Deduplication Analyses of Multimedia System Images

Tim Süß
Johannes Gutenberg University Mainz, Institute of Computer Science
Tunahan Kaya, Markus Mäsker, André Brinkmann
Johannes Gutenberg University Mainz, Zentrum für Datenverarbeitung

Abstract

The availability and usage of embedded systems increases permanently and the industry drives the IoT to become more and more relevant in daily life. Factory lines, planes and cars, traffic lights, or even clothes are equipped with sensors and small computers constantly communicating with the outside world. One challenge in maintaining those devices is updating their software. Due to slow connections or only because of the huge amount of devices data transfers can be problematic.

Data compression algorithms can be applied to reduce the amount of data that must be transferred. A data reduction technique that provides high efficiency, but which has not been considered so far for embedded systems is *data deduplication*. In this work we present the results of a long term study for updating a car multimedia system. The results show that deduplication can achieve significantly better results than commonly used data compression techniques.

1 Introduction

Software maintenance in general is a challenging and costly task [1, 8, 13]. Usually update techniques can be classified into two groups: *package*-driven and *image*-driven. While the package-driven approach replaces individual components and their dependencies, the image-driven approach replaces larger system parts.

Packages upgrades can become difficult for complex projects in general [5] and for client driven package managers [21, 31] in particular. Identifying conflicts and resolving dependencies among packages is a complex task [31] and might not be a suitable approach for clients with limited resources such as embedded devices.

Instead of client driven package managers, embedded system firmware can be upgraded by flashing full system images [10]. To reduce the image size different approaches have been presented [15,22,26], which in some cases also try to minimize the energy consumption during updates [18], [14].

In this paper we present an approach that takes advantage of the combination of both techniques. It can be used to update full images, while it transfers only those parts that have been modified. The technique is tailored for updating large data sets on systems that offer only low computational power and/or limited bandwidth.

The approach bases on *data deduplication* which is usually used for backups. It reduces the amount of data that must be transferred by exploiting redundancies between successive images. Data is therefore partitioned into similar sized chunks and a cryptographically secure fingerprint is computed for every chunk. This fingerprint can identify whether a chunk is new or if it was stored before in a previous image. When a new chunk is identified the fingerprint is added to an index and the chunk is persistently stored. However, chunking requires much computational efforts.

We analyze the feasibility of deduplication for software updates of IoT-systems and focus on updating automotive multimedia systems. Nowadays those systems have many responsibilities, such as navigation, entertainment, connectivity, and climate control (see Fig. 1). Instead of relying on the update procedures of those individual components, we present an universal image-driven approach.

Figure 1: A modern car multimedia system is connected to the vendor and its content is regularly updated.

Updating the software image of multimedia systems in cars using deduplication includes a central server, edge computing devices and the car itself. The central server provides the image, which can already be broken up into chunks. The edge computing servers only have to transfer data, which is not yet available in the car. It therefore applies a simple protocol, where the complete image is processed chunk by chunk and for each chunk, there is a short handshake between the car and the edge computing server, whether car already stores the chunk.

We collected two years of updates for a Debian-based Linux multimedia system whose navigation software uses *Open Street Maps* to evaluate the update process. We have chosen Debian since it is the base for many SoC linux distributions (e.g. Raspbian, Ubuntu, Bananian). Every week we downloaded a new software version and a map of a fixed sized region. The complete system occupied about 4.9 GiB storage in the beginning, while it has grown to 5.4 GiB at the end of our experiment. One challenge for the deduplication is the heterogeneity of the data. This means that the system without the map has directly shown good deduplication behavior while the map's properties had a negative effect on data reduction. By removing the origins of these effects we reduced the amount of data to be transferred on average by a factor of 3.7 in comparison to LZMA compressed data. Experiments with rsync using delta updates have shown that this application is unsuitable for our needs. A single update transfers more than twice as much data as our approach, which also shows that deduplication is applicable for system updates.

2 Background

The following sections provide the background about system image updates and deduplication techniques.

2.1 Update Encoding And Distribution

Embedded systems usually operate on limited resources. This requires system updates to be encoded, transmitted and installed in a resource-efficient manner. TinyOs, a popular operating system for wireless sensor networks, comes with an update protocol named *Deluge* [10]. Deluge can be used to disseminate large data objects and thereby replace system images of embedded devices. However, diff-like algorithms for embedded systems [15,22,24–26] are more memory and bandwidth efficient and minimizing the number of (flash) writes is beneficial on systems with limited energy resources [25].

Byte Pair Encoding (BPE) [16] is a lossless compression technique presented by Kiyohara et al. BPE replaces common byte sequences with single byte representatives that did not occur in the original data. The low decompression effort is advantageous for embedded devices operating on limited resources.

Another issue in sensor networks is to enable system upgrades without downtime. Koshy and Pandey propose remote incremental linking for energy efficient reprogramming of sensor networks [17]. Zephyr uses a modified rsync diff approach to reduce data size [25] and Elon [6] uses *replaceable components* to minimize downtime during updates.

2.2 Data Deduplication

A major issue for data deduplication environments is to overcome the *fingerprint-lookup disk bottleneck* [19], which is typically achieved by combinations of caching, bloom filters, and locality-preserving data structures [11]. There are two deduplication approaches: exact (e. g., used in DDFS [34] or BLC [23]) and approximate deduplication (e. g., Sparse Indexing [19]). In contrast to exact deduplication, where every chunk is stored exactly once [27, 32], chunks might be stored redundantly in the approximate method [2, 9, 28, 33].

One challenge in data deduplication is the identification of chunks that have been parts of previous images. Usually chunks do not have a static size, since already small insertions would affect all following chunks. Instead, the data is screened for bit sequences with special properties that are expected to appear in regular distances. In these places the data is cut into pieces. For this so-called *content defined chunking* algorithms like Rabin-fingerprinting are used [3].

Tan et al. present a causality-based backup system for clouds called *CABdedupe* [29]. For reducing the data transferred clients require information about previous backup runs. Kaiser et al. present an approach designed for deduplicating thousands of data streams in parallel on the same fingerprint index [12]. The approach creates a new locality by sorting the fingerprints and thereby it achieves that the backup server can always access the index sequentially.

Several systems exploit the similarity of different backup data [2, 7, 30, 33] using Broder's theorem [4] and comparing the smallest fingerprints of two sets of fingerprints. Douglis et al. [7] extend this idea; they sort both sets and compare the n smallest fingerprints of each set. In our system we also sort fingerprints, but for the purpose of avoiding I/O operations on the server.

3 Methodology

Modern cars are highly connected and can include a multitude of client applications. The software is a crucial and safety critical component of today's *drive-by-wire*

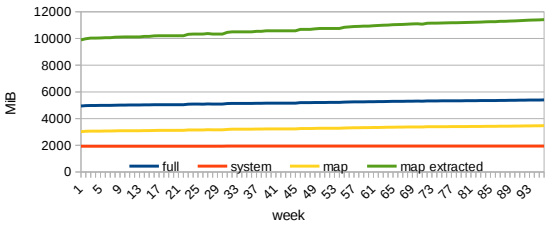


Figure 2: Development of the image size over time.

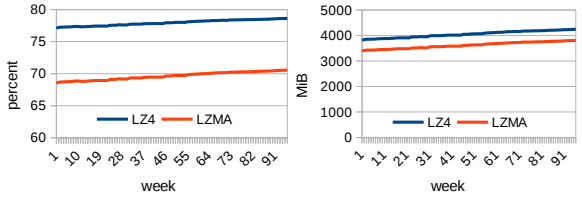
cars and upcoming autonomous vehicles. Being able to guarantee the correctness of over-the-air (OTA) upgrades could become an obligation imposed by insurance companies or even a legal requirement [20]. A full system image update has the advantage of unifying the update procedures of those clients and to deploy bit-exact images, which can significantly reduce testing overheads.

For our experiments we chose Debian 8.0 as Linux distribution. We installed a minimal system and extended it by the display manager *Mate*, the video system VLC, and the navigation system OSM. The initial size of this system was 1.923 GiB. Additionally, we installed Open Street Map for Germany with a 3.021 GiB compressed map (zip format) and 9.898 GiB when it was extracted. Every week we performed an update of the complete system: the OS, the installed applications, and the map. Once the updates were performed, we created a system backup using tar. The sizes of the image (full), the system part (system), the compressed map (map), and the extracted map (map extracted) can be seen in Figure 2.

While the system image size merely grew to 1.935 GiB (see Figure 2), the map size increased significantly. The volume of the compressed map grew to 3.465 GiB and to 11.415 GiB for the extracted map.

We analyzed if the image sizes could be reduced by compressing the data. In our tests we used LZ4 with the option -9 (best compression) and the program xz (using LZMA) with the option -e. The compression ratios of both programs are illustrated in Figure 3a and the sizes of the compressed images are shown in Figure 3b.

The size of this compressed images is the base line for our tests since the transferred data volume of the deduplication must be lower than the compressed data.



(a) compression ratio (b) compressed image size

Figure 3: Properties of the compressed data.

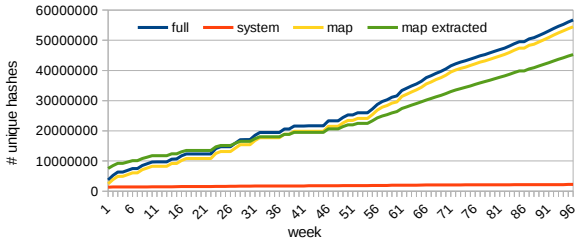
4 Evaluation

In this section we present our evaluation. First we show how the number of unique fingerprints increases in the successive update generations. The growth of the index storing the chunks' fingerprints indicates the amount of data that must be transferred. Furthermore, we show the negative impact of compression on deduplication.

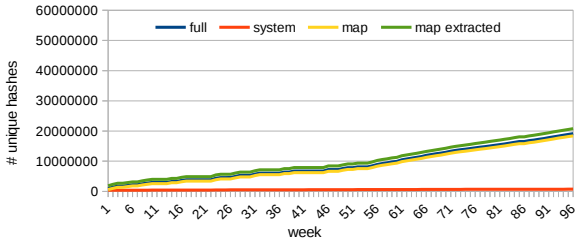
In the real world it might happen that an update is skipped. In this case more data must be transferred during the next update. We analyze the impact on the deduplication ratio when one or three updates are skipped.

4.1 Data Properties

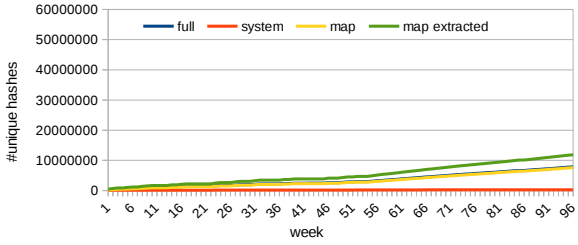
In our first experiment we analyzed the growth of the fingerprint index for different chunk sizes. Therefore, we partitioned the data set into chunks of different average sizes using Rabin's *content defined chunking* algorithm. We configured the algorithm so that the minimal chunk size was 50 % smaller and the maximum size was 50 % larger than the aimed average. Figures 4a-4c show the growth of the indices.



(a) 1 KiB chunks



(b) 4 KiB chunks



(c) 16 KiB chunks

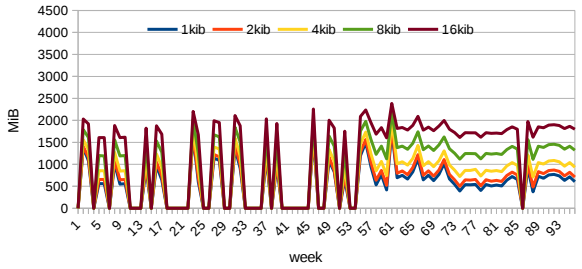
Figure 4: Growth of the fingerprint index

These diagrams indicate that the system has been mod-

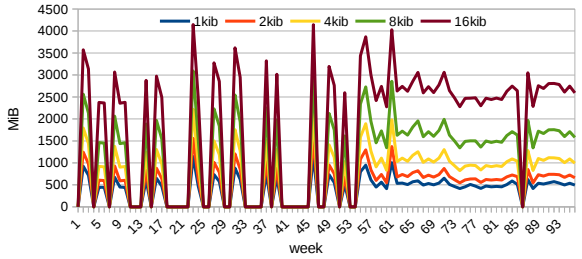
ified rarely. The amount of data that must be transferred for this component is negligible in comparison to the map content. The number of full fingerprints for the complete image is (almost) the same as the sum of the fingerprints for the compressed map and the system.

The diagrams also show the impact on the index size when the compressed map is extracted and stored in a single file. At a chunk size of 4 KiB more fingerprints are produced for the extracted map than the compressed counterpart (see Figure 4b). The difference is (almost) constant for this chunk size, but this changes for varying chunk sizes. The extracted map fingerprint number grows significantly faster than it does for the compressed map when 16 KiB chunks are used (see Figure 4c). However, when the chunk size is decreased fewer fingerprints are produced for the compressed map only for the first updates. Quickly the extracted map generates fewer fingerprints than its compressed counterpart as the crossing of the lines in Figure 4a illustrates.

The index growth is an indicator for the data that must be transferred per update. Since the system part behaves nicely for all chunk sizes we focus only on the compressed and extracted map. For this we multiplied the number of new fingerprints with the average chunk size. We did this for every update and display the results in Figures 5a and 5b.



(a) Compressed data



(b) Extracted map data

Figure 5: Transferred data

The diagrams illustrate that the transferred data volume can be reduced if the map is extracted and only 1 KiB chunks are used. That reduces the transferred data volume by 33 %. However, whenever large chunks are required the data should not be extracted. For different chunk sizes Figure 6 shows how much data can be saved

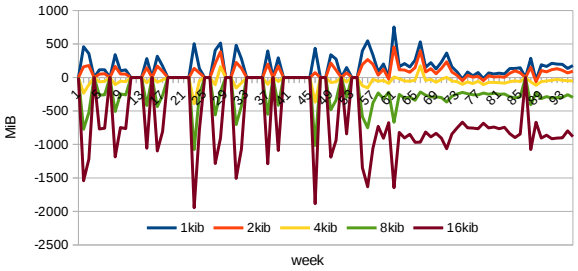


Figure 6: Transfer savings when extracted data is used for deduplication.

or must be additionally transferred if an extracted map is used instead of a compressed one. On the one hand it is possible to save up to 750 MiB when uncompressed maps at a chunk size of 1 KiB are used. On the other hand additional 2,000 MiB of data must be transferred when 16k chunks are used instead. Rabin's algorithm and small chunks are better suited to determine redundancies than large chunks.

Although metadata handling is a major challenge in data deduplication systems, this only applies to the server side of our system. In contrast to traditional backup scenarios, where full files or complete systems need to be recovered, our clients update existing files. Instead of a so called *file receipt*, clients could receive an efficiently encoded patch-like data structure.

Next we determine the efficiency of the different chunkings. In our experiment the smallest fingerprint index size is achieved with the smallest chunk size. To determine the efficiency of the other chunk sizes, we compute the ratio of the index size when 1 KiB chunks are used to the index size when another chunk size x is used. For every week i we calculate:

$$eff_i = \frac{1\,\text{KiB} \cdot \#1\,\text{KiB hashes in week i}}{x\,\text{KiB} \cdot \#\text{x KiB hashes in week i}}, x \in \{1,2,4,8,16\}$$

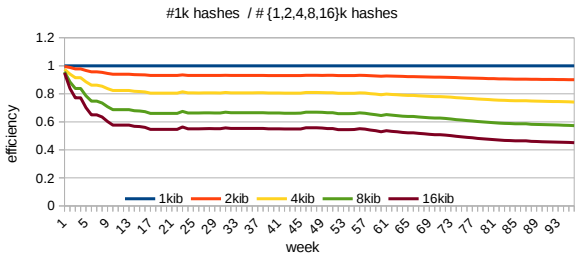


Figure 7: Efficiency of the different chunk sizes.

An average chunk size of 1 KiB has a constant efficiency of 1 by definition. That neither of the other chunk sizes accomplish a higher efficiency indicates that 1 KiB chunks achieve the lowest update size. Furthermore, the graph in Figure 7 shows that the larger the chunks the less

efficient they are. This follows the expected behavior of deduplication.

4.2 Skipping Updates

There are different situations when an update cannot be performed (e.g., when there is no connection). This influences the data that must be transferred. In our next experiment we show the impact on the required data transfers when the extracted maps are only updated every second respectively fourth week. We only use chunk sizes of 1 KiB, 2 KiB, and 4 KiB because only with these values data could be saved in the previous experiments.

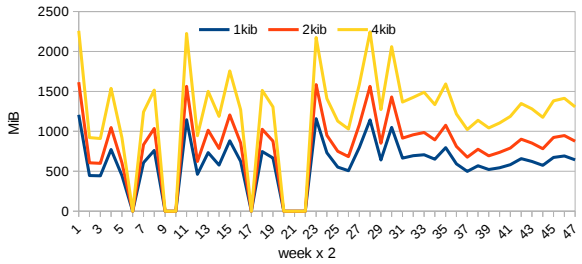


Figure 8: Transferred extracted data on 2-weekly update cycles.

The comparison of Figure 8 and Figure 5b shows that more data must be transferred in all cases. While for 1 KiB chunks and a weekly update only about 500 MiB were transferred this value was rarely achieved for a bi-weekly update frequency. Nevertheless, in comparison to the compressed maps an equivalent amount of transferred data is saved (see Figure 9).

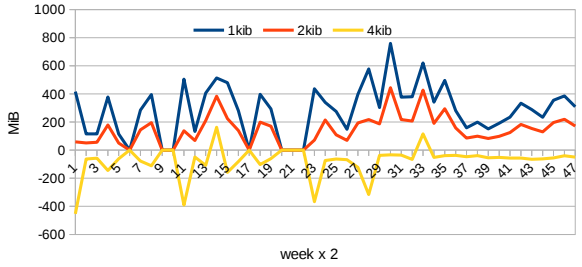


Figure 9: Transfer savings when extracted data is deduplicated on 2-weekly update cycles.

Finally, we evaluate the effects when updates are performed every four weeks instead of every week. As expected the transferred data volume increases because of the bigger differences (see Figure 10). When the frequency is reduced, more data must be transferred during an update and therefore, also the savings for the average chunk sizes of 1 KiB and 2 KiB increase (see Figure 11). Consequently, the larger 4 KiB chunks suffer a larger penalty.

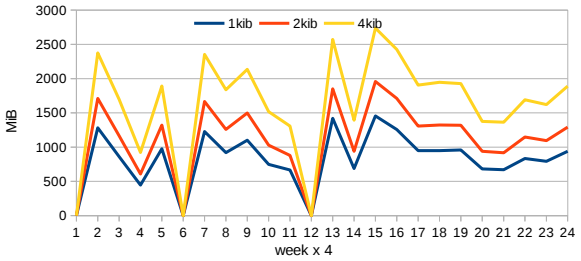


Figure 10: Transferred extracted data on 4-weekly update cycles.

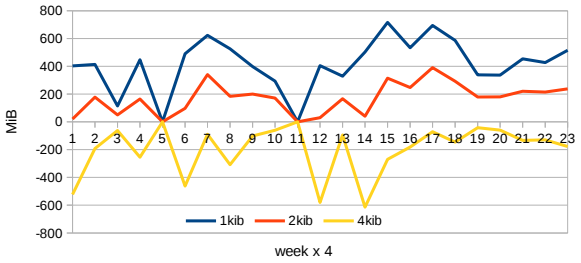


Figure 11: Transfer savings when extracted data is deduplicated on 4-weekly update cycles.

The results of our experiments show, that there can be a benefit when extracted data is transferred instead of compressed. Smaller chunks can compensate the penalties of a bigger real (logical) data volume. The disadvantage of small chunk sizes is a larger fingerprint index and increased effort of identifying unique fingerprints. However, those are offline server-side costs and not relevant for the clients.

5 Conclusion

We have shown that deduplication is a suitable technique for updating systems. In comparison to other techniques, such as packaging or replacing complete images, deduplication requires fewer data transfers and less computational power. Operating on small chunk sizes, our experiments have shown that only a few hundred megabytes must be transferred per update instead of several gigabytes. For rolling updates, typically used compression techniques cannot compete with the data reduction of deduplication. Additionally, we observed that compressed content limits the effectiveness of our approach. The reason for this is loss of redundancies between successive updates. Therefore, data should be decompressed and transferred in smaller chunks. In the future, we will develop a real environment for updating huge numbers of clients. Using this system, the impacts of varying bandwidths and client numbers can be analyzed.

The used data set is still updated and will be made available at `https://research.zdv.uni-mainz.de`.

References

[1] BENNETT, K. H., AND RAJLICH, V. T. Software maintenance and evolution: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (New York, NY, USA, 2000), ICSE '00, ACM, pp. 73–87.

[2] BHAGWAT, D., ESHGHI, K., LONG, D. D. E., AND LILLIBRIDGE, M. Extreme binning: Scalable, parallel deduplication for chunk-based file backup. In *Proceedings of the 17th IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS)* (2009), pp. 1–9.

[3] BRODER, A. Z. Some applications of rabin's fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science* (1993), vol. 993, p. 143152.

[4] BRODER, A. Z. On the resemblance and containment of documents. In *Proceedings of the Conference on Compression and Complexity of Sequences* (1997), IEEE, pp. 21–29.

[5] COSMO, R. D., ZACCHIROLI, S., AND TREZENTOS, P. Package upgrades in FOSS distributions: Details and challenges. In *Proceedings of the 1st ACM Workshop on Hot Topics in Software Upgrades, HotSWUp 2008, Nashville, TN, USA, October 20, 2008.* (2008).

[6] DONG, W., LIU, Y., WU, X., GU, L., AND CHEN, C. Elon: enabling efficient and long-term reprogramming for wireless sensor networks. In *SIGMETRICS 2010, Proceedings of the 2010 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, New York, USA, 14-18 June 2010* (2010), pp. 49–60.

[7] DOUGLIS, F., BHARDWAJ, D., QIAN, H., AND SHILANE, P. Content-aware load balancing for distributed backup. In *Proceedings of the 25th Large Installation System Administration Conference (LISA)* (2011).

[8] GERACE, T. A., AND MOUTON, J. The challenges and successes of implementing an enterprise patch management solution. In *Proceedings of the 32nd Annual ACM SIGUCCS Conference on User Services 2004, Baltimore, MD, USA, October 10-13, 2004* (2004), pp. 30–33.

[9] GUO, F., AND EFSTATHOPOULOS, P. Building a highperformance deduplication system. In *Proceedings of the USENIX Annual Technical Conference (ATC)* (2011), USENIX, pp. 25–25.

[10] HUI, J. W., AND CULLER, D. E. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, Baltimore, MD, USA, November 3-5, 2004* (2004), pp. 81–94.

[11] KAISER, J., BRINKMANN, A., SÜSS, T., AND MEISTER, D. Deriving and comparing deduplication techniques using a model-based classification. In *Proceedings of the Tenth European Conference on Computer Systems (EuroSys), Bordeaux, France, April 21-24, 2015* (2015), pp. 11:1–11:13.

[12] KAISER, J., SÜSS, T., NAGEL, L., AND BRINKMANN, A. Sorted deduplication: How to process thousands of backup streams. In *32nd Symposium on Mass Storage Systems and Technologies, MSST 2016, Santa Clara, CA, USA, May 2-6, 2016* (2016), pp. 1–14.

[13] KEMERER, C. F. Software complexity and software maintenance: A survey of empirical research. *Annals of Software Engineering 1*, 1 (Dec 1995), 1–22.

[14] KIM, J., AND CHOU, P. H. Remote progressive firmware update for flash-based networked embedded systems. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design* (New York, NY, USA, 2009), ISLPED '09, ACM, pp. 407–412.

[15] KIM, J., AND CHOU, P. H. Energy-efficient progressive remote update for flash-based firmware of networked embedded systems. *ACM Trans. Design Autom. Electr. Syst. 16*, 1 (2010), 7:1–7:26.

[16] KIYOHARA, R., MII, S., MATSUMOTO, M., NUMAO, M., AND KURIHARA, S. Method for fast compression of program codes for remote updates in embedded systems. In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC), Honolulu, Hawaii, USA, March 9-12, 2009* (2009), pp. 1683–1684.

[17] KOSHY, J., AND PANDEY, R. Remote incremental linking for energy-efficient reprogramming of sensor networks. In *Wireless Sensor Networks, Second European Workshop, EWSN 2005, Istanbul, Turkey, January 31 - February 2, 2005, Proceedings* (2005), pp. 354–365.

[18] LI, W., ZHANG, Y., YANG, J., AND ZHENG, J. UCC: update-conscious compilation for energy efficiency in wireless sensor networks. In *Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation, San Diego, California, USA, June 10-13, 2007* (2007), pp. 383–393.

[19] LILLIBRIDGE, M., ESHGHI, K., BHAGWAT, D., DEOLALIKAR, V., TREZIS, G., AND CAMBLE, P. Sparse indexing: Large scale, inline deduplication using sampling and locality. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST)* (2009), pp. 111–123.

[20] LITTKE, T. Key legal issues: Automotive over-the-air updates. Tech. rep., ATS Advanced Telematic Systems GmbH Germany, October 2017.

[21] MANCINELLI, F., BOENDER, J., COSMO, R. D., VOUILLON, J., DURAK, B., LEROY, X., AND TREINEN, R. Managing the complexity of large free and open source package-based software distributions. In *21st IEEE/ACM International Conference on Automated Software Engineering (ASE 2006), 18-22 September 2006, Tokyo, Japan* (2006), pp. 199–208.

[22] MARRÓN, P. J., GAUGER, M., LACHENMANN, A., MINDER, D., SAUKH, O., AND ROTHERMEL, K. Flexcup: A flexible and efficient code update mechanism for sensor networks. In *Wireless Sensor Networks* (Berlin, Heidelberg, 2006), K. Römer, H. Karl, and F. Mattern, Eds., Springer Berlin Heidelberg, pp. 212–227.

[23] MEISTER, D., KAISER, J., AND BRINKMANN, A. Block locality caching for data deduplication. In *6th Annual International Systems and Storage Conference (SYSTOR), Haifa, Israel - June 30 - July 02, 2013* (2013), pp. 15:1–15:12.

[24] PANTA, R. K., AND BAGCHI, S. Hermes: Fast and energy efficient incremental code updates for wireless sensor networks. In *INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil* (2009), pp. 639–647.

[25] PANTA, R. K., BAGCHI, S., AND MIDKIFF, S. P. Efficient incremental code update for sensor networks. *TOSN 7*, 4 (2011), 30:1–30:32.

[26] REIJERS, N., AND LANGENDOEN, K. Efficient code distribution in wireless sensor networks. In *Proceedings of the Second ACM International Conference on Wireless Sensor Networks and Applications, WSNA 2003, San Diego, CA, USA, September 19, 2003* (2003), pp. 60–67.

[27] RHEA, S., COX, R., AND PESTEREV, A. Fast, inexpensive content-addressed storage in Foundation. In *Proceedings of the USENIX 2008 Annual Technical Conference (ATC)* (2008), USENIX.

[28] SIMHA, D. N., LU, M., AND CHIUEH, T. A scalable deduplication and garbage collection engine for incremental backup. In *Proceedings of the 6th Annual International Systems and Storage Conference (SYSTOR)* (2013), p. 16.

[29] TAN, Y., JIANG, H., FENG, D., TIAN, L., AND YAN, Z. CABdedupe: A Causality-Based Deduplication Performance Booster for Cloud Backup Services. In *Proceedings of the 25th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)* (2011), pp. 1266–1277.

[30] TEODOSIU, D., BJORNER, N., GUREVICH, Y., MANASSE, M., AND PORKKA, J. Optimizing file replication over limited bandwidth networks using remote differential compression. *Microsoft Research TR-2006-157* (2006).

[31] TUCKER, C., SHUFFELTON, D., JHALA, R., AND LERNER, S. OPIUM: optimal package install/uninstall manager. In *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007* (2007), pp. 178–188.

[32] WEI, J., JIANG, H., ZHOU, K., AND FENG, D. MAD2: A scalable high-throughput exact deduplication approach for network backup services. In *Proceedings of the 26th IEEE Conference on Mass Storage Systems and Technologies (MSST)* (May 2010), IEEE, pp. 1–14.

[33] XIA, W., JIANG, H., FENG, D., AND HUA, Y. SiLo: A Similarity-Locality based Near-Exact Deduplication Scheme with low RAM Overhead and High Throughput. In *Proceedings of the USENIX Annual Technical Conference (ATC)* (2011).

[34] ZHU, B., LI, K., AND PATTERSON, R. H. Avoiding the disk bottleneck in the data domain deduplication file system. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST)* (2008), pp. 269–282.