyzed so as to extract HTTP packets including information of the user's location from original captured data. The location information is recorded on HTTP requests by several types of web applications (e.g., navigation) that access a GPS function of the user terminal in order to specify the user's location. After that, the packet sequence of each TCP connection is extracted from the traffic log, and is divided into several groups based on the location information so that each group includes packets transferred in the same area. The third function, Metric Derivation, derives network performance metrics (e.g., RTT) in each area by analyzing the packets. Here, the metrics should be carefully selected so as to include a sign of the network congestion. Finally, the forth function, Congestion Detection and Notification, notifies a network administrator of the sign of network congestion when the congestion is detected in specific areas.

In the following sections, a detailed derivation procedure of several network performance metrics in the third function (Metric Derivation) is proposed, and appropriate metrics for the congestion detection are decided by analyzing actual traffic log that was captured in a specific area during a large-scale event (Section 4). Furthermore, we explain an implementation design of the CEP-based system that can rapidly calculate the metrics for congestion detection (Section 5).

4. Selection of Network Performance Metrics for Congestion Detection

4.1 Derivation Method of Network Performance Metrics

In our proposed system, we focus on two network performance metrics (Round-Trip Time (RTT) and State of TCP Session Termination) as analysis targets for congestion detection. **Figure 2** shows functional structure of the Metric Derivation function of the system. In the following part of this subsection, a derivation method of the two metrics is introduced in the detail. Furthermore, by analyzing a large amount of actual traffic data using the proposed derivation method, effectiveness of the metrics for network congestion detection is revealed in Section 4.2.

4.1.1 Derivation Procedure of RTT

As shown in **Fig. 3**, the proposed system derives an RTT when each TCP session is established over a target communication path. A packet capture point is deployed on the communication path between a server and a user terminal, and obtains a time-stamp when capturing "SYN+ACK" packet from the server and "ACK" packet from the user terminal. Here, the RTT between the capture point and the user terminal is derived by calculating a difference between these time-stamps, which can be defined as one of network performance metric of each TCP session. This is because the derived RTT indicates network performance of an access network where the communication resources are shared by many users concentrating on a specific and limited area (e.g., venue of a large-scale festival, disaster-stricken area).

Here, the network performance of the access network can be derived even when a server instead of a user terminal starts to establish the TCP session, but we consider only a condition where a TCP session is established by the user terminal as shown in Fig. 3. This is because the number of TCP sessions that are started by the server is very small compared with the number of sessions that are started by the user terminal.

Furthermore, the RTT is analyzed for detecting a sign of the network congestion as follows.

(1) An RTT of each TCP session is calculated when a TCP session is established.
(2) An average value and a standard deviation of RTT in each one minute are calculated. Here, we assume that a scattering pattern of the RTT follows a standard distribution, and only RTT values that do not exceed 99% confidence interval are extracted from all of them.
(3) An average of RTT values extracted in the previous step is calculated and utilized as a metric for congestion detection.

4.1.2 Derivation Procedure of State of TCP Session Termination

In the proposed system, we judge that a TCP session is terminated due to "timeout," when packets related with the session termination (i.e., "FIN" and "FIN+ACK") cannot be found in the packet sequence of the TCP session. Especially, when a user terminal does not respond after a server transmits a FIN packet, a state of the TCP session which is terminated due to the timeout is defined as "FIN-No-ACK." The "FIN-No-ACK" state represents a condition where a packet drops in a client-side network (i.e., mobile network) due to network congestion as shown in **Fig. 4**.

A network performance metric is derived by analyzing a state of the TCP session termination as explained in the following steps.

(1) The termination state of each TCP session is identified by analyzing the packet sequence (especially, "FIN" and "FIN+ACK" packets).
(2) Ratio of TCP sessions whose termination state is "FIN-No-ACK" to all of them is calculated in each one minute, and is utilized as a metric for congestion detection.

4.2 Evaluation of Network Performance Metrics

4.2.1 Target Traffic Log

In order to evaluate the feasibility of our proposed network performance metrics for congestion detection, we analyze a traffic log which was generated in Nagaoka-city, Niigata-prefecture, Japan during the Nagaoka Fireworks Festival. The Nagaoka Fireworks Festival is ranked as one of the three biggest Japanese firework festivals which is held in August 2 and 3, and more than 900,000 people visit the Nagaoka-city to attend the festival every year. Therefore, during the festival, a serious congestion is expected to occur in the Nagaoka-city. By analyzing the captured traffic data, we attempt to extract a sign of the network congestion.

The target dataset of the analysis was captured by a traffic capture point deployed on the border between a 3G mobile access network and the Internet in August 2 and 3, 2012 and 2013. Therefore, the target area of the network congestion detection has already been identified, hence the Packet Classification function of the proposed system is not considered in the following evaluation.

4.2.2 State of TCP Session Termination

Tables 1 and **2** show observation results of each state of TCP session termination in 2012 and 2013, respectively. In these tables, "first/second" in each row represents the first/second reason of the session termination.

"clt_fin/srv_fin" means a case that a server responds and transmits a FIN packet after a user terminal requests active close by sending a FIN packet, and then the TCP connection is normally closed. On the other hand, the state of "clt_rst/timeout" and "srv_rst/timeout" may be assumed as abnormal session termination, because a receiver of the RST packet did not transmit any packet when the TCP session was terminated. However, this is typical behavior of implementation of the TCP protocol, hence these states are categorized as normal session termination.

On the contrary, "timeout" indicates that any packet related with the session termination was not recorded on the traffic log. Therefore, it cannot be judged whether the TCP sessions could not correctly terminate due to network congestion or not, hence this state cannot be used as a metric for congestion detection.

As a result, "srv_fin/timeout" and "clt_fin/timeout" can be treated as candidate states of abnormal TCP session termination due to network congestion. **Figure 5** shows the packet sequence in a case that a user terminal starts to terminate the TCP session (i.e., clt_fin/timeout). As shown in this figure, in the clt_fin/timeout state of the session termination, packet losses occurred on an upstream network of the capture point. Therefore, only a "srv_fin/timeout" (highlighted in Tables 1 and 2) state (i.e., "FIN-No-ACK" explained in Section 4.1.2) can be utilized as a

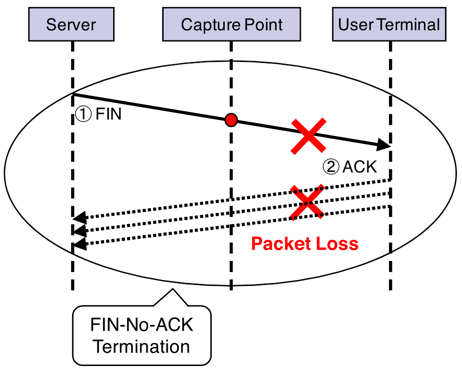


Fig. 4 Metric-2: FIN-No-ACK.

Table 1 Observations of each type of TCP session termination in 2012.

	The Number of Observed Data			The Number of Observed Data	
first/second	2012/8/1	2012/8/2	first/second	2012/8/1	2012/8/2
clt_fin/srv_fin	2287	2065	srv_fin/clt_fin	2719	2496
clt_fin/srv_rst	0	4	srv_fin/clt_rst	72	91
clt_fin/timeout	61	131	srv_fin/timeout	97	196
clt_rst/srv_fin	0	0	srv_rst/clt_fin	0	4
clt_rst/srv_rst	35	60	srv_rst/clt_rst	4	9
clt_rst/timeout	296	25	srv_rst/timeout	6	125
timeout	513	641			

Table 2 Observations of each type of TCP session termination in 2013.

	The Number of Observed Data			The Number of Observed Data	
first/second	2013/8/1	2013/8/2	first/second	2013/8/1	2013/8/2
clt_fin/srv_fin	2648	2297	srv_fin/clt_fin	2443	2713
clt_fin/srv_rst	8	6	srv_fin/clt_rst	28	138
clt_fin/timeout	156	128	srv_fin/timeout	205	237
clt_rst/srv_fin	0	3	srv_rst/clt_fin	1	1
clt_rst/srv_rst	10	9	srv_rst/clt_rst	1	2
clt_rst/timeout	63	109	srv_rst/timeout	9	49
timeout	1178	1718			

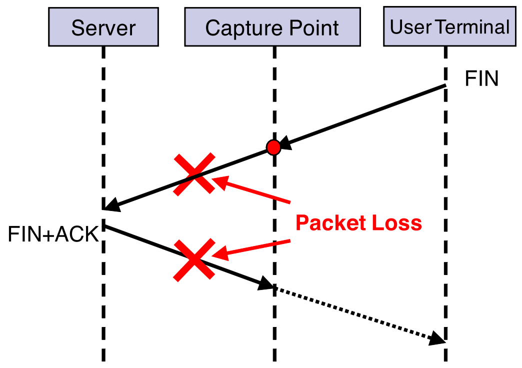


Fig. 5 FIN-No-ACK of client start-up.

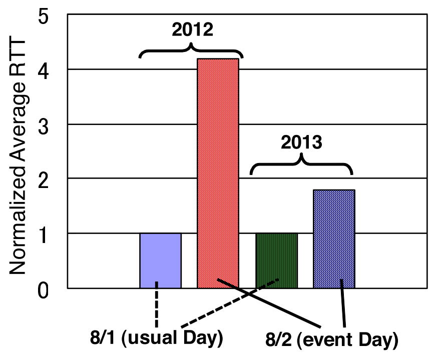


Fig. 6 Average RTT.

network performance metric.

Furthermore, in order to detect the network congestion, the metric should satisfy the following requirements.

(1) A large number of samples can be observed so as to detect occurrence of network congestion.

(2) When the network congestion occurs, the number of samples specifically changes from the stable state.

The network performance metric which does not satisfy the second condition is not suitable for the congestion detection because the occurrence of the congestion cannot be identified if the metric does not drastically change with time even when the network congestion occurs. The FIN-No-ACK state satisfies the first requirement because the number of samples is large compared with other states as shown in Tables 1 and 2. Therefore, in the next subsection, we clarify whether the FIN-No-ACK state satisfies the second requirement or not by analyzing an actual traffic log.

4.2.3 Effectiveness of Each Network Performance Metric

Figures 6 and **7** show an average RTT and a ratio of FIN-No-ACK state that were observed during the time interval from 19:30 to 21:00 in August 1 (usual day) and 2 (day of the festival), 2012 and 2013. In Fig. 6, the average RTT of each day is normalized by that of August 1 in each year. In addition, Fig. 7 illustrates a ratio of TCP sessions whose termination state is "FIN-No-ACK" to all of them. As shown in these figures, it is clarified that both metrics increase in the day of the festival in both years. In addition, tendency of the change of the network performance metric is different between 2012 and 2013, because the number of participants of the festival, the number of users of smart-phones/tablets, and network equipment of a cellular carrier may change. Especially, the number of users of the smart-phones/tablets has markedly increased in recent years, hence the network condition of the cellular network in 2013 has become worse than 2012. On the contrary, the cellular carrier in Japan is preparing the special cellular base station when the large-scale event is held in order to prevent the network congestion. As a result, the difference of the performance metric between the usual day and the event day becomes smaller in 2013. However, in both years, each metric becomes larger in the day of the festival than that in the usual day. Therefore, it can be concluded that these metrics are appropriate for detecting network congestion.

4.2.4 Time Variation of Each Network Performance Metric

Figures 8 and **9** show time variation of the average RTT and the ratio of FIN-No-ACK in 2012. Here, Fig. 8 depicts the average RTT in each one minute which is normalized by that in time interval from 19:30 to 21:00, and Fig. 9 depicts the ratio of FIN-No-ACK to all session termination in each one minute.

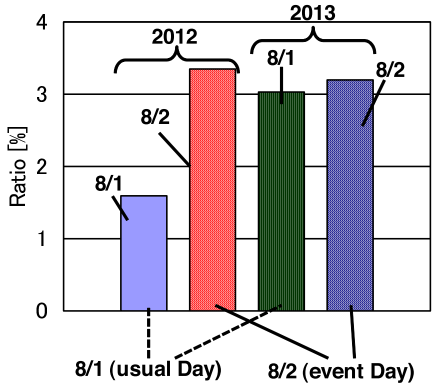


Fig. 7 Ratio of TCP session termination by FIN-No-ACK.

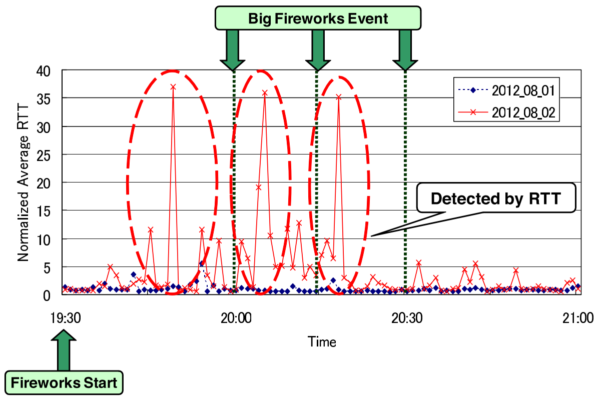


Fig. 8 Time variation of average RTT.

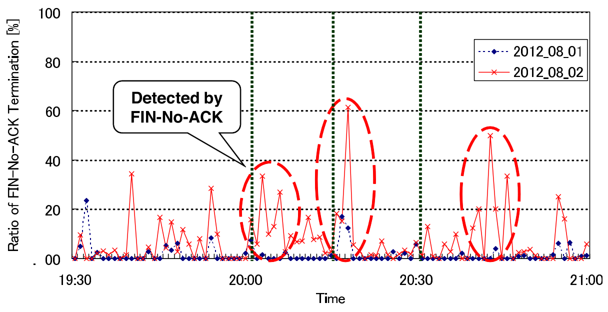


Fig. 9 Time variation of FIN-No-ACK ratio.

As shown in these figures, the network congestion was not detected on August 1 because Nagaoka Fireworks Festival was not held on this day. On the other hand, these metrics increased when remarkable big fireworks were set off on August 2. Time variation of these metrics is roughly synchronized, but the degree of the variation is much different between these metrics. This is because the impact of the network congestion on the packet transfer can appear in the different metric, i.e., some packets may wait for a long time in the base station, and others may be dropped on the congested wireless channel. If all packets can be captured and analyzed for deriving the metrics, the average values of the metrics may be highly synchronized. However, in the experiment, all packets that were generated in Nagaoka city could not be captured, because the capture point could not identify the generation site of the packets that did not include the GPS information.

The average RTT increased approximately 40 times, and the ratio of FIN-No-ACK termination increased 50% compared with the usual day. This result indicates that the communication traffic increased because many participants may send e-mails with pictures and/or use SNS for expressing impression of the fireworks to their friends or family.

Furthermore, before the large network congestion occurs, these network metrics start to increase as a sign of the congestion, and then keep larger values than the usual for a long time. Therefore, by specifying such an increase in the metric, the network congestion can be detected in advance.

As shown in Figs. 8 and 9, the network performance metrics largely increased before the drastic deterioration (part enclosed by the dashed line) due to the network congestion. Concretely, the RTT exceeded five times the average value in the usual, and the ratio of FIN-No-ACK became larger than 10% as a sign of the network congestion. Therefore, the appropriate threshold for the Nagaoka Fireworks Festival can be set to five times the average RTT in the usual, and can be set to 10% of the ratio of FIN-No-ACK. However, such an appropriate threshold for the network performance metric should be carefully selected for each large-scale event. Therefore, in the future study, we will analyze other dataset of a traffic log which is captured during other events in order to establish a general method of determining the threshold.

5. Design of CEP-based Congestion Detection System

In the previous sections, an overview of our proposed CEP-based congestion detection system has been introduced. Furthermore, we have selected appropriate network performance metrics that are utilized for detecting network congestion on mobile networks where many users concentrate when a large-scale event is held.

In this section, a distributed processing method of our proposed congestion detection system is designed so as to minimize processing time for extracting the network metrics from a large amount of traffic logs, and for specifying a sign of the network congestion. After that, the proposed system is implemented by using a popular commercial cloud computing service.

5.1 Distributed Processing Method

In order to minimize processing time of the congestion detection, a distributed computing method using a large number of computers is a reasonable solution. The distributed processing method can roughly be categorized to two types, "Vertical" and "Horizontal." **Figure 10** shows system structure of the distributed processing system. In the vertical distribution, each server executes a different function from others in order to achieve load balancing. On the other hand, in the horizontal distribution, the same function is allocated to multiple servers, and a large amount of data is processed by them simultaneously.

In the functional structure of the proposed congestion distribution system (See Section 3), the Congestion Detection and Notification function should be executed after all network performance metrics have been extracted from the traffic log by the Metric Derivation function. Therefore, these functions should be "vertically" distributed into servers. In addition, the Packet Classification function and the Metric Derivation function should be vertically distributed or should be processed in the same server, because the Metric Derivation function cannot be executed before the packet sequence of each TCP session is extracted from the traffic log by the Packet Classification function. On the other hand, a combination of the Packet Classification and Metric Derivation functions can be horizontally distributed, because each piece of the traffic log can be processed independently from others.

Therefore, the system structure of our proposed congestion detection system is composed of both of the vertical and horizontal distribution parts as shown in **Fig. 11**. As explained above, the workload of the Packet Classification is horizontally distributed into virtual servers by utilizing the load distribution mechanism of the Oracle CEP. Furthermore, the results of the Packet Classification are automatically transferred to one computing resource running the Metric Derivation using the process

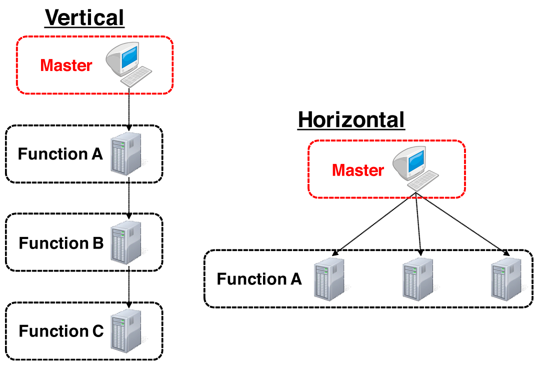


Fig. 10 Method of distributed processing.

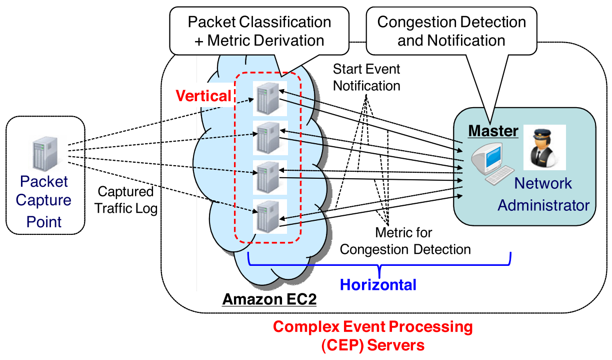

Packet Classification + Metric Derivation
Congestion Detection and Notification
Start Event Notification
Vertical
Master
Network Administrator
Packet Capture Point
Captured Traffic Log
Metric for Congestion Detection
Horizontal
Amazon EC2
Complex Event Processing (CEP) Servers

Fig. 11 System structure and deployment of CEP-based system.

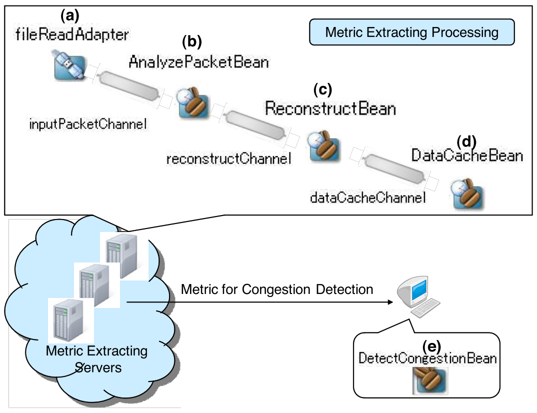


Fig. 12 Block diagram of CEP-based system.

distribution mechanism of the CEP-based system. The combination of the load/process distribution mechanisms for detecting network congestion is the main originality of our proposed system.

5.2 Functional Structure of CEP-based System

As mentioned in Section 5.1, the horizontal distributed processing method is applied to the combination of the Packet Classification and Metric Derivation functions. In the horizontal distribution part, the two functions should be processed sequentially for deriving the network performance metrics. Here, in order to minimize a processing time, we want to avoid a condition where one function gets in an idle state while another function is processing the data. Therefore, in our proposed congestion detection system, a CEP technology is adopted for implementing the distributed processing method. The CEP has been released as an easy-to-use software package that can construct a distributed environment for extracting important events by analyzing a large amount of streaming data. Furthermore, the CEP-based system has a unique characteristic for minimizing the idle time of each function as explained in Section 2.

Figure 12 shows a block diagram of CEP-based implementation of our proposed congestion detection system. Each functional component in this figure is an event driven, hence it is invoked when receiving an event from others. A detailed procedure of the CEP-based congestion detection system is explained as follows.

(1) A "fileReadAdapter" component reads a PCAP (Packet CAPture) file consisting of a traffic log which is captured on a packet capture point. The traffic log is divided into packets, and is transmitted to a next component.

(2) An "AnalyzePacketBean" component extracts information which is needed to identify the packet sequence of each TCP session, and sends that to a next component.

(3) A "ReconstructBean" component reconstructs the packet sequence of each TCP session, and attempts to extract a state of TCP session termination by analyzing the sequence. The state is transmitted to a next component together with the time when the session termination was observed.

(4) A "DataCacheBean" component gathers information required for deriving network performance metrics in each one minute, and derives network performance metrics using the derivation method explained in Section 4.1. The network metrics are transmitted to a next component.

(5) Finally, a "DetectCongestionBean" component detects the occurrence of network congestion by analyzing the dynamic time change of the network performance metrics.

In the Oracle CEP, a special query language, named Oracle Continuous Query Language (Oracle CQL), is used to express queries on data streams to perform a complex event processing in each component using Oracle CEP. However, the proposed system does not treat complex queries that the CQL is necessary to express but basically processes only simple queries, i.e., one kind of input stream is processed in the component (Java program), and the output of the component is simply transferred to the output stream as explained above.

In this study, our CEP-based congestion detection system analyzes only the PCAP file captured for deriving the network performance metric of each TCP session. This is because we have focused on the detection of the network congestion which affects the end-to-end quality of mobile users. The procedure of the CEP-based system can be enhanced to accommodate other measurement results (e.g., ping, SNMP, etc.) for detecting other kinds of congestion (e.g., failure of the equipment in the core network) in the future study.

5.3 Deployment on Amazon EC2

As shown in Fig. 11, functional components of our proposed CEP-based congestion detection system are deployed on virtual servers in a cloud computing environment. By utilizing the cloud computing service, a large amount of traffic logs can efficiently be obtained from multiple packet capture points deployed on the Internet.

In this study, we select an Amazon Elastic Compute Cloud (Amazon EC2) as the cloud computing environment, because the setup of the evaluation environment (e.g., the number of virtual servers) can easily and dynamically be changed on the Amazon EC2. **Table 3** shows specifications of each virtual machine utilized for constructing the evaluation environment [19]. Furthermore, an Oracle CEP is utilized for building our proposed CEP-based system, because the Oracle CEP is free for prototyping, and has been applied to various commercial systems as explained in Section 2.

Table 3 Evaluation environment.

Virtual Server	Amazon EC2 (Small Instance)
OS	Ubuntu 12.04 LTS
CPU	Equivalent of 1 GHz
Memory	1.7 GiB

Through the experimental evaluation on the environment, we disclose the feasibility of our proposed CEP-based system deployed on the cloud computing environment.

6. Experimental Evaluation

6.1 Evaluation Result of Distributed Processing Method

In order to clarify the feasibility of our proposed distributed processing method using CEP, we evaluate the processing time of our proposed system deployed on the cloud computing envi-

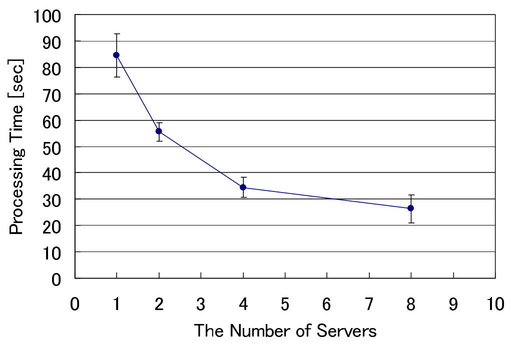
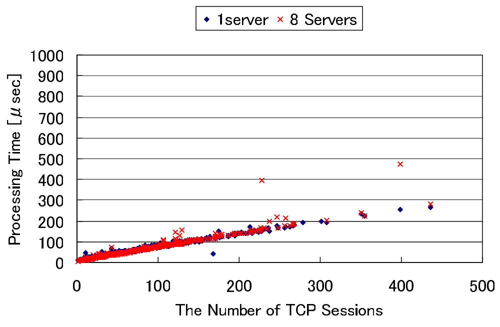
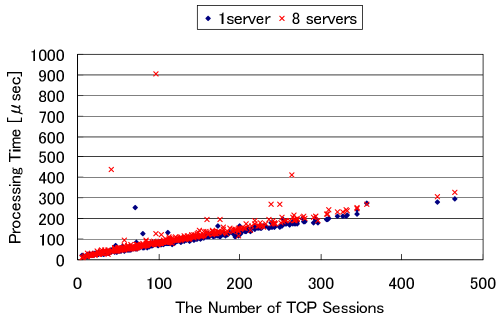


Fig. 13 Relationship between number of virtual servers and processing time.



(a) Processing Time: RTT



(b) Processing Time: FIN-No-ACK

Fig. 14 Relationship between the number of TCP sessions and processing time.

ronment. The specification of the experimental environment is explained in Section 5.3.

Figure 13 shows the relationship between the processing time and the number of virtual servers where the horizontal distribution part of the proposed system (See Section 5.1) is deployed. In this figure, the processing time is the time interval needed for extracting the network performance metrics (i.e., RTT, state of TCP session termination) from the traffic log, and the error bar represents one standard deviation of the processing time. Here, the traffic log includes information about 1,740,000 packets that were captured on 2nd August, 2013.

As shown in Fig. 13, with the increase in the number of virtual servers, the proposed distributed processing method improves the processing time, and the processing time decreases to 26.3 [sec] when eight servers are utilized for the distributed processing. Of course, the processing time cannot equally be divided by the number of virtual servers, because the fixed amount of processing load occurred on each server (e.g., invocation time of each process). In addition, even if the same amount of workload is dispatched to the servers, the processing time largely fluctuates due to the existence of the virtual server with the lower computation performance than others. Although there are the above-mentioned limitations, the processing time markedly decreases with the increase in the number of the servers. As a result, it can be concluded that the larger amount of traffic log can be analyzed in each unit time as the more computing resources are utilized for the distributed processing.

Next, in order to evaluate feasibility of the vertical distribution part, **Fig. 14** shows the relationship between the processing time of each metric and the number of TCP sessions that the Metric Derivation function has analyzed. As shown in this figure, the processing time does not change even when the number of virtual servers increase. The Congestion Detection function should group a set of extracted metrics that were captured in each unit time interval, and then should process them at one time because the derivation process of the statistics of the metrics (i.e., average RTT, ratio of FIN-No-ACK) in the unit time cannot be divided to several parts. Therefore, the process of the function cannot horizontally be distributed, and the processing time of the metrics (RTT and FIN-No-ACK) is simply proportional to the number of the target TCP sessions. However, the congestion detection can be completed within 1 [msec] even when a large number of TCP sessions (more than 400) is processed at a time, hence the pro-

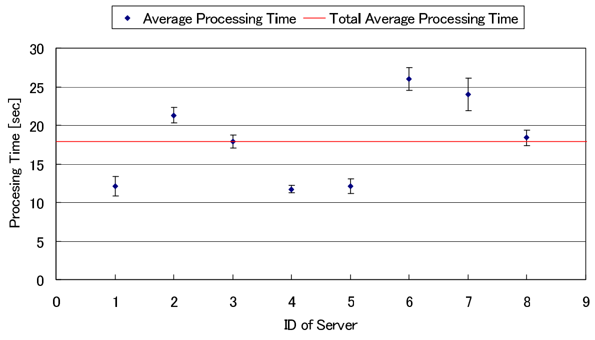


Fig. 15 Average processing time of each server.

cessing time is much shorter than that of deriving the network performance metrics. As a result, it can be concluded that our proposed system can detect network congestion by analyzing a large amount of traffic logs in real time.

Finally, **Fig. 15** shows the average processing time of each server when the congestion detection process is executed by eight virtual servers ten times, and the error bar represents the standard deviation. As shown in this figure, the computing performance of virtual servers is much different between each other in the Amazon EC2, hence the difference of the processing time becomes longer than 10 [sec]. Furthermore, it takes 26 [sec] for all servers to finish analyzing the traffic log while the average processing time of all servers is 17.9 [sec]. Therefore, if a virtual server which has worse performance than others is selected for the distributed processing, the performance of our proposed system markedly degrades. In order to solve the problem, we will consider an optimal load balancing method between the virtual servers in the future study. For example, the amount of the traffic log which is allocated to each virtual server should be carefully decided so as to finish the computation by almost the same time among the virtual servers. Especially, the optimization of scheduling workload in each server becomes an important issue especially when the number of capture points is large, because each capture point generates the traffic log of a different size from others and the size of the traffic-log varies with time. Furthermore, the performance of the virtual server cannot be accurately estimated because the computing resource of the virtual server should be shared with other users. Therefore, a use of real servers whose computing resources can be occupied for the CEP-based system may be a reasonable solution for preventing the fluctuation of the processing time for a mission critical task.

6.2 Capability of Congested Area Estimation

An objective of our proposed system is to estimate an area where the network congestion occurs. An appropriate congested area estimation method depends on the deployment policy of the packet capture points (i.e., location/the number of capture points). In the following part, we consider capability of the estimation method for each case of the deployment policy.

6.2.1 Case 1: Capture Point is Deployed in Each Area

If the packet capture point can be deployed in the access network of each target area of the congestion detection, the congested area can easily be estimated. This is because, location information of the capture point can be recorded on the captured packets when the packets are obtained by the packet capture point, which is very useful for dividing packets related with the same area into the same group. If the occurrence of the network congestion is detected by analyzing the captured packets in a certain group, the congested area can be uniquely estimated.

Here, when the user terminal moves to the different place, the packets generated from the terminal are captured by the different capture point, hence are categorized to the different group. Therefore, in this case, the movement of each user terminal does not have to be completely traced.

By utilizing a cloud computing environment as the same as our proposed system, a large amount of traffic logs can directly be transmitted from many capture points to the cloud computing environment, and can be analyzed by many virtual servers. However, the equipment for constructing the packet capture point should be prepared in various places in this case.

6.2.2 Case 2: Capture Point is Deployed in Limited Points on the Internet

If the packet capture point can be deployed in only few points (i.e., the border between the core and access networks), the captured traffic log includes a large number of packets generated in various areas. Therefore, in order to identify the location where the packet is generated, a reverse-geocoding technology that estimates address information corresponding to the GPS information is an appropriate solution. In recent years, many web services (e.g., map, navigation) using the reverse-geocoding have already been released [20], and the reverse-geocoding API has been made available for implementing such a service [21].

As explained in Section 3, our proposed system extracts HTTP packets including GPS information from the captured traffic log, and the information is analyzed by using the reverse-geocoding API in order to identify the user's location. By extracting only packets that are related with the target location of congestion detection system by collaborating with the reverse-geocoding API, the processing load that is needed for deriving network metrics and for detecting network congestion can be mitigated.

In this case, when the user terminal moves to the different place, the accurate location of the terminal cannot be identified until the terminal sends the packet including the GPS information. Therefore, the accuracy of the congestion detection slightly deteriorates compared with the Case 1. In order to mitigate the performance deterioration of the congestion detection, we will consider a new tracking method of the user terminal using not only GPS information in the HTTP packets but also other identification information in the future study.

As explained in Section 4.2.1, the dataset including packets, whose generated location has already been identified as the Nagaoka-city, was analyzed in the experimental evaluation, hence the reverse-geocoding and packet extraction functions have not been implemented in this study.

In an actual congestion detection system, extension of these functions yields additional processing time. However, the processing load for the additional functions can horizontally be distributed into multiple virtual servers as the same as the functions evaluated in Section 6.1. Therefore, in the future study, we will assume a condition where the packet capture points can be deployed in limited points on the Internet (i.e., Case 2), and will implement a horizontal distributed processing method for the reverse-geocoding and packet extraction functions.

7. Conclusions

In this study, we have proposed a new system structure of a CEP-based network congestion detection system using distributed computing resources on a virtual cloud computing service. First, in order to determine network performance metrics that can be used to estimate a sign of the network congestion, we have analyzed a large amount of traffic logs which were captured on a mobile network when a large-scale event, the Nagaoka Fireworks Festival, was held. From the analytical results, it has been revealed that an average RTT and a specific state of TCP session termination (i.e., FIN-No-ACK event) are appropriate for detecting the network congestion.

In addition, in order to minimize the processing time of the congestion detection, a distributed computing method using a large number of computing resources has been proposed. In the proposed method, a function for deriving the network metrics by analyzing a large amount of traffic logs has horizontally been distributed and allocated to multiple servers. Furthermore, a CEP technology has been used to implement the proposed method in the cloud computing environment so as to minimize the idle time of the computing resources. Through experimental evaluation using the traffic log captured during the Nagaoka Fireworks Festival, it has been clarified that our proposed system can reduce the processing time by 70% when eight servers are utilized for the distributed processing.

In the future study, we will consider the feasibility of other network metrics (e.g., packet loss rate) for network congestion detection, and will develop a general method that can be used to judge whether network congestion occurs or not based on the extracted network metrics. Furthermore, an optimal load balancing method between virtual servers will be proposed in order to further reduce the processing time.

References

[1] Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems (May 2002).
[2] Nagaoka Fireworks Festival, available from ⟨http://nagaokamatsuri.com/index.html⟩ (accessed 2014-08-14).
[3] Takahashi, T., Yamamoto, H., Fukumoto, N., Ano, S. and Yamazaki, K.: Congestion Detection in Mobile Network towards Complex Event Processing, *Proc. IEEE Annual International Computers, Software & Applications Conference 2013*, pp.459–462 (July 2013).
[4] Thottan, M. and Ji, C.: Anomaly detection in IP networks, *IEEE Trans. Signal Processing*, Vol.51, No.8, pp.2191–2204 (2003).
[5] Kihara, T., Tateishi, N. and Seto, S.: A Study on a Fault Detection Method with Relation Analysis of Network Data, *IEICE Technical Report, Technical Committee on Information and Communication Management*, Vol.110, No.466, pp.17–22 (Mar. 2011) (in Japanese).
[6] News Releases from NTT docomo, available from ⟨http://www.nttdocomo.co.jp/info/news_release/2011/06/14_00.html⟩ (accessed 2014-08-14).
[7] News Releases from KDDI, available from ⟨http://www.kddi.com/corporate/news_release/2013/0610a/⟩ (accessed 2014-08-14).
[8] Apache S4, available from ⟨http://incubator.apache.org/s4/⟩ (accessed 2014-08-14).
[9] Strom, available from ⟨http://storm-project.net/⟩ (accessed 2014-08-14).
[10] Imai, T., Ebiyama, T., Kida, K., Fujiyama, K. and Nakamura, N.: A Web Access Analysis System Using Data-Stream Processing Technique, *Forum on Information Technology*, Vol.8, No.2, pp.207–208 (Aug. 2009).
[11] Kida, K., Fujiyama, K., Imai, T. and Nakamura, N.: Development and Evaluation of High Performance Floating Car Data System Based on Data-stream Processing, *IEICE Technical Report, Intelligent Transport Systems*, Vol.108, No.200, pp.1–8 (Sep. 2008).
[12] Sato, K.: Sensor Data Processing System for Automotive Driving Environment Recognition, *IEICE Technical Report, Data Engineering*, Vol.110, No.107, pp.51–56 (Oct. 2010).
[13] Kuwata, S., Inaba, Y., Yokogawa, M., Namatame, T. and Nakagawa, K.: Stream Data Analysis Application for Customer Behavior with Complex Event Processing, *IEICE Technical Report, Data Engineering*, Vol.110, No.107, pp.13–18 (June 2010).
[14] Tanaka, T.: Evidence-Based Execution Realized by Algorithmic Trading, *Journal of the Japanese Society for Artificial Intelligence*, Vol.24, No.3, pp.376–384 (May 2009).
[15] Oracle CEP, available from ⟨http://www.oracle.com/technetwork/middleware/complex-event-processing/overview/index.html⟩ (accessed 2014-08-14).
[16] Amazon Elastic Compute Cloud, available from ⟨http://aws.amazon.com/ec2⟩ (accessed 2014-08-14).
[17] Amazon Web Service, available from ⟨http://aws.amazon.com/⟩ (accessed 2014-08-14).
[18] Toyoshima, S., Yamaguchi, S. and Oguchi, M.: Storage Access Optimization with Virtual Machine Migration During Execution of Parallel Data Processing on a Virtual Machine PC Cluster, *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops* (*WAINA*), pp.177–182 (April 2010).
[19] AWS documentation - Amazon EC2 User Guide: Instance Type, available from ⟨http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html⟩ (accessed 2014-08-14).
[20] Google Map, available from ⟨https://maps.google.co.jp⟩ (accessed 2014-08-14).
[21] Google Geocoding API, available from ⟨https://developers.google.com/maps/documentation/geocoding/⟩ (accessed 2014-08-14).
[22] Dean, J. and Ghemawat, S.: MapReduce: Simplified data processing on large clusters, *Comm. ACM*, Vol.51 No.1, pp.107–113 (Jan. 2008).
[23] White, T.: *Hadoop: The Definitive Guide*, Oreilly & Associates Inc. (May 2012).
[24] Terroso-Saenz, F. et al.: A Cooperative Approach to Traffic Congestion Detection With Complex Event Processing and VANET, *IEEE Trans. Intelligent Transportation Systems*, Vol.13, No.2, pp.914–929 (2012).

Hiroshi Yamamoto received his M.E. and D.E. degrees from Kyushu Institute of Technology, Iizuka, Japan in 2003 and 2006. He worked at FUJITSU LABORATORIES LTD., Kawasaki, Japan from April 2006 to March 2010. Since April 2010, he has been an Assistant Professor in the Department of Electrical Engineering at Nagaoka University of Technology. His research interests include computer networks, distributed applications, and networked services. He is a member of IEICE and IEEE.

Tatsuya Takahashi received his B.E. degree from Nagaoka University of Technology in 2012. He is currently a graduate school student in Nagaoka University of Technology. His research interests include computer networks and network management.

Norihiro Fukumoto received his B.E. and M.E. degrees in Information and Computer Science from Waseda University, Tokyo in 1999 and 2001. He joined KDDI R&D Laboratories, Inc. in 2001 and has been engaged in research and development on speech application services, voice packetization system over IP networks, and QoS/QoE Management. He is currently a research engineer at the Communications Network Planning Laboratory of KDDI R&D Laboratories, Inc.

Shigehiro Ano received his B.E., M.E. and D.E. degrees in electronics and communication engineering from Waseda University, Japan in 1987, 1989 and 2015, respectively. Since joining KDD in 1989, he has been engaged in the field of ATM switching system and ATM networking. His current research interests are traffic routing, control and management schemes over the next generation IP networks. He is currently an Executive Director of Network Operation and Administration Division in KDDI R&D Laboratories Inc. He received IPSJ Convention Award in 1995 and IEICE Communications Society Best Paper Award in 2010 and 2012, respectively.

Katsuyuki Yamazaki received his B.E. and D.E degrees from the University of Electro-communications and Kyushu Institute of Technology in 1980 and 2001. At KDD Co. Ltd., he had been engaged in the R&D and international standardization of ISDN, S.S. No.7, ATM networks, L2 networks, IP networks, mobile and ubiquitous networks, etc., and was responsible for the R&D strategy of KDDI R&D Labs. He is currently a Professor at Nagaoka University of Technology.

© 2015 Information Processing Society of Japan