Bandwidth Reduction in SNMP Monitoring System with Bloom Filter using Lossless Compression

Warinda Kiatdherarat and Chayakorn Netramai
The Sirindhorn International Thai-German Graduate School of Engineering (TGGS),
KMUTNB,
Bangkok, Thailand,
m12e.warinda@tggs-bangkok.org, chayakorn.n.sse@tggs-bangkok.org

***Abstract*— This work focuses on the further reduction of the bandwidth usage of the Bloom Filter based SNMP by means of an additional compression scheme. The zip compression technique is used for the zip encryption and the zip decryption of the Bloom Filter. The experiment with compression ratio was done and the results show that the SNMP bandwidth that utilizes the proposed compression scheme is reduced to 10.23% and the compression time is small and negligible and SNMP information still retain its 100% correctness.**

Keywords*— Network Management System; SNMP; Lossless Compression; Bandwidth Reduction; Bloom Filter

I. INTRODUCTION

It is important to manage a network system in order to keep all devices population working properly. It is also very important for network administrators to have a way to manage and control all pieces of network equipment easily and conveniently. The basic features of network management system are to retrieve device information and to control device remotely, both are very essential to keep quality of the network up to the network customer's expectation.

Usually network administrator can use basic network commands such as ping and ipconfig to manage the network. Therefore, it is not a problem for managing small and medium size network due to number of devices attach to network. However this is different for a larger network, the bigger the network, the harder to manage by such conventional mean.

There are several protocols that can be used to manage the network system such as mobile agent, CORBA, web service (xml), and compress xml. All these protocols have concern regarding the implementation and the compatibility issues. Simple Network Management Protocol (SNMP) provide a simplest way to manage the network system. It is a default protocol that is available on every network device. It is highly compatible and most Network Management System (NMS) software uses it as a default protocol. Therefore SNMP is preferred in a large network with a mix of different devices across different vendors because it doesn't need any extra implementation beforehand in order to address the compatibility issues. However, SNMP has one major drawback which is its large bandwidth requirement. If the size of the network is small, it can operate fairly well. But when the size of network becomes bigger, the amount of bandwidth needed for SNMP will also increase and it might significantly effects the performance of the network quality.

Currently there are several techniques to reduce the SNMP bandwidth usage. According to the experimental results from [2], the implementation of SNMP with compression method requires only 75% of the normal SNMP implementation. From the same publication, in case of web services, the required bandwidth is only 50% of the standard SNMP implementation.

In our previous work [1], Bloom Filter has been introduced to further reduce the SNMP bandwidth usage. It has been shown that such approach can reduce the bandwidth usage by 67% [1].

This work focuses on the further reduction of the bandwidth usage of the Bloom Filter based SNMP by means of an additional compression scheme. It also takes a look into the questions regarding the possible drawbacks from such approach in terms of speed and correctness.

II. BLOOM FILTER WITH BIT VECTOR COMPRESSION SCHEME

Bloom Filter is a space efficiency randomizes data structure technique for finding the member in set queries that allow false positive with 100% recall rate [5]. The query would return either "possible in set" or "definitely not in set" as shown in Figure 1.

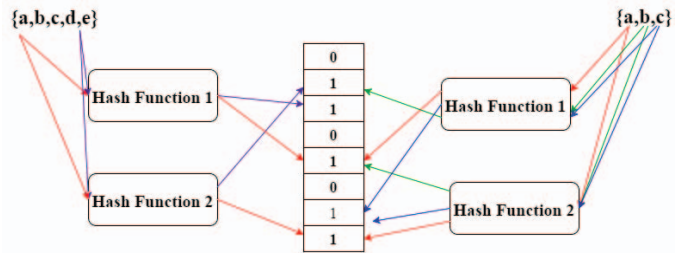

Figure 1. An example of a Bloom Filter usage with Hash function.

The bit array of Bloom Filter is initially contains 0 in all location. On the left side show the set with members *{a,b,c,d,e}*. When a value of member a passes through *Hash*

978-1-4673-7670-9/15/$31.00 ©2015 IEEE

function 1 and *Hash function 2*, then the result is stored on the bit array by changing from 0 to 1 on the respected index, inserting process of Bloom Filter. On the right side show the set with members *{a,b,c}*, show the checking process of Bloom Filter, it would need to use the same *Hash function 1* and *Hash function 2*, to check each existing of its member in bit array of Bloom Filter. Note that the chosen number of hash function can impact the false positive error rate whereas the chosen size of each member in set has no effect with the size of Bloom Filter.

Bloom Filter is a binary bit vector array with the false positive error. To further reduce the bandwidth usage of the network means to reduce the size of Bloom Filter. In order to avoid changing the size of the Bloom Filter, the best way to reduce the bandwidth usage in SNMP with Bloom Filter is by applying a lossless compression. Lossless compression is a compressing technique that allows the original data to be perfectly reconstructed from the compressed data.

III. System Overview

Figure 2 shows the conventional SNMP network system with Bloom Filter module as implemented in [1]. In order to utilized the Bloom Filter in a conventional SNMP there is a Bloom Filter inserting before SNMP enter the network and there is a Bloom Filter checking right out of the network.

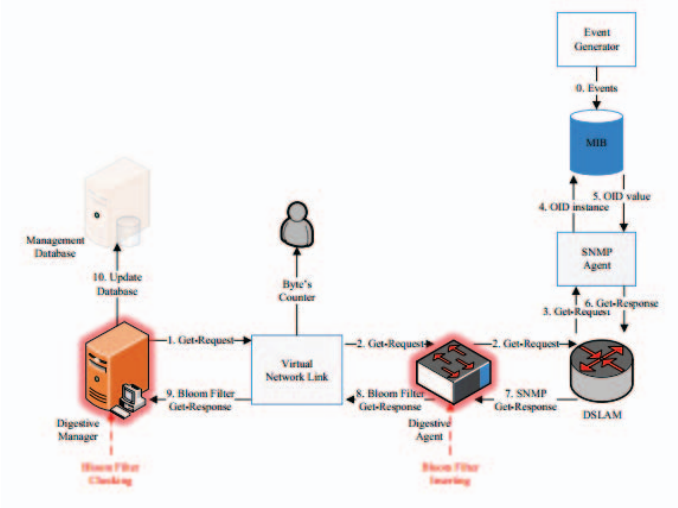

Figure 2. Overview of the conventional SNMP system with Bloom Filter technique

Now in order to further compress the Bloom Filter, the encryption agent was placed after the Bloom Filter inserting and decryption agent was placed before the Bloom Filter checking. This can be illustrated in Figure 3.

Figure 3. The network system with Bloom Filter and the compression

IV. Design and Implementation

We used the model of the stimulator previously develop in [1] and maintain all the optimal number of hash function, devices, and size of the Bloom Filter. Then we add the lossless compression component to the stimulator as mentioned in the previous section.

In this work, the zip compression was chosen as the compression method. The zip compression permits a number of compression algorithms, the most common is DEFLATE. This format was originally created in 1989 by Phi Katz[4]. For our implementation the java.util.zip[3] is used for the zip encryption and the zip decryption of the Bloom Filter.

V. Experiments and Results

The goals of the experiments are to prove that the proposed compression scheme can further reduce the bandwidth usage of the Bloom Filter-based SNMP. They will also prove that such approach has no effect on the correctness during compress and decompress of Bloom Filter information.

Several experiments were done with the input of Bloom Filter files obtained this information listed in Table I from the stimulator mentioned in the previous section. The size of input Bloom Filter files range from 2^{16} to 2^{30} byte, they contain the randomized IP address.

In order to evaluate the purposed method, we need to check two criteria which are (1) compression ratio and (2) The compression time.

1. Compression ratio test:

This experiment tested how big the compression ratio can be achieved by utilizing the zip compression. The bigger the ration rate it is the lesser bandwidth usage. We create set of bit vectors with different size of input and feed them in to the compression module then compare the compression ratio. The set of data is the bit vector of size range from 2^{16} to 2^{30}.

TABLE I. THE DATA SIZE FOR EACH GROUP.

The actual size of input group (byte)	Group name
$2^{17} - 2^{16} = 65536$	Group 1
$2^{18} - 2^{16} = 196608$	Group 2
$2^{20} - 2^{16} = 983040$	Group 3
$2^{22} - 2^{16} = 4128768$	Group 4
$2^{24} - 2^{16} = 16711680$	Group 5
$2^{26} - 2^{16} = 67043328$	Group 6
$2^{28} - 2^{16} = 268369920$	Group 7
$2^{30} - 2^{16} = 1073576288$	Group 8

The compression ratio was calculated by comparing the original size of Bloom Filter file for each length of 2^{16} to

2^{30} to the file size after passing through the zip compressing. Each test was repeated for 100 times for each group of data size.

Table II shows the average size of Bloom Filter before and after compression for each group data size in detail. The same information is then used to construct Figure 4. From this information, the trend of the compression ratio versus the data size can be shown in Figure 5. We can see that, by using the zip compression, the size of Bloom Filters is reduced by the average of 69% of Bloom Filter-based SNMP, which is equal to 10.23% of the standard SNMP bandwidth usage.

TABLE II. THE AVERAGE SIZE OF THE BLOOM FILTER BEFORE AND AFTER COMPRESSION AND THE COMPRESSION RATIO.

Group number	Average size before compression (byte)	Average size after compression (byte)	Compression ratio (%)
Group 1	607574.90	187913.90	66.28
Group 2	853358.63	257673.27	69.27
Group 3	1356134.82	401550.92	69.72
Group 4	1787179.43	519985.80	70.03
Group 5	2104379.24	618804.18	69.84
Group 6	2357828.20	705430.48	69.84
Group 7	2569695.03	785092.55	69.01
Group 8	2749759.85	860094.45	68.47

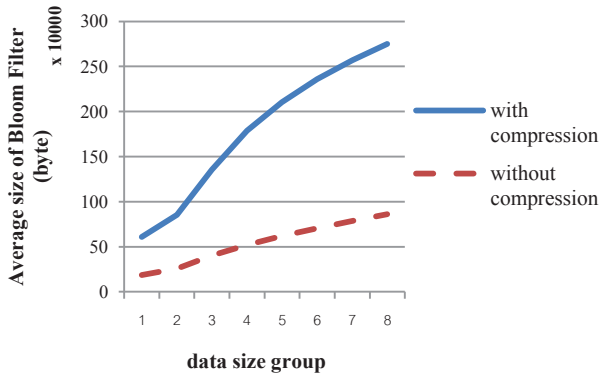

Figure 4. Average size of Bloom Filter before and after compression.

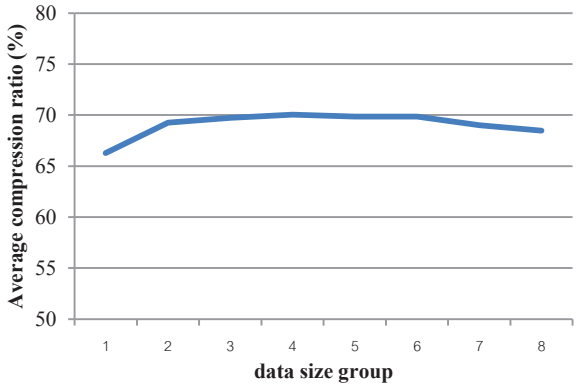

Figure 5. Average compression ratio with compression compared to without compression.

2. Compression time test:

In this experiment we compare the time needed for the SNMP operation with and without utilizing the compression. The experiment was done using the same data grouping as shown in Table I. If it take a lot of time for compression and decompression, it will be bad for the SNMP process because the device status wouldn't be able to monitor in real time and it would be affecting the performance on the management system. From the study in [1], the polling time for network management system is set to around 5 - 30 minute. The default value is around 15 minute. Therefore, the good process time should not be exceeding 15 minutes.

We use the set of bit vector array of size range from 2^{16} to2^{30}, they contain the randomized IP address as input. We run the simulator without the compressing and decompressing process then measure the process time. Then we repeat the test using the same input but this time also utilizing the zip compression and decompression on the input. The test was repeated for 100 times for each group of data size. We used the java library System.currentTimeMillis() to measure the elapse time of each process. The measured time is in millisecond.

Table III shows the measured average process time different (in percentage) for with and without compression in detail. This is illustrated in Figure 6. Figure 7 shows the trend of the time ratio versus data size. From the results, it can be seen that the maximum time in the case with compression scheme is 83 seconds.

TABLE III. THE AVERAGE TIME FOR WITH AND WITHOUT COMPRESSION FOR EACH DATA SIZE

Group number	Average time with compression (ms)	Average time without compression (ms)	Process time difference (%)
Group 1	17206.94	14169.32	21.65
Group 2	20652.44	18220.88	13.35
Group 3	36374.28	29552.08	23.32
Group 4	53896.15	39344.71	37.20
Group 5	68842.41	47619.65	44.59
Group 6	66892.96	92928.48	-27.98
Group 7	70871.50	108576.79	-34.72
Group 8	83252.44	132114.39	-36.98

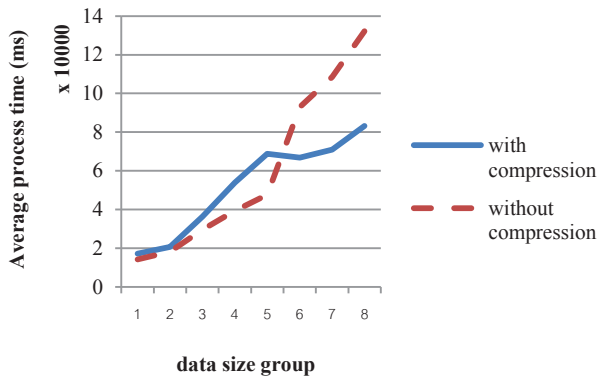

Figure 6. Average time with and without compression.

Figure 7. Average % time increase with the compression compared to without compression.

VI. CONCLUSION

From the results, it can be seen that the size of data after compression is decreasing around 69 percent of Bloom Filter-based SNMP which is equal to 10.23% of the standard SNMP bandwidth usage. This is an obvious improvement in terms of reducing the bandwidth usage. It can also be seen that the longest time with compression is 83 second, which is still faster than network polling time of 15 minutes. This means that the proposed compression scheme will not effect the quality and performance of the network management system.

Note that the decompressed Bloom Filter files have also been check on its correctness compare to its Bloom Filter original files before the compression process. The result shows 100% correctness.

The goal of this experiment is to reduce the bandwidth usage of Bloom Filter's based SNMP with the lossless compression technique. From the experiment it had prove that the bandwidth usage is reduce without any other major drawback to the operation of the SNMP. The compression time is small and negligible and SNMP information still retains its 100% correctness.

VII. REFERENCES

[1] Wannachakrit C., Anwar T. 2012. "Managing Network System by Bloom Filter ," in International Journal of Digital Content Technology and its Applications(JDCTA), vol. 6, Number22, December 2012J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] Pras, A., Thomas Drevers, R. van de Meent, and D. Quartel. 2004. "Comparing the Performance of SNMP and Web Services-Based Management." IEEE Transactions on Network and Service Management 1 (2): 72–82. doi:10.1109/TNSM.2004.4798292.

[3] java.util.zip (java platform SE7) , http://docs.oracle.com/javase/7/docs/api/java/util/zip/package-summary.html, Oct 2015

[4] PKWARE ZIP File Format Specification, https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT, Oct 2015

[5] Broder A., Mitzenmacher M., 2004. "Network applications of Bloom filters: A survey," Internet Mathematics, vol. 1 no. 4, pp. 485-509, 2004.