Master's Thesis

Title

Hierarchically Structured Spatio-Temporal Traffic Measurement utilizing Compressive Sensing

Supervisor

Professor Masayuki Murata

Author

Yoshihiro Tsuji

February 10th, 2015

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Hierarchically Structured Spatio-Temporal Traffic Measurement utilizing Compressive Sensing

Yoshihiro Tsuji

Abstract

The collection of traffic information at the short-term interval is required to perform the TE at the short-term interval. However, the collection of traffic information at the short term interval causes a large overhead; a large number of traffic information from a large number of nodes are sent to the control server periodically at the short interval. As a result, especially links near the control server may be congested. However, approaches against this problem have not been sufficiently discussed.

In this thesis, we propose a method to collect traffic information without causing a large overhead to the network. In our method, the traffic monitor creates the traffic model for each flow to be monitored, and share it with the control server. Then, unless the model becomes no longer suitable to the current traffic, the traffic monitor sends only the information on the difference from the models, which we call the differential traffic information. If we can create the accurate traffic model, most of the elements in the differential traffic information are small. Thus, the traffic monitor compresses the differential traffic data by using the compressive sensing, which is a technique to compress the sparse data and recover it from the compressed data.

We introduce a method to compress the differential traffic information along the route from traffic monitors to the controller by using the compressive sensing. In this method, we construct a tree whose root is the controller. The differential traffic information is sent along the tree from the leaves to the root. Each traffic monitor compresses the differential traffic information generated by it and those sent from its children into a few number of packets. Thus, this approach avoids the concentration of traffic information on the links near the controller.

In this thesis, we evaluate our method by numerical simulation. The results demonstrate that we can reduce the amount of the information required to be collected by half without causing the errors larger than 1 [Mbps], if the number of flow entries which cannot be predicted accurately is smaller than a quarter of the number of the total entries.

Keywords

Traffic Measurement

Traffic Data Collection

Compressive Sensing

Traffic Modeling

Contents

1 Introduction **5**

2 Related Work **7**

2.1 Traffic Engineering 7

2.2 Traffic Measurement 7

2.3 Traffic Prediction 8

3 Traffic Measurement utilizing Compressive Sensing **10**

3.1 Compressive Sensing 10

3.2 Compression and Recovery 10

3.2.1 Quantization 11

3.2.2 Compression 11

3.2.3 Recovery 11

3.2.4 Reverse Quantization 12

4 Hiearchically Structured Spatio-Temporal Traffic Measurement **14**

4.1 Overview 14

4.2 Hierarchical Traffic Data Collection 14

4.3 Updating Traffic Model 15

5 Evaluation **18**

5.1 Evaluation Environment 18

5.2 Number of non-Zero Entries after Quantization 18

5.3 Errors caused by Compression 19

5.4 Overall Errors 19

5.5 Discussion on Intervals to Update Traffic Models 20

6 Conclusion **30**

Acknowledgments **31**

References **32**

List of Figures

1 Coding the traffic information 13
2 Overview of collection of traffic information 16
3 Flowchart to update the traffic model 17
4 Number of non-zero entries 21
5 Error caused by CS ($|y| = 100$, compression ratio is 1) 22
6 Error caused by CS ($|y| = 75$, compression ratio is 1.33) 23
7 Error caused by CS ($|y| = 50$, compression ratio is 2) 24
8 Error caused by CS ($|y| = 25$, compression ratio is 4) 25
9 Overall error ($|y| = 100$, compression ratio is 1) 26
10 Overall error ($|y| = 75$, compression ratio is 1.333) 27
11 Overall error ($|y| = 50$, compression ratio is 2) 28
12 Overall error ($|y| = 25$, compression ratio is 4) 29

1 Introduction

The network operators must accommodate the fluctuating traffic without causing congestion. So far, network operators prepare the redundant link capacity so as to accommodate the possible traffic. However, this approach requires high cost according as the variation of traffic increases.

One approach to accommodating fluctuating traffic without a large cost is to dynamically change the routes so as to follow the traffic changes. This approach is called traffic engineering (TE), and many methods have been proposed [1, 2]. In these methods, the network controller is deployed. The network controller collects the traffic information from the nodes within the network, and then calculates the suitable routes based on the collected traffic information.

In the TE, the traffic information is an important input. The existing TE methods use the traffic measurement technologies such as Netflow [3] and sFlow [4]. In NetFlow, traffic monitors have traffic counters per flow and increment their counters when a packet arrives, and then, the control server periodically collects flow-statistics from all traffic monitors in the network and obtains the traffic information. On the other hand, in sFlow, traffic monitors summarize sampled packet information and send directly to the control server. Then the control server obtains the traffic information by the calculation based on sampled packet information.

Most of TE methods focus on the daily traffic changes, and configure the routes periodically at the intervals of a few hours. However, the route control at the hourly intervals may be insufficient when the traffic fluctuation is significantly large. For example, Benson et al illustrated that the route control at the intervals of a few seconds is required to provide sufficient bandwidth to communicating nodes in the data center network [5].

The collection of traffic information at the short-term interval is required to perform the TE at the short-term interval. However, the collection of traffic information at the short term interval causes a large overhead; a large number of traffic information from a large number of nodes are sent to the control server periodically at the short interval. As a result, especially links near the control server may be congested. However, approaches against this problem have not been sufficiently discussed.

In this thesis, we propose a method to collect traffic information without causing a large overhead to the network. In our method, the traffic monitor creates the traffic model for each flow to be monitored, and share it with the control server. Then, unless the model becomes no longer

suitable to the current traffic, the traffic monitor sends only the information on the difference from the models. Hereafter we call the information on the difference from the models *differential traffic information*. Then, the control server obtains the traffic data by adding the differential traffic information to the traffic rate generated by the traffic model.

If we can create the accurate traffic model, most of the elements in the differential traffic information are small. Thus, the traffic monitor compresses the differential traffic data by using the compressive sensing (CS), which is a technique to compress the sparse data or recover it from the compressed data.

We introduce a method to compress the differential traffic information along the route from traffic monitors to the controller by using the compressive sensing. In this method, we construct a tree whose root is the controller. The differential traffic information is sent along the tree from the leaves to the root. Each traffic monitor compresses the differential traffic information generated by it and those sent from its children into a few number of packets. Thus, this approach avoids the concentration of traffic information on the links near the controller.

In this thesis, we evaluate our method by numerical simulation. The results demonstrate that we can reduce the amount of the information required to be collected by half without causing the errors larger than 1 [Mbps], if the number of flow entries which cannot be predicted accurately is smaller than a quarter of the number of the total entries.

The rest of this thesis is organized as follows. First, Section 2 explains the related work, especially focused on the TE, traffic measurement and traffic prediction. Section 3 introduces the method to compress and recovery the traffic information based on compressive sensing. Section 4 presents the method of hierarchical collection and updating traffic models in our traffic monitoring. Section 5 evaluates our method by numerical simulation. Section 6 concludes this thesis.

2 Related Work

2.1 Traffic Engineering

Traffic engineering (TE) has been studied as one approach to accommodating the traffic changes by dynamic changing routes. The process of TE method consists of following three steps; (1) the collection of traffic information, (2) the calculation of the routes so as to accommodate the traffic, and (3) the implementation of calculated routes into the actual network.

Most of them focused on the daily traffic changes, and configures the routes periodically at the intervals of a few hours [1, 2]. However, the recent growth of the Internet services such as streaming and clouds has enlarged the time variation of the Internet traffic. In addition, the growth of Machine-to-Machine communication will accelerate the increase of the traffic variation. As the time variation of the network traffic becomes large, more short-term control is required. In fact, Benson *et al.* illustrated that dynamical route changes at the interval of a few seconds are required to provide sufficient bandwidth in the data center network [5]. To perform TE at such short-term intervals, the control server must collect traffic information at the short-term intervals, which may cause a large overhead.

2.2 Traffic Measurement

Traffic information is required to manage the network. Especially, traffic matrices (TMs), which are the matrices whose elements are traffic rates between all nodes in network, are important for the network management. TM gives important input for many applications such as future traffic prediction, network optimization, protocol design, anomaly detection, and dynamic routing control and so on.

TM is obtained by the following three steps; (1) each traffic monitor monitors the traffic per flow, (2) it forwards the monitored traffic information to the control server, and (3) the control server analyzes the received traffic information. Major traffic monitoring technologies, which are widely used, are NetFlow [3] and sFlow [4]. In NetFlow, traffic monitors prepare traffic counters per and increment their counters when a packet arrives, and then, the control server periodically collects flow-statistics from all traffic monitors in network. On the other hand, in sFlow, traffic monitors summarize sampled packet information and send directly to the control server, and then, the control server obtains the traffic information based on sampled packet information.

However, as a traffic rate per each flow becomes large, the overhead of these traffic monitoring methods may be a problem; the traffic monitor requires more CPU resources to maintain the traffic counter, and a large number of samples may be forwarded to the control server. So far, this problem has been addressed by setting the sampling rate to a small value, or by monitoring traffic only within the predefined small range of the target traffic to be monitored. However, these approaches may cause the large monitoring errors.

Therefore, several approaches to mitigating the overhead of the traffic monitoring have been proposed [6–8]. L. Yuan *et al.* have proposed a programmable measurement architecture [6], where the algorithms to monitor traffic can be implemented. While G. Cormode *et al.* have proposed the method to identify the hierarchical heavy hitters in multi-dimension [7], L. Jose *et al.* also implemented the method to identify the hierarchical heavy hitters using commodity switchs [8]. They focused only on the overhead of the traffic monitors. The overhead of the collection of the traffic information has not been sufficiently discussed.

However, the overhead of the collection of the traffic information is a non-negligible problem especially for the manager of a large network. As the network becomes large, a large number of traffic information concentrate on the links near the control server. Moreover, as the control server performs the control of the network at the short-term intervals, the intervals to collect traffic information become smaller, which also increase the rate of the traffic information sent by each traffic monitor. Therefore, in this thesis, we focus on the problem of the overhead of the collection of the traffic information.

The final steps to obtain the traffic information may also require the large CPU resources, as the number of collected traffic information becomes large. Several approaches to mitigate the overhead caused by analyzing the traffic information have been proposed [9]. Y. Lee *et al.* has developed the Internet traffic measurement and analysis system on the parallel distributed computing architecture Hadoop. Therefore, if we can collect the traffic information with a small overhead, we can obtain the traffic information immediately by using these approaches to analyze the received information.

2.3 Traffic Prediction

The predictability of the Internet traffic has paid much attention from many domains like capacity planning, anomaly detection, admission control, and TE. To predict the traffic in network, many

prediction models have been proposed such as ARMA, ARIMA [10], ARCH [11], GARCH [12], Neural Network [13] and so on. In these methods, traffic is predicted by the two phases, learning phase and prediction phase. In the learning phase, the traffic data of past time slots is analyzed to tune the parameters of the traffic model. Then, in the prediction phase, the future traffic rate is predicted by calculating the traffic model whose parameters are tuned in the learning phase to learn real-time network condition.

In our method, the traffic monitor creates the traffic model for each flow to be monitored, and share it with the control server. Then, unless the model becomes no longer suitable to the current traffic, the traffic monitor sends only the differential traffic information from the models. If we can create the accurate traffic model, most of the elements in the differential traffic information are small. Thus, the traffic monitor compresses the differential traffic data by using the compressive sensing, where the sparsity of the data determines its performance.

3 Traffic Measurement utilizing Compressive Sensing

3.1 Compressive Sensing

In this thesis, we apply the compressive sensing (CS) to traffic measurement. The CS reduces the size of the data by using its redundancy. In the CS, we only measure the signal $y \in R^M$, whose size is smaller than the original signal $x \in R^N$. y and x have the relation of $y = Ax$, where A is the matrix called the measurement matrix. That is, in the CS, the original signal x should be estimated from measurement signal y such that $y = Ax$. This is a ill-posed problem, because the number of simultaneous equations is smaller than the number of variables. However if the coherence of the measurement matrix is sufficiently small and the original signal is sparse, the original signal can be recovered with slight errors [14]. In this thesis, we use the CS to compress the traffic measurement. Though using the CS to compress the traffic measurement causes the errors, the errors included in the traffic measurement are acceptable, unless the errors do not degrade the performance of the application using the traffic measurement. For example, the TE can set the suitable routes even when the relative errors included in the traffic matrix are 20 % [15].

Figure 1 shows the steps to encode or decode the traffic measurement based on the CS. First, we create the traffic model. If we can create the accurate traffic model, most of the elements in the differential traffic information are small. As long as the traffic monitor and the control server share the model in advance, all we have to collect is the differential traffic information. By applying the CS to these data, we collect the differential traffic information of all nodes with a few packets.

3.2 Compression and Recovery

The traffic monitor creates the traffic model for each flow to be monitored, and shares it with the control server. The traffic monitor sends only the information on the differential traffic information. By using the information, the control server collects the traffic information. Here, if we can create the accurate traffic model, most of the entries in the differential traffic information are quite small. The differential traffic information can be sent with compressed by using the CS. Hereafter in this section, we describe the method to compress and recovery the traffic information.

3.2.1 Quantization

First, the traffic monitor quantizes the differential traffic information by the predefined quantization size. The quantized differential traffic information x_f corresponding to the flow f is obtained by Eq. (1).

$$x_f = \text{trunc}\left(\frac{\epsilon_f}{\rho}\right) \tag{1}$$

where ϵ_f is the difference obtained by Eq. (2), ρ is the quantization size, and trunc (a) is the same signed integer that does not exceed $|a|$. For example, trunc $(0.4) = 0$, trunc $(-1.2) = -1$.

$$\epsilon_f = t_f - z_f, \tag{2}$$

where t_f is the actual traffic information, and z_f is the predicted traffic rate generated by the traffic model. By this quantization in Eq. (1), most of the elements in the vector $x = (x_{f_1}, x_{f_2}, ..., x_{f_n})$ are zero. That is, x is sparse.

3.2.2 Compression

The compressed differential data y is calculated by Eq. (1), where A is the measurement matrix.

$$y = Ax \tag{3}$$

The compression ratio is defined by N/M, where N and M is the sizes of x and y respectively. In this thesis, we generate A so that the coherence of A becomes small by the method proposed by Abolghasemi *et al.* [16]. In this method, A is obtained by

$$A = \arg\min_{A} \|A^t A - I\|_F^2. \tag{4}$$

By setting the measurement A whose coherence is small, we can accurately estimate x from y, when x is sparse.

3.2.3 Recovery

The control server recovers the differential traffic information from the compressed differential data y. In this thesis, we use Orthogonal Matching Pursuit (OMP) algorithm [17], which is one of

the well-known algorithms to recover the sparse signals. By using OMP algorithm, the estimated differential traffic data $\hat{x}$ is obtained by

$$\hat{x} = \arg\min_{x} \|y - Ax\|_2^2 \quad \text{s.}t. \quad \|x\|_0 \leq \theta. \tag{5}$$

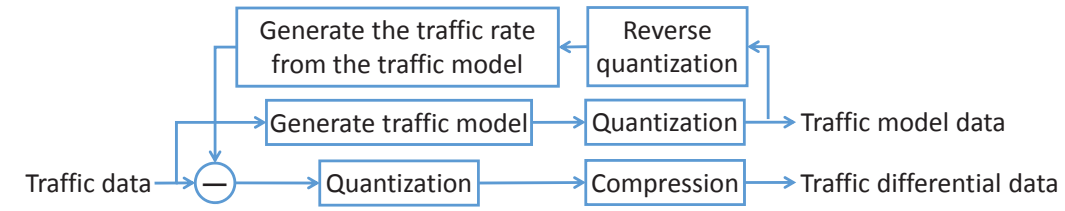
3.2.4 Reverse Quantization

The recovered differential traffic information x is reverse quantized by the quantization size ρ into $\hat{\epsilon}_f$. This $\hat{\epsilon}_f$ is obtained by

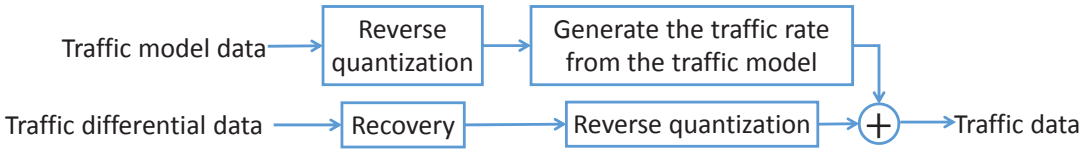
$$\hat{\epsilon}_f = \rho \hat{x}_f. \tag{6}$$

Then, by using the estimated difference $\hat{\epsilon}_f$ and the predicted traffic rate z_f generated by the traffic model, the estimated traffic information z_f is calculated by the following Eq. (7).

$$\hat{t}_f = z_f + \hat{\epsilon}_f \tag{7}$$



(a) Encoding



(b) Decoding

Figure 1: Coding the traffic information

4 Hiearchically Structured Spatio-Temporal Traffic Measurement

4.1 Overview

Figure 2 shows the overview of our method to collect traffic information. In this method, each traffic monitor creates the traffic models for the flows monitored by it and sends them to the control server. Since then, the traffic monitor calculates the differential traffic information by using the traffic model generated for the flow. And, if the differential traffic information exceeds predefined threshold, the traffic monitor $Node_i$ creates the quantized differential traffic information x^{Node_i}. The difference x^{Node_i} generated on each traffic monitor is transmitted along the collection route which forms a tree topology whose root is the control server. The differential traffic data is compressed along the route; the compressed differential traffic data y^{Node_i} is generated by summing the packets received from its children and the compressed differential traffic data of itself. This method enables to collect traffic information without concentration of packets near the control server. The control server recovers the differential traffic information $\hat{x}_f$ for each flow entry from the received packet. Then, the control server gets the corresponding information $\hat{t}_f$ by adding the traffic rate z_f generated by the traffic model and the reverse quantized differential traffic information $\hat{\epsilon}_f$.

4.2 Hierarchical Traffic Data Collection

In this thesis, we use the mechanism based on CDG (Compressive Data Gathering, [18]) to collect the differential traffic information. The CDG is a technique to collect sensor data with a low bandwidth based on the CS. In this technique, the data is compressed along the route from sensor to the sink node. By applying this technique, we compress the traffic differential data monitored by multiple traffic monitors into a small number of packets.

The compressed data y^{Node_i} generated at the traffic monitor $Node_i$ on the collection path is calculated by

$$y^{Node_i} = Ax^{Node_i} + \sum_{Node_j \in C_{Node_i}} y^{Node_j}, \tag{8}$$

where x^{Nide_i} is the differential traffic information for the flows monitored by the traffic monitor $Node_i$, and C_{Node_i} is the set of the children of the traffic monitor $Node_i$. Finally, the control

server receives the amount of the compressed data at k traffic monitors which is obtained by

$$\sum_{i=1,..,k} Ax^{Node_i}. \tag{9}$$

This value is equivalent to the compressed data y from the difference x. This is because of the following deformation.

$$\sum_{i=1,..,k} Ax^{Node_i} = A \sum_{i=1,..,k} x^{Node_i} = Ax = y \tag{10}$$

4.3 Updating Traffic Model

If the traffic models shared by the traffic monitors and the control server are accurate, the traffic differential information becomes sparse. However, the traffic pattern may change significantly, and the traffic model may no longer be unsuitable. In this case, the differential traffic information is not sparse, and is difficult to compress and recover based on the CS.

To address this problem, the traffic monitor and the control server cooperate with each other in updating the traffic models. Figure 3 shows the flowchart to update the traffic model. At each time slot, each traffic monitor checks whether the current traffic model is suitable by checking the difference between the actual traffic information t_f and the predicted traffic rate z_f generate by the traffic model for the flow f. If the current model is unsuitable, (1) the traffic monitor renews the traffic model and sends traffic model data to the control server, (2) the control server updates the traffic model data. Otherwise, unless the model becomes no longer suitable to the current traffic, the traffic monitor sends only the differential traffic information.

There may be several policies to check whether the update is necessary. One example is that the traffic monitor updates the traffic model only when the differential traffic information becomes larger than a threshold at the consecutive λ time slots.

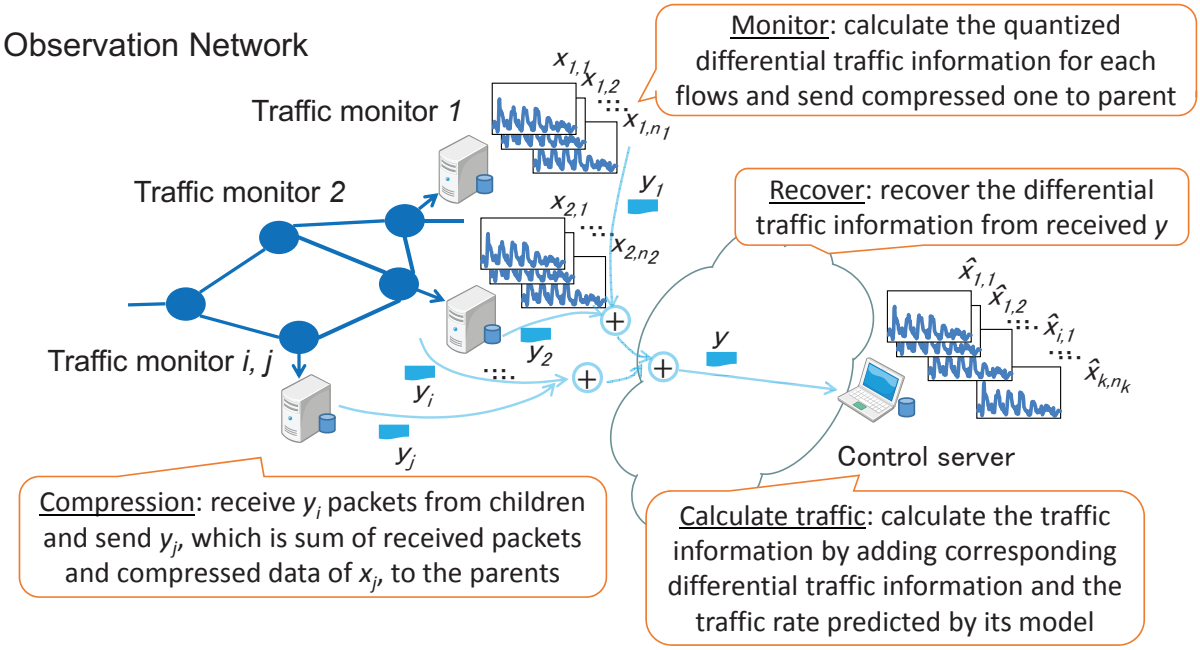
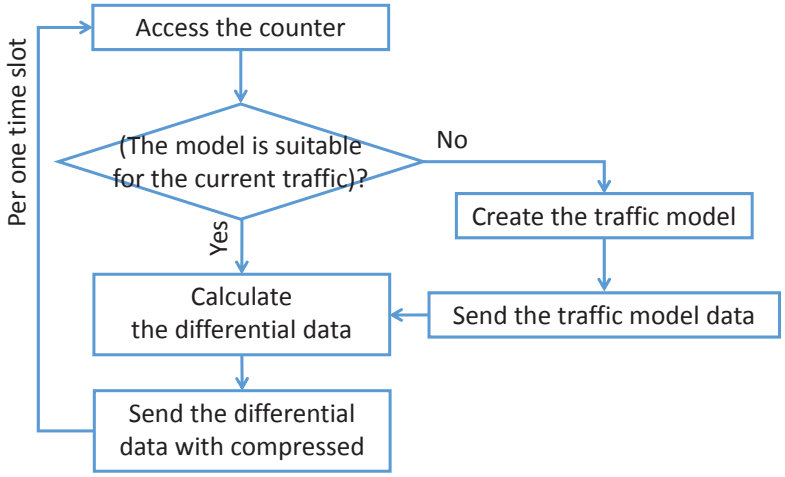
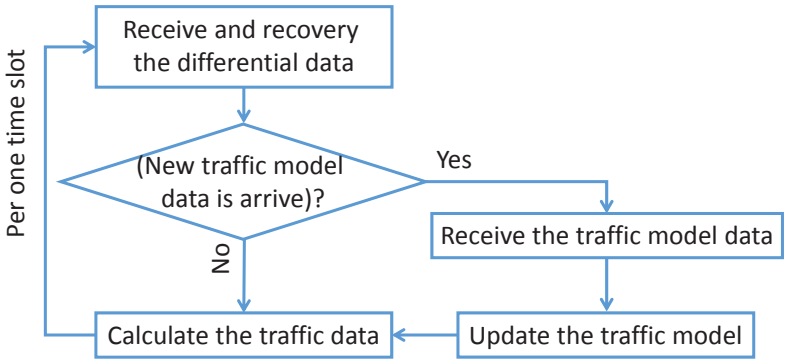


Figure 2: Overview of collection of traffic information



(a) Traffic monitor



(b) Control server

Figure 3: Flowchart to update the traffic model

5 Evaluation

5.1 Evaluation Environment

In this thesis, we evaluate our method, focusing on the steps to compress and recover the traffic information after the traffic models are obtained. In this evaluation, we generate two kinds of the prediction errors of the traffic model. The first one is the white noise, which is the relatively small errors. The other one is the spikes, which cause the significant prediction errors. If the traffic model is accurate, most of the flows include only the white noises, but a small number of flows, whose traffic rates change suddenly, may include the spikes. In this section, we evaluate our method by changing the number of spikes.

The evaluation environment is described as follows. We assume that each traffic monitor monitors 100 flows whose average rate is 10 [Mbps]. We assume that the predictive error ϵ per each time slot follows the Gaussian distribution N $\left(0, \sigma^2\right)$. We set σ to 100 [Kbps] and 3 [Mbps] for the flow with only the white noise and for the flow with the spike respectively. We set the quantization size ρ to $(0.00, 2.50]$ [Mbps] and we set the size of measurement signal $|y|$ to 25, 50, 75, and 100.

In our evaluation, we generate the measurement matrix A by Vahid's method [16] setting the step size to 0.02 and the number of the iteration to 100. We use the OMP algorithm to recover the traffic information. We implement the OMP algorithm by using sklearn.linear_model.OrthogonalMatchingPursuit function, which is available on Python's machine learning package scikit-learn [19].

5.2 Number of non-Zero Entries after Quantization

Figure 4 shows the number of non-zero elements in the quantized differential data x by each quantization size. As the quantization size becomes large, the non-zero entries in the quantized differential data become small. The result demonstrates that the quantization size should be set to a large value so as to keep the number of non-zero entries small, as the number of spikes increases.

5.3 Errors caused by Compression

Figure 5 through 8 show the error caused by the CS. In these figures, the horizontal axis is the quantization size, and the vertical axis is the error caused by the CS. These figures show the results for the different compression ratio; Figures 5, 6, 7 and 8 show the results for the cases that compression ratio is 1, 1.33, 2, and 4 respectively. In this evaluation, we use the two kinds of errors as the metrics. The first one is the average error defined by

$$\frac{1}{n}\sum_{i=1}^{n}(\rho \cdot |\hat{x}_{f_i} - x_{f_i}|), \tag{11}$$

where ρ is the quantization size, x_{f_i} is the quantized differential traffic information for the flow f_i, and $\hat{x}_{f_i}$ is the recovered differential traffic information for the flow by the CS. The other one is the worst error defined by

$$\max_{i=1,\ldots,n}(\rho \cdot |\hat{x}_{f_i} - x_{f_i}|). \tag{12}$$

These figures show that the errors become large as the quantization size becomes small. This is because the smaller quantization size causes more non-zero entries in the differential traffic information, which degrades the performance of the CS.

5.4 Overall Errors

Figures 9 through 12 show the overall errors in the measured traffic. In these figures, the horizontal axis is the quantization size, and the vertical axis is the errors. Similar to Section 5.3, we evaluate the errors with various compression ratio, and we use two metrics on errors; the first one is the average error defined by

$$\frac{1}{n}\sum_{i=1}^{n}(|\epsilon_{f_i} - \epsilon_{f_i}|), \tag{13}$$

and the other one is the worst error defined by

$$\max_{i=1,\ldots,n}(|\epsilon_{f_i} - \epsilon_{f_i}|). \tag{14}$$

From these figures, we discuss the impact of the compression ratio on the number of spikes that can be accurately recovered. These figures indicate that our method allows 90, 50, 25, and 7 spikes without causing overall error larger than 1 [Mbps], by setting the compression ratio to 1,

1.33, 2, and 4 respectively. In other words, if the number of spikes included in the traffic becomes smaller than the quarter of the total number of the entries, we can compress the traffic information by half, though the compression ratio should be set to a small value if more spikes are included in the traffic.

5.5 Discussion on Intervals to Update Traffic Models

When the significant changes of the traffic pattern occur, the traffic model may no longer be suitable to the current traffic, and a large prediction error may occur. If many flows include large prediction errors, the differential traffic information includes a large number of non-zero entries, and our compression based on the CS causes the large errors. To avoid large errors, we should keep the traffic model suitable to the current traffic by updating the traffic model. In this subsection, we discuss the policy to update the traffic model.

According to the results discussed in Section 5.4, the number of non-zero entries should be less than 25 $\%$ of the number of the total entries so as to make the compression ratio 50 $\%$ without causing errors larger than 1 [Mbps]. The interval to update the traffic models can be discussed based on this result. Assuming that the number of non-zero entries in the differential traffic information is 10 $\%$ of the number of total entries at the first steps after the models are updated, and the number of non-zero entries increases by 1.1 times at each time slot, the required interval to update the traffic model λ should satisfy

$$0.1 \times 1.1^{\lambda} < 0.25. \tag{15}$$

Solving this equation, we obtain $\lambda = 9$.

To obtain the interval to update the traffic mode, we need the number of non-zero entries of the traffic information of whole network. However, the number of non-zero entries is unknown for the traffic monitors. Instead, each traffic monitor can use the number of non-zero entries in the differential traffic information generated by it to estimate the number of non-zero entries of the traffic information of whole network.

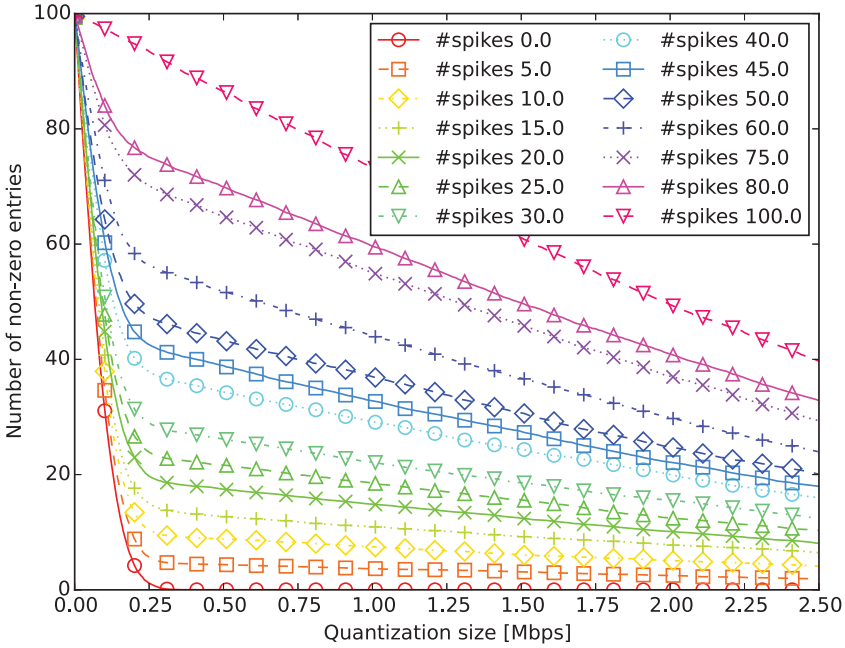
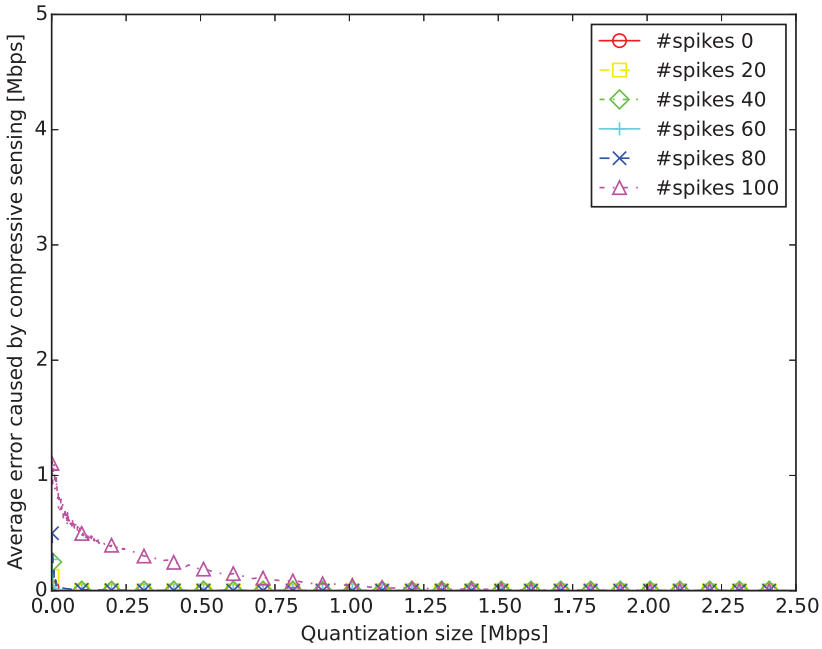
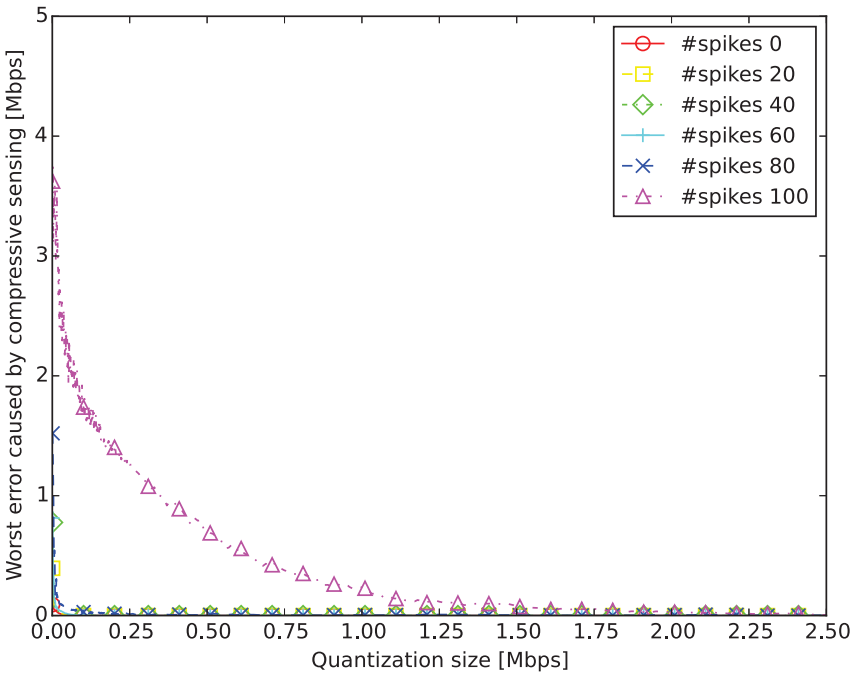


Figure 4: Number of non-zero entries

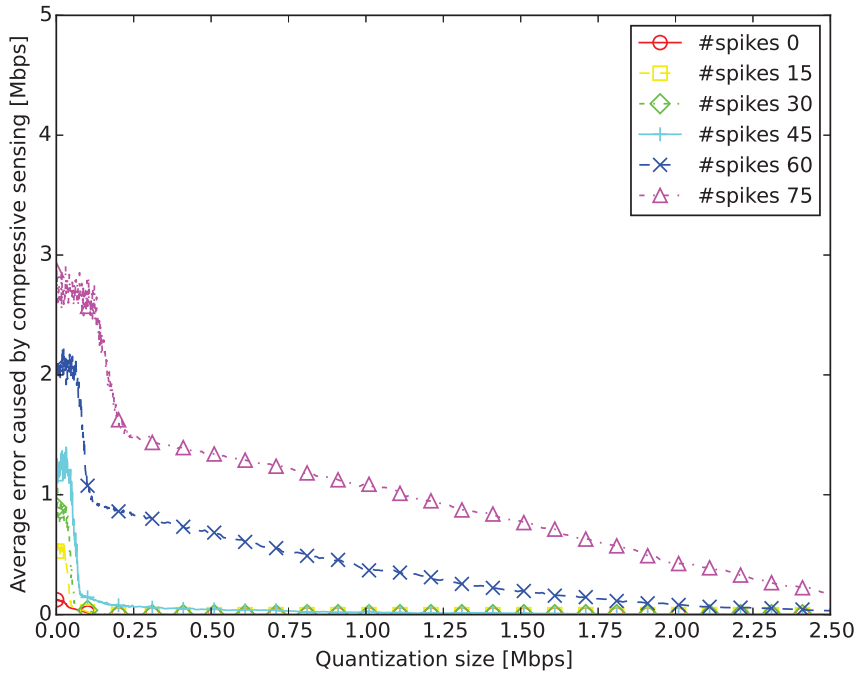


(a) Average error

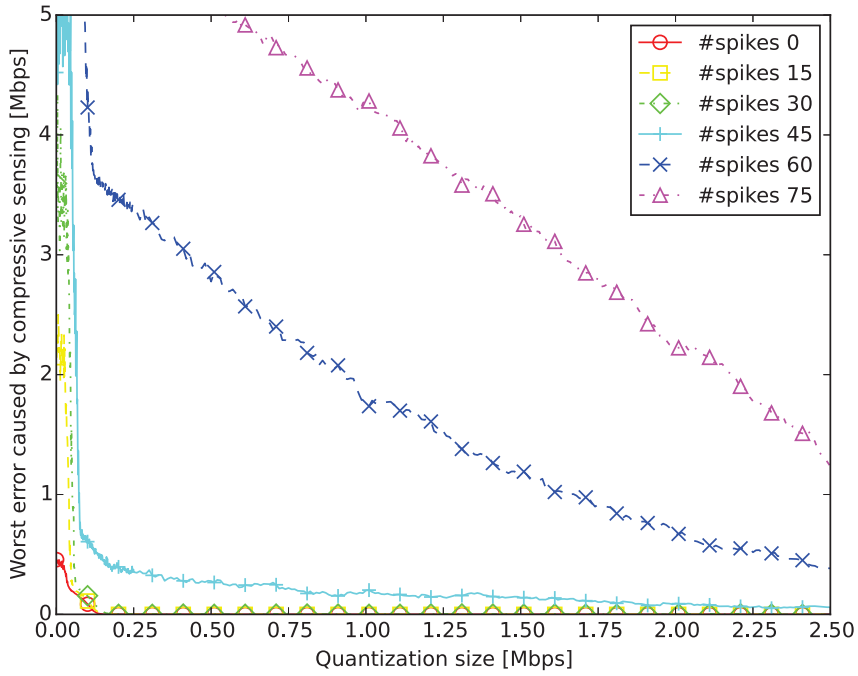


(b) Worst error

Figure 5: Error caused by CS ($|y| = 100$, compression ratio is 1)

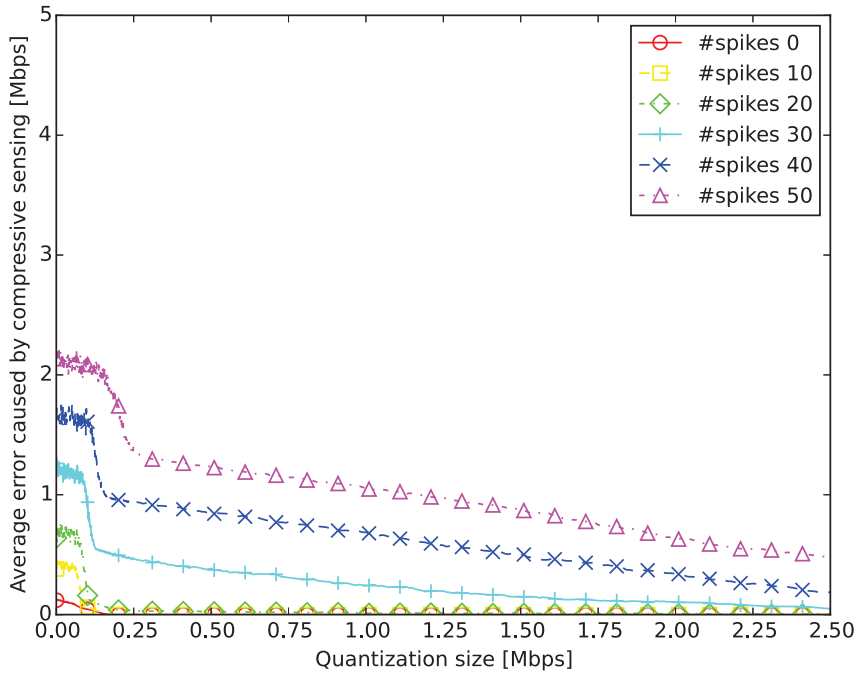


(a) Average error

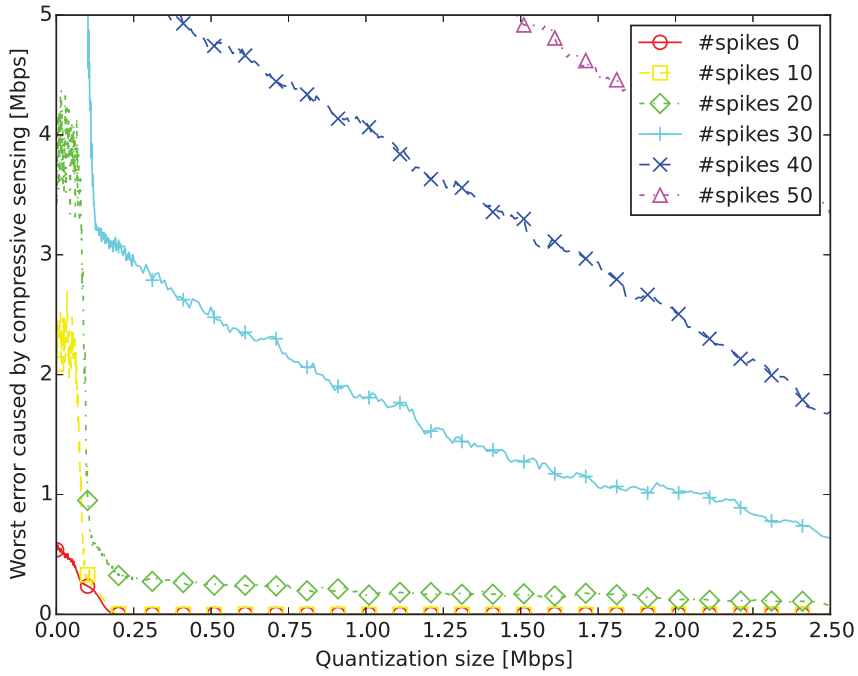


(b) Worst error

Figure 6: Error caused by CS ($|y| = 75$, compression ratio is 1.33)

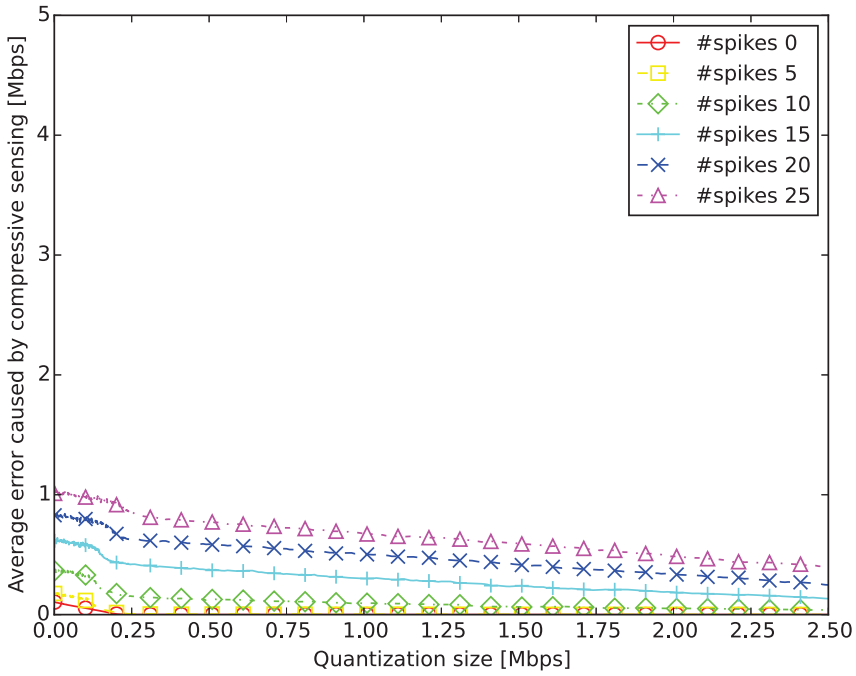


(a) Average error

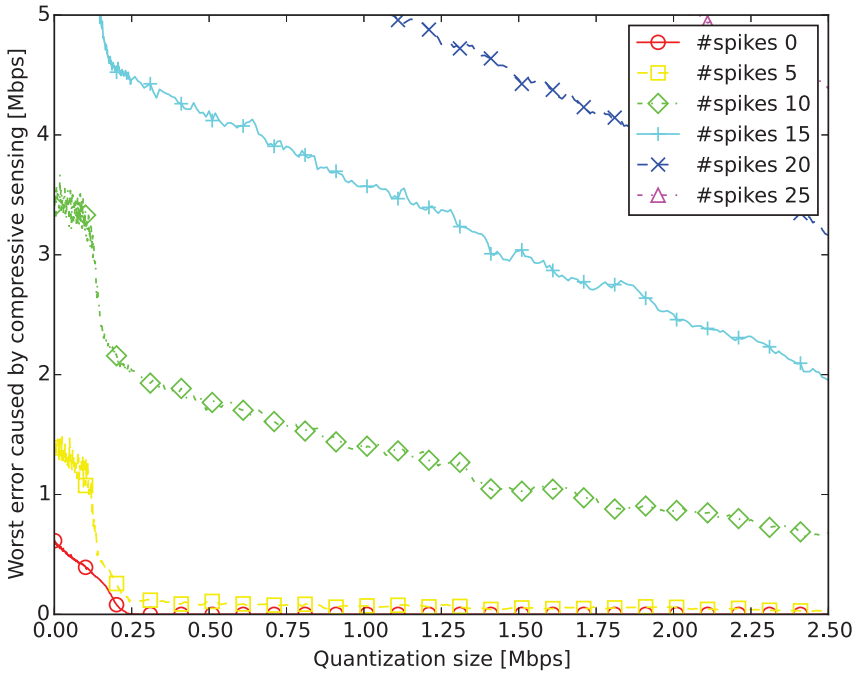


(b) Worst error

Figure 7: Error caused by CS ($|y| = 50$, compression ratio is 2)

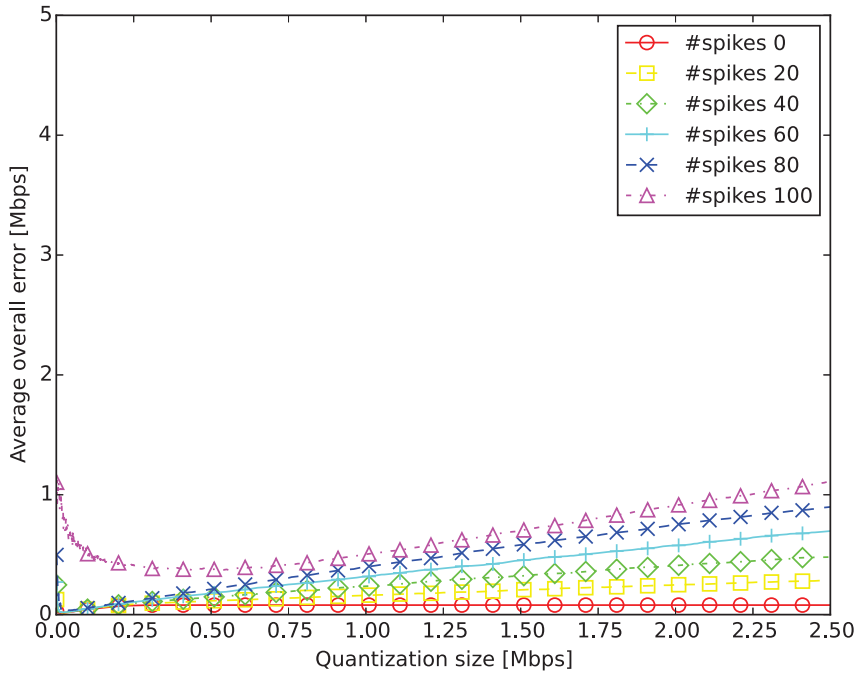


(a) Average error

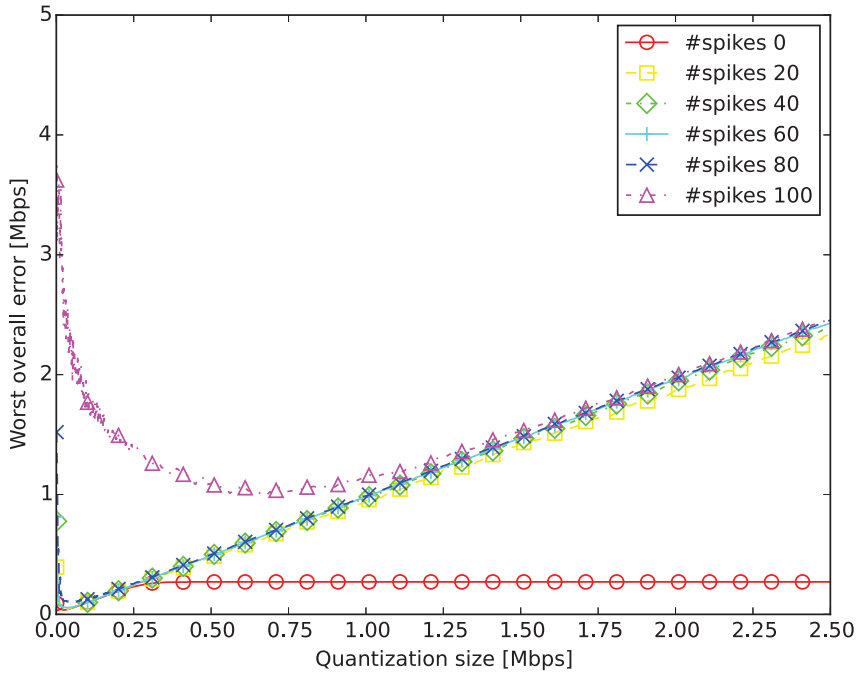


(b) Worst error

Figure 8: Error caused by CS ($|y| = 25$, compression ratio is 4)

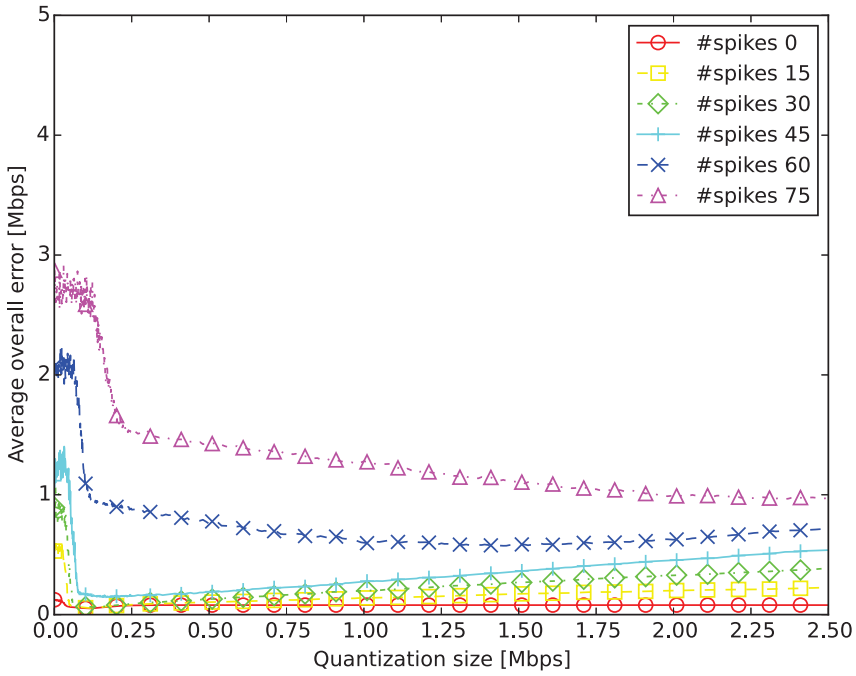


(a) Average error

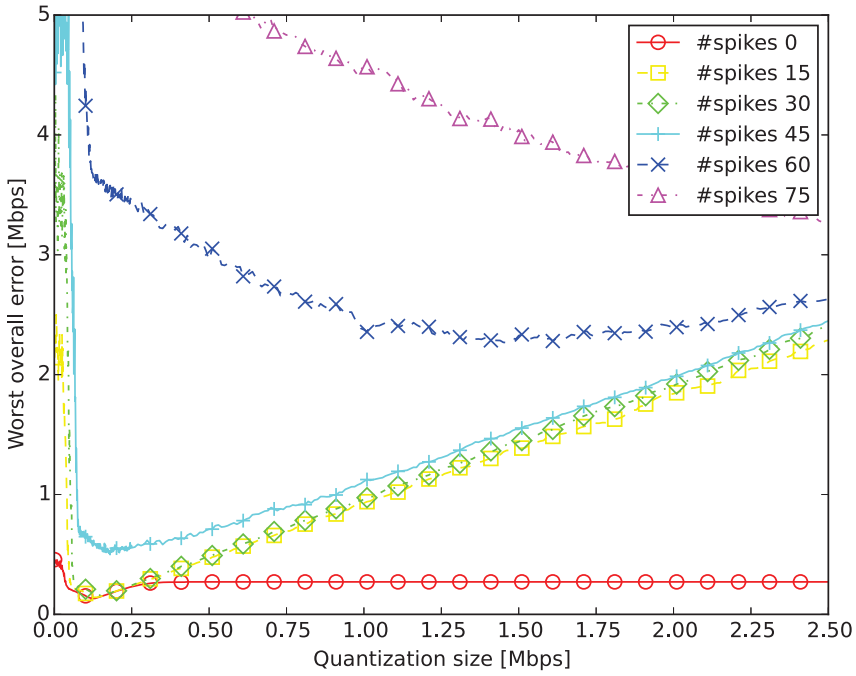


(b) Worst error

Figure 9: Overall error ($|y| = 100$, compression ratio is 1)

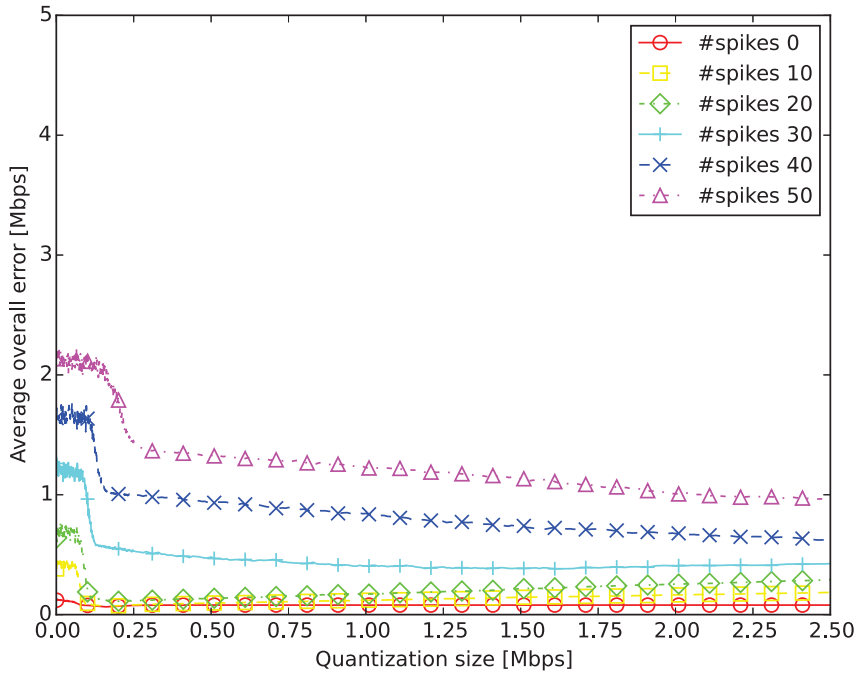


(a) Average error

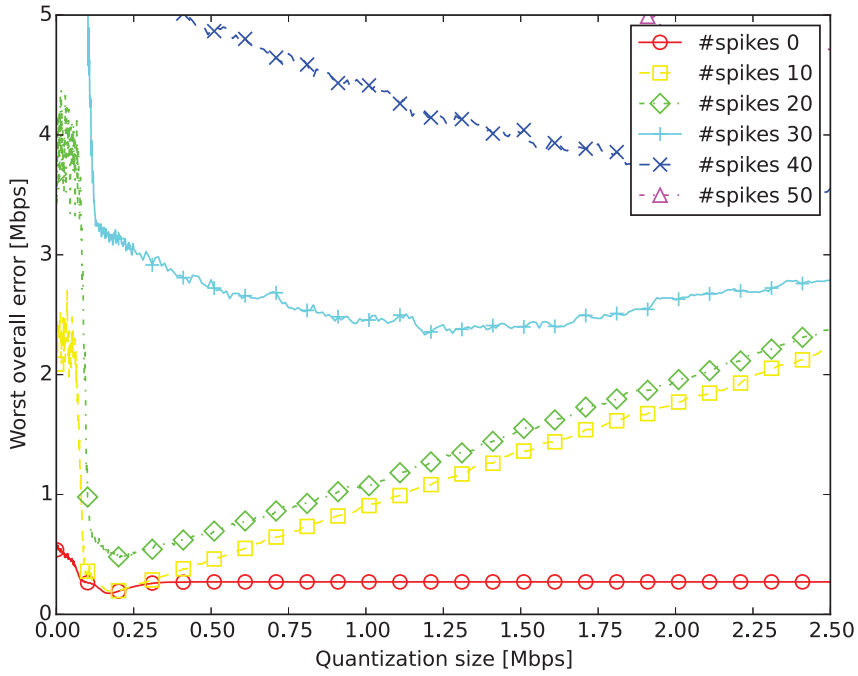


(b) Worst error

Figure 10: Overall error ($|y| = 75$, compression ratio is 1.333)

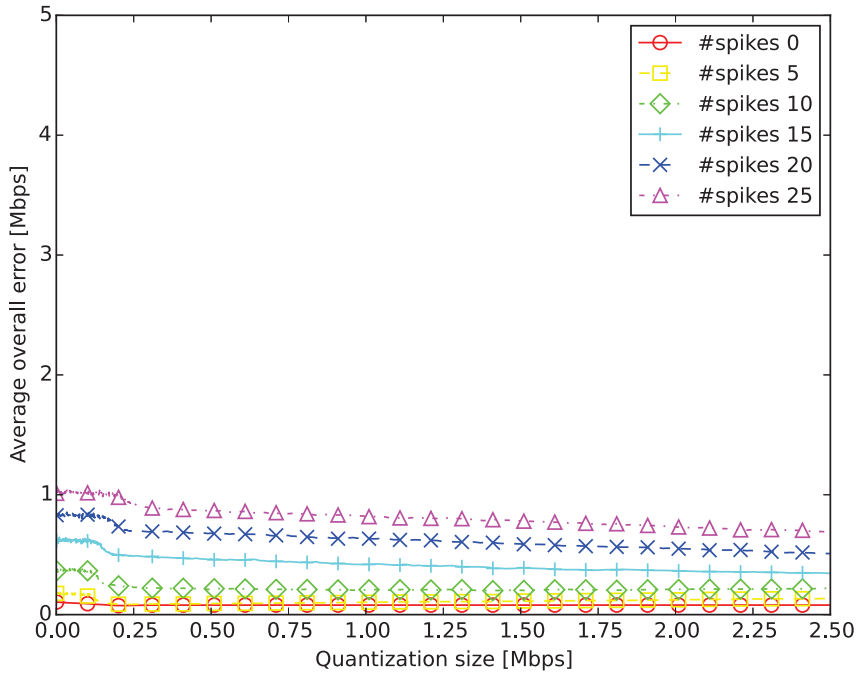


(a) Average error

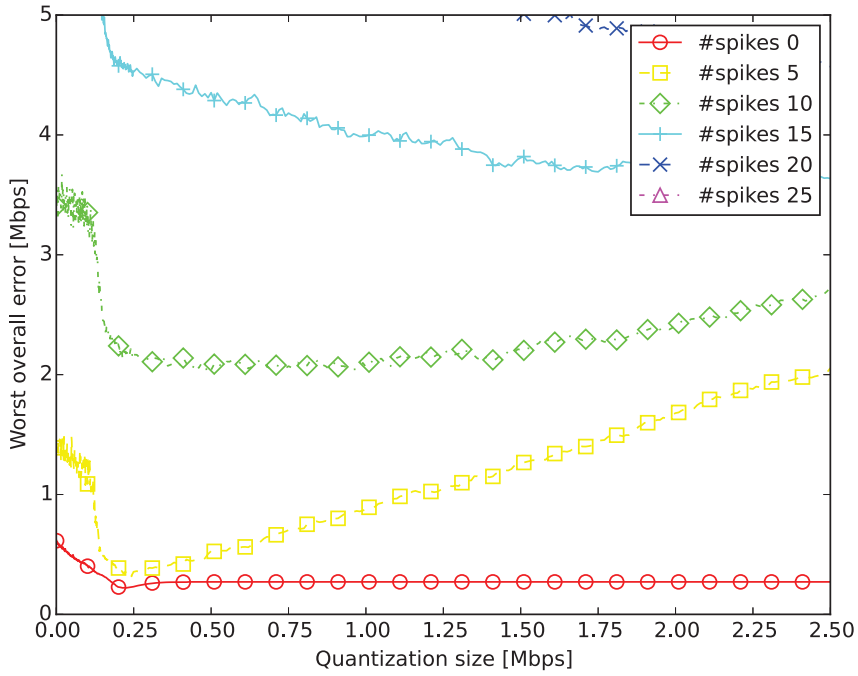


(b) Worst error

Figure 11: Overall error ($|y| = 50$, compression ratio is 2)



(a) Average error



(b) Worst error

Figure 12: Overall error ($|y| = 25$, compression ratio is 4)

6 Conclusion

We proposed a method to collect traffic information without causing a large overhead to the network. In our method, the traffic monitor creates the traffic model for the flow to be monitored, and share it with the control server. Then, unless the model becomes no longer suitable to the current traffic, the traffic monitor sends only the information on the difference from the models. Then, the control server obtains the traffic data by adding the differential traffic information to the traffic rate generated by the traffic model.

We also introduced a method to compress the differential traffic information along with the route to the controller by using the compressive sensing. In this method, we construct a tree whose root is the controller. The differential traffic information is sent along the tree from the leaves to the root. Each traffic monitor compresses the differential traffic information generated by it and those sent from its children into a few number of packets. Thus, this approach avoids the concentration of traffic information on the links near the controller.

We evaluated our method by numerical simulations. The results demonstrate that we can reduce the amount of the information required to be collected by half without causing the errors larger than 1 [Mbps], if the number of flow entries which cannot be predicted accurately is smaller than a quarter of the number of the total entries.

Our future work includes the evaluation of our method combined with the creation of the traffic models. Moreover, we will also discuss the method to reduce the overhead to collect traffic information more by utilizing the correlation between flow entries.

Acknowledgments

Foremost, I would like to express my deepest gratitude to Professor Masayuki Murata of Osaka University for his exact guidance and insightful comments with foresight. Furthermore, I would like to show my sincere appreciation to Assistant Professor Yuichi Ohsita of Osaka University for continuous support, helpful discussions, and insightful advice. Without their support, I cannot achieve all results in my research life.

My sincere appreciation also goes to Mr. Akihiko Miyazaki, and Mr. Koji Yamazaki of NTT Device Technology Laboratories for their helpful comments and fruitful discussions. Moreover, I would like to show my appreciation to Professor Kazunari Inoue of Nara National College of Technology for support in the implementation aspect.

I am also grateful to the helpful advice from Associate Professor Shin'ichi Arakawa, Assistant Professor Daichi Kominami, and Assistant Professor Yuya Tarutani of Osaka University. Finally, I would like to thank my senior associates including Mr. Tatsuya Otoshi, and my colleagues in the Department of Information Networking, Graduate School of Information Science and Technology of Osaka University for their kindness.

References

[1] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic Engineering in Dynamic Networks," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 99–110, Aug. 2006.

[2] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and Principles of Internet Traffic Engineering," RFC 3272, 2002.

[3] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, Oct. 2004.

[4] P. Phaal, S. Panchen, and N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," RFC 3176, 2001.

[5] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine Grained Traffic Engineering for Data Centers," in *Proceedings of ACM CoNEXT*, 2011, pp. 8:1–8:12.

[6] L. Yuan, C.-N. Chuah, and P. Mohapatra, "ProgME: Towards Programmable Network Measurement," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 97–108, Aug. 2007.

[7] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Diamond in the Rough: Finding Hierarchical Heavy Hitters in Multi-dimensional Data," in *Proceedings of ACM SIGMOD ICMD*, 2004, pp. 155–166.

[8] L. Jose, M. Yu, and J. Rexford, "Online Measurement of Large Traffic Aggregates on Commodity Switches," in *Proceedings of USENIX Hot-ICE*, 2011.

[9] Y. Lee and Y. Lee, "Toward Scalable Internet Traffic Measurement and Analysis with Hadoop," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 1, pp. 5–13, Jan. 2012.

[10] M. F. Zhani, H. Elbiaze, and F. Kamoun, "Analysis and Prediction of Real Network Traffic," *Journal of Networks*, vol. 4, no. 9, pp. 855–865, 2009.

[11] B. Krithikaivasan, Y. Zeng, K. Deka, and D. Medhi, "ARCH-Based Traffic Forecasting and Dynamic Bandwidth Provisioning for Periodically Measured Nonstationary Traffic," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 683–696, 2007.

[12] B. Zhou, D. He, Z. Sun, and W. H. Ng, "Network Traffic Modeling and Prediction with ARIMA/GARCH," in *Proceedings HET-NETs Conference*, 2005, pp. 1–10.

[13] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet Traffic Forecasting using Neural Networks," in *IEEE International Joint Conference on Neural Networks*, 2006, pp. 2635–2642.

[14] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge Univ. Press, 2012.

[15] M. Roughan, M. Thorup, and Y. Zhang, "Traffic Engineering with Estimated Traffic Matrices," in *Proceedings of ACM SIGCOMM IMC*, 2003, pp. 248–258.

[16] V. Abolghasemi, S. Ferdowsi, B. Makkiabadi, and S. Sanei, "On Optimization of the Measurement Matrix for Compressive Sensing," in *Proceedings of European Signal Processing Conference*, 2010, pp. 427–431.

[17] J. Tropp and A. Gilbert, "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, Dec 2007.

[18] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive Data Gathering for Large-scale Wireless Sensor Networks," in *Proceedings of ACM MobiCom*, 2009, pp. 145–156.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.