Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 153034, 7 pages
http://dx.doi.org/10.1155/2013/153034

Research Article

Data Deduplication in Wireless Multimedia Monitoring Network

Yitao Yang,[1,2] Xiaolin Qin,[1] Guozi Sun,[2] Yong Xu,[1,3] Zhongxue Yang,[1] and Zhiyue Zu[2]

[1] *College of Computer Science & Technology, Nanjing University of Aeronautics & Astronautics, Nanjing, Jiangsu 210016, China*
[2] *College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210003, China*
[3] *Department of Computer Science & Technology, Anhui University of Finance & Economics, Bengbu, Anhui 233030, China*

Correspondence should be addressed to Xiaolin Qin; qinxcs@nuaa.edu.cn

Received 17 July 2013; Accepted 28 October 2013

Academic Editor: Zhijie Han

Wireless sensor network has been applied to many areas for a long time. A new kind of wireless sensors equipped with a camera and a microphone has been emerging recently. This kind of sensor is called wireless multimedia sensor (WMS) because it can capture and process multimedia data such as image, sound, and video. The visual monitoring network is a typical scenario of WMS application. Massive data would be produced in a short time because of the intensive WMS deployment. Many data aggregation and compression technologies have been proposed for addressing how to transfer data efficiently. However, data aggregation technologies need highly efficient router algorithm, and compression algorithms might consume more computation time and memory because of the high complexity. This paper applies data deduplication technology to this scenario. It can eliminate the redundant data from raw data to exploit the network bandwidth efficiently. Moreover, a chunking algorithm with low computation complexity is presented in this paper, and its efficiency has been proved through the experiments.

1. Introduction

Wireless sensor network (WSN) consists of a large number of spatially distributed autonomous sensors which are connected by wireless communication. It is mainly used for dynamic acquisition of physical information within the network coverage which will be delivered to users later. Currently, WSN is widely applied to the capture, processing, and transmission of the data such as temperature, light intensity, humidity, and gas concentration [1, 2]. In the recent years, as the production level of sensors improves, additional camera, microphone, and other functional devices are installed on traditional wireless sensors and enable them to capture, process, and transfer multimedia information such as image, sound, and video, so that users can obtain improved physical information which is more vivid and more accurate. This new kind of sensor can be called wireless multimedia sensor (WMS). Wireless multimedia sensor network (WMSN) consists of WMS nodes, gateway nodes with storage, sink nodes, and so on. Then, a typical WMSN architecture is proposed in [3], as shown in Figure 1. WMSN is widely used in many areas including visual monitoring, individual positioning, industrial control, intelligent transportation, environmental monitoring, smart home, and telemedicine [3–5].

Multimedia sensor nodes have limited resources such as energy capacity of battery, storage space, and computing power, while the information they acquire features large data size and high requirement for computing. The data in-network processing technology [6, 7], as one of the key technologies used to save bandwidth and network energy in data-centered WMSN, can eliminate the redundancy of source data and minimize the data traffic between nodes by applying data fusion technology or data compression. The data fusion technology uses different routing methods to combine data packages and eliminate redundant data from them. This technology relies on three basic modules:routing algorithm, data fusion, and data presentation [6]. However, in addition to the advantage of high data aggregation, the application of data fusion technology may also lose raw data structure, while data compression can prevent this problem. Data compression has been applied to WMSN, and the compression algorithm can be divided into two categories:distributed data compression scheme and local data compression scheme. The proposed data compression

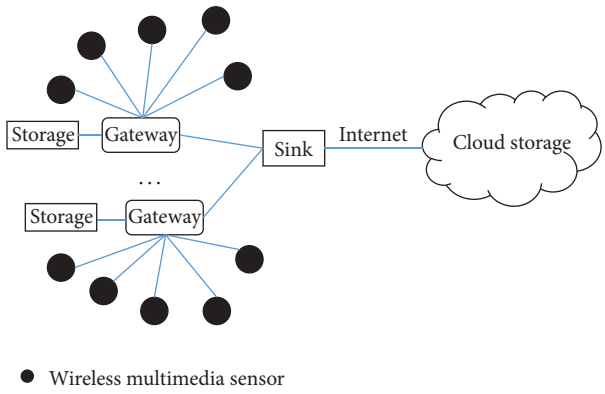

Figure 1: A typical system architecture of WMSN.

algorithm is the extension of data fusion technology in multiple-hop network topology, and compression algorithms are applied to the entire network [8, 9]. Other compression algorithms independently perform data compression on local node, so they do not depend on intensive network or routing algorithm and are more suitable for sparse sensor network [10, 11]. The data compression scheme can reduce the amount of data transmission in sensor networks to save bandwidth and energy consumption; however, as compression algorithm is relatively complicated and needs more computation time and memory, the energy cost of nodes that apply compression algorithm is much higher than that of nodes without using compression algorithm [12].

Event-based monitoring scenario is a typical application of WMSN. When the triggering condition of an event is met, the wireless multimedia sensor node will capture the image data at that moment and upload the real-time data to the storage device of gateway node. Then, the gateway node collects data from different sensor nodes and transmits the data to the cloud for storage and backup through sink node based on corresponding strategies [13]. In this scenario, the WMS node deployment in WMSN is very intensive and will produce a large amount of data in a very short period. These data have distinctive characteristics so it is suitable for applying data deduplication technology to this scenario. Data deduplication can effectively eliminate the redundant data to reduce the multimedia data traffic between nodes and improve the bandwidth utilization of WMSN. At the same time, its computation complexity is less than that of compression, which means less computation energy consumption. The chunking algorithm is the key factor that impacts the effect of deduplication. Basic sliding window (BSW) algorithm is one of the classical chunking algorithms. This paper proposes a specific chunking algorithm based on BSW. The algorithm can decrease system overhead and increase deduplication efficiency by merging continuous redundant chunks. In contrast with BSW, the experiment results will show how efficient the new algorithm is. The rest of this paper is organized as follows. Section 2 firstly describes the background of data deduplication and introduces BSW algorithm. A continuous redundant chunk-merging algorithm is proposed in Section 2.4; Section 3 performs a simulation experiment and takes the basic sliding window algorithm for comparison to analyze the advantages and disadvantages of both; Section 4 will draw a conclusion.

2. Data Deduplication

Data deduplication [14] is firstly applied in the field of storage backup to reduce storage costs and improve storage space utilization. The technology segments the source data S into a continuous chunk set $C = \{c_1, c_2, \ldots, c_n\}$ according to chunking algorithm. If small changes are made on S, S should be denoted as S' and should be segmented into a new continuous chunk set $C' = \{c'_1, c'_2, \ldots, c'_2\}$ by using the same algorithm. Suppose that S is already in memory; only the chunks in C' but not in C should be stored when saving S'. If $|C \cap C'| \to n$ $(n \leq m)$, that is, most of elements in C' are the same as the elements in C, and then the chunk sequences are regarded as stable. The more stable the sequence is, the smaller the storage cost is. Data deduplication can be applied in WMSN to reduce the amount of data transmission between nodes. If the WMS node is denoted as *sender* and the gateway node is denoted as *receiver*, the *sender* is ready to send the data S to the *receiver*. Assume that the *receiver* already has the data R and the *sender* knows what chunk in S is exactly the same as the chunk in R, so the content of these chunks needs not be transmitted, but only their index in R should be transferred. The *receiver* will find these chunks locally to restructure S according to the index to achieve the goal of saving network bandwidth. It is obvious that the chunking algorithm is important to ensure the stability of chunking sequence.

Data deduplication can be divided into file level, block level, and byte level according to operating granularity. The block level can be divided into fixed-size chunk and variable-size chunk. The smaller the granularity is, the more redundant data are removed, but the implementation complexity and system overhead also increase accordingly. The redundant data that file level can eliminate is the least, but it is easy to achieve. The byte level has the best deduplication effect, but its overhead is extremely expensive and difficult to implement. If fixed-size chunks are used, their matching probability will be greatly reduced due to fixed boundary. In monitoring networks, the position of multimedia sensor is generally fixed, so the images have a small amount of information changes. Good variable-size chunking algorithm can separate duplicate and nonduplicate information as much as possible to reduce the transmission amount of duplicate information. The content-based chunking algorithm applies sliding window technology to determine the boundary of the chunk based on file content. It can control the impact of update on data partitioning within a small range, so that only several chunks near the update location will be affected while other chunks remain unchanged. Therefore, this chunking algorithm is very suitable for the scenario above.

This section will first introduce basic sliding window algorithm and will propose an improved continuous redundant chunk-merging algorithm. Then, the advantages and disadvantages of both methods are analyzed in terms of system overhead.

2.1. Basic Sliding Window Algorithm. The process of basic sliding window algorithm [15] is as follows: a window with fixed size is moved cross the data, and, at every position in the data, a Rabin fingerprint [16] of data in the window is calculated. If an rf-*Match* appears in position k, then k is declared as a chunk boundary. Repeat these steps until it reaches the end of data. Finally, a sequence of chunks $c_1, c_2, \ldots, c_n$ is generated.

Definition 1. A Rabin fingerprint for a sequence of bytes $t_1, t_2, \ldots, t_\beta$ of length β is given by the following expression, where p and M are constant integers:

$$\mathrm{rf}\left(t_1, t_2, \ldots, t_\beta\right) = \left(t_1 p^{\beta-1} + t_2 p^{\beta-2} \cdots + t_{\beta-1} p + t_\beta\right) \mathrm{mod}\ M. \quad (1)$$

Definition 2 (rf-*Match*). For a given sequence $S = s_1, s_2, \ldots, s_n$, rf represents Rabin fingerprint function, l is the window size, and r is a constant integer; then, the rf-*Match* at position k is defined as follows: the last r bits of rf(W) value are zero, in which $W = s_{k-l+1}, s_{k-l+2}, \ldots, s_k$ is the subsequence of length l preceding the position k in S.

2.2. Analysis of Duplicate Data. In order to analyze the redundancy degree of the image data captured from the monitoring network, we simulated a scenario in the laboratory, where a path was left in the middle and fixed objects were placed at both sides. Excluding the impact of light change, a CMOS camera was fixed to monitor the scenario. If any objects appeared on the path, the camera would capture the images at that moment and store them in external storage immediately. We asked different groups of testers to move on the path, and a total of 300 monitored images were collected. All image data were stored in nondestructive bitmap format (BMP; resolution 640∗480; bit depth 16 bits). The file name was numbered in ascending order: $0, 1, \ldots, 299$. The 0th image was the initial scenario.

We made two groups of experiments. Group 1 applied byte comparison tool and took the 0th image as the benchmark to compare it with the remaining 299 images and record the number of bytes at identical parts. The experimental results are shown in Figure 2. It can be seen that, compared to the initial scenario, only the image value on the path is changing in other images, and nearly 78% of pixels in average remain unchanged. We call this part of data as redundant data that can be eliminated, namely, duplicate data.

Group 2 was used to verify the validity of BSW algorithm. In Group 2 experiment, BSW algorithm was used to segment all images and count the frequency that each chunk appeared. Theoretically, if the frequency is greater than 1, this means that the content of this chunk is only required to transmit one time. Before the BSW algorithm is implemented, three parameter values M, r, and l should be determined firstly. M determines the possibility of collision for Rabin fingerprint value, r directly affects the final size of chunk, and l determines the minimum chunk size. We built a hash table in memory to store the content of chunk, the 512 bytes SHA1 of chunk, and the frequency chunk appeared. The BSW algorithm was then implemented. Each time a new chunk was generated, the hash table was queried one time. If there were identical chunks, only one chunk was stored and its frequency was added as 1. If BSW algorithm was applied to segment the byte sequence set $S = \{S_1, S_2, \ldots, S_n\}$, then finally a chunking sequence set $C = \{\{c_{11}, c_{12}, \ldots, c_{1n}\}, \{c_{21}, c_{22}, \ldots, c_{2n}\}, \ldots, \{c_{n1}, c_{n2}, \ldots, c_{nn}\}\}$ was generated. If the frequency of each chunk was $k_{11}, k_{12}, \ldots, k_{1n}, \ldots, k_{n1}, k_{n2}, \ldots, k_{nn}$, respectively, then the duplicate chunk set that could be eliminated in C was represented by $C_{\mathrm{dup}} = \{c_{ij} \mid k_{ij} > 1\}$, and the ratio of duplicate chunk in S was represented by $\delta = |C'_{\mathrm{dup}}|/|C|$.

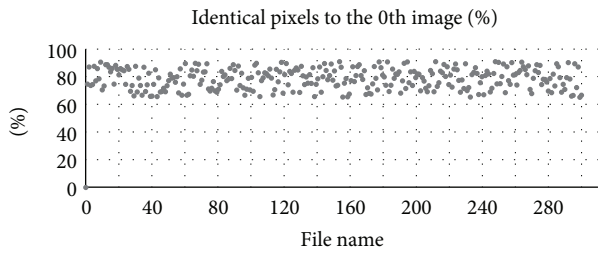

Figure 2: Percentage of identical pixels to the 0th image.

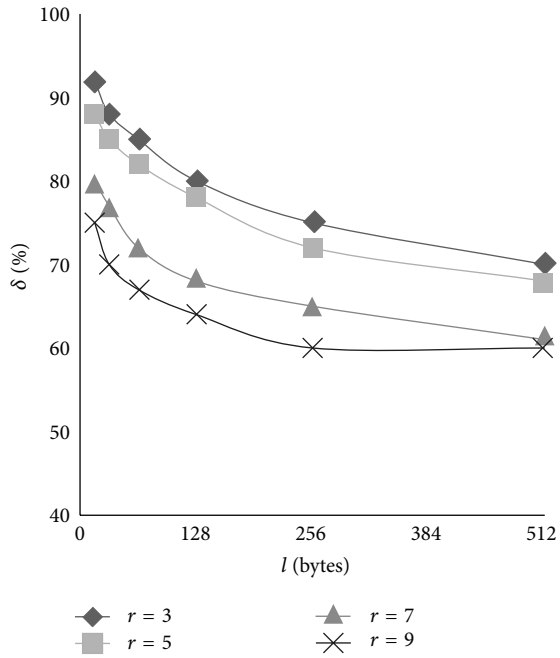

Figure 3: Impact of r and l on the ratio of duplicate chunk in S.

Considering that the sensor memory capacity was limited, we chose $M = 2^{30}$, and the collision probability of fingerprint at this length was small. Figure 3 shows different impact of r and l on δ, and, the same as in theoretical analysis, larger r and longer l led to the increase of chunk size and the decrease of chunk number, finally resulting in the decline of δ. However, the excess reduction of r and l would cause extra system overhead. In the experiment, we allocated 20-megabyte memory to hash table. The impact of r and l on the elimination rate of hash table is shown as in Figure 4.

The results show that, because the capacity of hash table was limited, excessive chunks would lead to the frequent elimination of hash table items, thus increasing the system overhead. Therefore, a more balanced choice was $r = 5$ and $l = 64$ bytes. The chunk statistic results with these parameter values are shown in Table 1. In a follow-up experiment, the two parameters selected these two values.

2.3. Evaluation Indicators. In this section, we introduce an evaluation indicator mentioned in [17] for evaluating our works.

For two byte sequences P and Q, use PQ to represent their concatenation. If $PQ = S$, then P is the prefix of S and Q is the postfix of S. $|S|$ represents the length of S in bytes.

For a byte sequence S, $H(S)$ denotes the chunk sequence $c_1, c_2, \ldots, c_n$ generated by a chunking algorithm on S. $\mu(S)$ = (total size of all chunks/the number of chunks) denotes the average chunk size in $H(S)$.

For the given threshold t, if (shared $(S, S')/|S| > t$), then S' is a local modification sequence of S, in which shared (S, S') refers to the sum of bytes with the same prefix and postfix of S and S', and it is defined that new $(S, S') = |S'| - \text{shared}(S, S')$.

$\Delta(S, S')$ is defined as the total size of chunks that appear in $H(S')$ but not in $H(S)$.

Definition 3. Assume that S is a byte sequence; S' is the local modification sequence of S. $V(S)$ is defined as follows: $\Delta(S, S') - \text{new}(S, S')$.

Definition 4. For a defined $V(S)$, the algorithm overhead index α is defined as follows:

$$\alpha = \frac{V}{\mu}. \tag{2}$$

With chunking algorithm, if duplicate data and nonduplicate data are divided into one chunk, the duplicate data in this chunk is an extra cost because it should not be stored and transmitted. V is the average number of bytes used to evaluate the overhead. The parameters r and l in Definition 2 can be adjusted to make V smaller, but this will also increase the metadata overhead, so μ is introduced to balance the overall cost. μ is the average chunk size which is inversely proportional to the metadata overhead and proportional to V. By calculating the ratio of V and μ, a comprehensive evaluation on chunking algorithm and cost of metadata can be achieved.

2.4. Continuous Redundant Chunk-Merging Algorithm. The chunks created by the BSW algorithm in Section 2.2 are not equal in size. The chunk size has very large impact on the algorithm effect, so we try to improve BSW algorithm by adjusting the chunk size. Group 2 experiment in Section 2.2 has recorded all chunks sizes. 79% chunk size is less than the average chunk size. If smaller chunks can be merged to reduce the total number of chunks, the average chunk length μ will increase; thus the overhead index α will decrease. For our simulation scenario, most of the pixels are basically unchanged. BSW algorithm is a content-based chunking algorithm, so the boundary between chunks is stable. For example, Figures 5(a) and 5(b) shows two continuous image files. The nonduplicate bytes part concentrates in one area, and the chunk boundary does not change in the first and the last duplicate bytes parts, so we declare that the chunk boundary is stable. If we merge the first three chunks and the last three chunks in Figures 5(a) and 5(b), respectively, total number of chunks will drop from 20 to 12, average chunk size μ will rise by 66%, and overhead index α will reduce by 39.7%.

We propose a continuous redundant chunk-merging (CRCM) algorithm. The algorithm works as follows: first, to determine the chunk boundary by applying BSW algorithm on files. Each time a new chunk is generated, the hash table is queried to determine whether the chunk has been stored before. If stored, the chunk will be sent to a data stack. If not stored, the algorithm continues to look for the next chunk and then repeat the above steps until meeting the end of

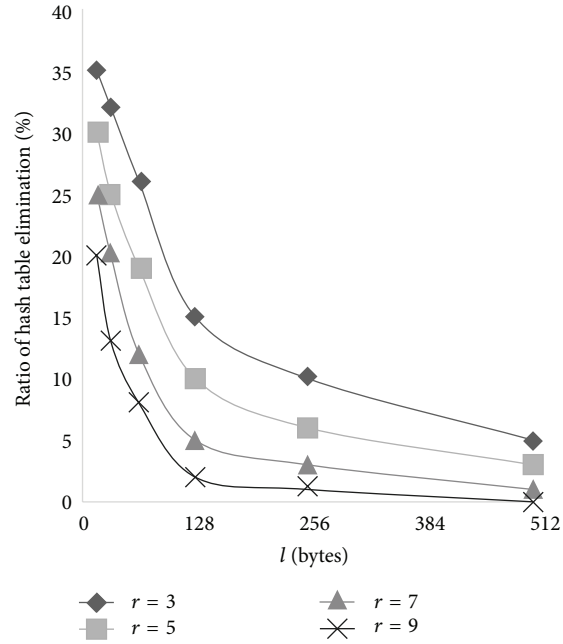

Figure 4: Impact of r and l on elimination rate of hash table.

Table 1: Chunk statistics of Group 2 experiment ($r = 5$ and $l = 64$ bytes).

Stat. item	Result
Total number of chunks	10068
Number of stored chunks	2395
Number of redundant chunks	8273
Percentage of redundancy	82.17%
Average chunk size (ACS)	19 K Bytes
Number of chunks whose size > ACS	2115
Number of chunks whose size < ACS	7953
Maximum chunk size	25 K Bytes
Minimum chunk size	1 K Bytes

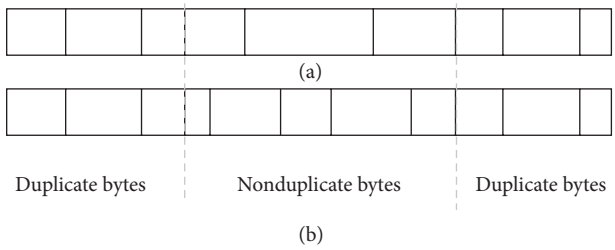

Figure 5: Example of stable chunk boundary.

file. The algorithm merges chunks according to the following rules: if the total length of chunk in the stack is greater than the threshold L, then merge all chunks in the stack to form a new chunk and insert it into the hash table. The detection of chunk merging is performed when a new chunk is generated.

Because the chunk boundary is stable, the probability of merging the same continuous redundant chunks is reasonably great when segmenting different files by applying CRCM algorithm. The algorithm effect is related to the threshold L and metadata costs. These will be discussed through the experiment in Section 3.

3. Result and Discussion

When wireless multimedia sensor is used for experiments in the simulation scenario, the following questions should be considered:

(i) hardware specification of WMS,

(ii) construction of wireless multimedia monitoring network,

(iii) data synchronization between nodes.

Computing power, storage capacity, and battery life of WMS have been improved greatly. The energy consumption of WMS is mainly in data processing, storage access, image acquisition, network communication, and so forth. Stargate [18] is a product of MEMSIC, and its hardware specification is Intel PXA-255Xscale 400 MHz CPU, 64 MB SDRAM, and 32 MB flash storage. Meantime, it also supports embedded Linux operating system and IEEE802.11 wireless communication protocol. Therefore, it is enough to complete some complicated computing tasks.

The wireless multimedia monitoring network composed of multiple WMS independently collects data and transmits information to sink node through gateway node. The computing power and storage capacity of gateway node are both far greater than those of WMS nodes. Multiple WMS can eliminate duplicate data together with gateway node at the same time, and the effect will be better, but this brings about a new problem of data synchronization between nodes. Some proposed synchronization methods such as distributed data deduplication [19] can address this problem.

This section will continue the experiment in the simulated scenario in Section 2.2. The chunking algorithm was replaced with CRCM algorithm, and then the evaluation was performed in terms of overhead index and PDR.

The WMS nodes in the experiment adopted the engineering board based on ARM9 architecture, equipped with 200 MHZ CPU, 64 M SDRAM, 1 GB flash storage, and onboard *QuickCam* camera. The maximum image resolution was 640 × 480, and the operating system was embedded Linux (2.4.19). Gateway node used DELL VOSTRO 3560 laptop, specification was the following: Intel i7-3632QM 2.2 GHz, 8 G RAM, and 250 GB hard of which drive. WMS node and gateway node used the IEEE802.11 wireless protocol to communication.

In the experiment, WMS node was used to monitor the simulation scenario. Any changes in the scenario would trigger sensors to capture and transmit real-time information to gateway node. CRCM algorithm was used for image chunking before the transmission of WMS node. Only those nonduplicate data chunks were transmitted. For duplicate chunks, only their index values were transmitted. The details of data synchronization between nodes would not be mentioned here.

Firstly, we observe the impact of threshold L on overhead index by running CRCM algorithm (see Figure 6). L is the multiple of the average chunk size of current file, that is, $L = x \cdot \mu$. When $L = 0$, CRCM algorithm was degenerated into the BSW algorithm and $\alpha \approx 2.5$. Greater L resulted in more merged chunks. When $L/\mu = 1.4$, the overhead index was the best, $\alpha \approx 1.3$, and the cost reduced by about 23.8%. As L continued to increase, the overhead was not reduced any longer. The reason was that if excessive chunks were merged, chunk size would be large, thus increasing the V value in α.

Metadata cost is another system overhead, so it must be involved in our evaluation. We focused on the metadata storage cost. WMS node's storage capacity is limited, so duplicate chunk data cannot be stored locally forever. Therefore, local metadata storage will encounter the problem of hash table elimination. We conducted four groups tests on different memory size allocated for hash table. The results were shown in Figure 7, and we can find that the ratio of hash table elimination by running CRCM was less than that of BSW algorithm, because BSW algorithm generated more chunks which needed more hash table items. According to our WMS hardware capacity, we considered 20 M would be a better choice in this scenario.

The computation complexity of both BSW algorithm and CRCM algorithm is much less than that of compression. The most time-consuming calculation is Rabin fingerprint calculation. Recalling rf in Definition 1, for fast execution, we can define an iterative formula as follows:

$$\begin{aligned} \mathrm{rf}\left(t_{i+1} \cdots t_{\beta+i}\right) = \left(\mathrm{rf}\left(t_i \cdots t_{\beta+i-1}\right) - t_i \cdot p^{\beta}\right) \cdot p \\ + t_{\beta+i} \mod M, \end{aligned} \tag{3}$$

where $t_i \cdot p^{\beta}$ can be precomputed and stored in a table. Rather than generating a new fingerprint from scratch, advancing rf calculation only requires three operations: a subtraction, a multiplication, and a mask mod.

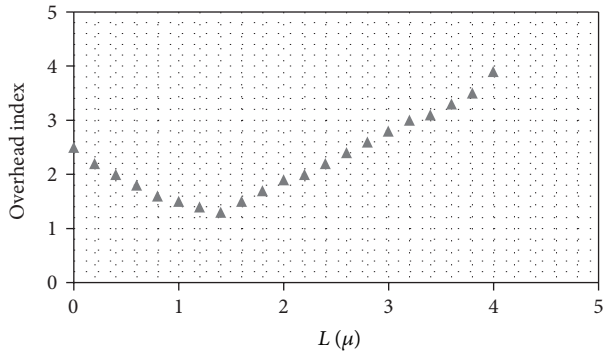

Figure 6: Impact of L on overhead index.

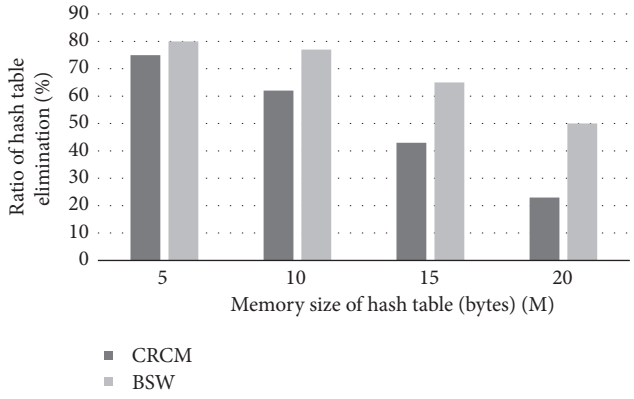

Figure 7: Ratio of hash table elimination by running CRCM versus BSW on different memory size.

4. Conclusion

In wireless multimedia sensor networks, massive data needs to be transmitted to the cloud storage. This paper applies data deduplication technology to WMSN. In the simulated monitoring scenario, this paper puts forward a continuous redundant chunk-merging algorithm by taking the characteristics of scenario data into full consideration. The algorithm considers the computing power and storage capacity of sensor hardware. Compared with BSW algorithm, it effectively reduces system overhead through experiments.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61373015 and 61272422), The Research Fund for the Doctoral Program of High Education of China (Grant no. 20103218110017), the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), and the Fundamental Research Funds for the Central Universities, NUAA (Grant nos. NP2013307 and NZ2013306).

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] Tubaishat, Malik, and S. Madria, "Sensor networks: an overview," *IEEE Potentials*, vol. 22, no. 2, pp. 20–23, 2003.

[3] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 921–960, 2007.

[4] F. Hu and S. Kumar, "QoS considerations in wireless sensor networks for telemedicine," in *Internet Multimedia Management Systems IV*, vol. 5242 of *Proceedings of SPIE*, pp. 217–227, November 2003.

[5] J. Campbell, P. B. Gibbons, S. Nath, P. Pillai, S. Seshan, and R. Sukthankar, "IrisNet: an internet-scale architecture for multimedia sensors," in *Proceedings of the 13th Annual ACM International Conference on Multimedia*, pp. 81–88, New York, NY, USA, 2005.

[6] J. N. Al-Karaki, R. Ul-Mustafa, and A. E. Kamal, "Data aggregation and routing in wireless sensor networks: optimal and heuristic algorithms," *Computer Networks*, vol. 53, no. 7, pp. 945–960, 2009.

[7] Z. Lu, Y. Wen, R. Fan, S. Tan, and J. Biswas, "Toward efficient distributed algorithms for in-network binary operator tree placement in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 743–755, 2013.

[8] A. Ciancio and A. Ortega, "A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting," in *Proceeding of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, pp. 825–828, March 2005.

[9] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *Proceeding of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 1–10, April 2004.

[10] F. Marcelloni and M. Vecchio, "An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks," *Computer Journal*, vol. 52, no. 8, pp. 969–987, 2009.

[11] Liang, Yao, and Wei Peng, "Minimizing energy consumptions in wireless sensor networks via two-modal transmission," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 12–18, 2010.

[12] K. C. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Transactions on Computer Systems*, vol. 24, no. 3, pp. 250–291, 2006.

[13] S. Misra, M. Reisslein, and G. Xue, "A survey of multimedia streaming in wireless sensor networks," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 4, pp. 18–39, 2008.

[14] Q. He, Z. Li, and X. Zhang, "Data deduplication techniques," in *Proceeding of the International Conference on Future Information Technology and Management Engineering (FITME '10)*, vol. 1, pp. 430–433, Changzhou, China, October 2010.

[15] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, 2001.

[16] Rabin and O. Michael, "Fingerprinting by random polynomials," Center for Research in Computing Technology, Aiken Computation Laboratory, 1981.

[17] K. Eshghi and K. T. Hsiu, "A framework for analyzing and improving content-based chunking algorithms," Hewlett-Packard Labs Technical Report TR 30, 2005.

[18] M. O. Farooq and T. Kunz, "Wireless multimedia sensor networks testbeds and state-of-the-art hardware: a survey," *Communications in Computer and Information Science*, vol. 265, no. 1, pp. 1–14, 2011.

[19] Y. Zhang, Y. Wu, and G. Yang, "Droplet: a distributed solution of data deduplication," in *Proceeding of the ACM/IEEE 13th International Conference on Grid Computing (ICGC '12)*, pp. 114–121, Beijing, China, 2012.