Sustainability **2014**, *6*, 6351-6361; doi:10.3390/su6096351

OPEN ACCESS

sustainability

ISSN 2071-1050

www.mdpi.com/journal/sustainability

Article

A Light-Weight Metering File System for Sustainable Real-Time Meter Data Management

Gangman Yi [1], Seok-Jun Choi [2] and Kwang-il Hwang [3,*]

[1] Department of Computer Science & Engineering, Gangneung-Wonju National University, Gangwon-do 220-711, Korea; E-Mail: gangman@cs.gwnu.ac.kr

[2] R&D Center Leotek Co., Ltd, Incheon 403-911, Korea; E-Mail: csj@leotek.co.kr

[3] Department of Embedded Systems Engineering, Incheon National University, Incheon 402-772, Korea

* Author to whom correspondence should be addressed; E-Mail: hkwangil@incheon.ac.kr; Tel.: +82-32-835-8762; Fax: +82-32-835-0782.

Received: 24 June 2014; in revised form: 15 August 2014 / Accepted: 5 September 2014 / Published: 15 September 2014

Abstract: A real-time smart metering system has strict requirements, since every piece of data gathered from various meters every hour is of importance, and each component consisting of metering infrastructure should be sustainable. Therefore, it is necessary to efficiently manage the meter data set in smart metering networks as well as in a server. Therefore, we propose a dedicated file system, a LIght-weight Metering File System (LIMFS), which is capable of not only efficiently storing and searching meter data but also performing distributed fault-tolerant meter data management for real-time smart meter devices. The proposed LIMFS exploits accumulated data sliding storage (ADSS) for lost data recovery and latest-first error-ignorant data management (LEDM) to reduce memory wastage, coping with dynamic report interval. Experimental results demonstrate that LIMFS has as a small enough overhead to be considered negligible, and provides flexible memory capacity according to dynamic report interval, in spite of lost data recovery functionality.

Keywords: file system; smart metering; real-time

1. Introduction

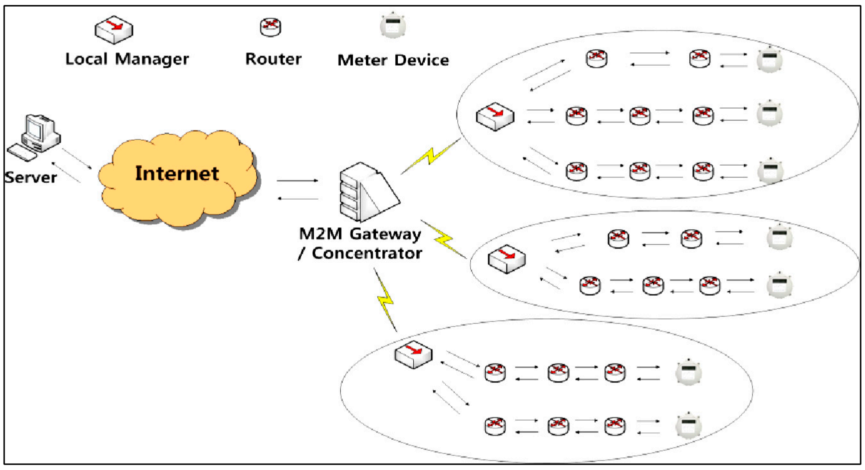
Beyond its basic functions, the automated metering system (AMR) is being evolved rapidly toward real-time smart metering system which can support dynamic pricing and provide efficient metering data management. Energy saving and demand forecasting with respect to energy consumed in each home can be more accomplished by real-time AMR systems. For the real-time smart metering system, the meter data of every house should be stored in a distributed way in metering infrastructure rather than only in a utility server, to facilitate dynamic meter data management and support various services. To satisfy the requirements, a dedicated AMR file system to be used in each device consisting of metering infrastructure is required. Even though there exist a number of file systems for embedded systems, a research on a dedicated file system for AMR systems has not been performed so far. In addition, because of the unique features of AMR systems, general file systems for various embedded systems are not suitable for AMR systems. Well-known embedded file systems [1–5] contain a large amount of redundant components to be associated with embedded operating systems, so that their size is too big to apply to real-time metering systems. Furthermore, general purpose file systems do not support data recovery functionality to deal with lost data occurring in a metering network. In addition, advances of memory devices in small-size and low-cost have enabled us to store and forward various kinds of data in an embedded system. Our AMR file system design is motivated by several in-network data management methods in sensor networks. Sinha *et al.* [6] and Dubey *et al.* [7] emphasized the importance of store and forward in sensor networks, and Lee [8] proposed an intelligent system for an autonomous groundwater management. In addition, new research [9–11] relating to various novel applications and services based on in-network data mining and analysis algorithms for wireless sensor networks are also actively being performed, and most of them are emphasizing the importance of efficient data management based on distributed local storage. Therefore, since smart metering systems can also have memory size sufficient to play a role in distributed local storage in network, in order to efficiently store and manage real-time metering data, a design of a dedicated metering file system is necessary for sustainable home energy management.

Therefore, in this paper we propose a dedicated file system, a light-weight metering file system, which is capable of not only efficiently storing and searching meter data but also performing distributed fault-tolerant meter data management for real-time smart meter devices. Basically, LIMFS is designed for each device consisting of real-time smart metering infrastructure, and enables efficient metering data search and management by exploiting Pointer Vector Table (PVT). In order to ensure fault-tolerant meter data management, a meter device, which is an end device in the smart metering networks, stores and manages metering data using ADSS. Furthermore, a local manager device, which aggregates meter data in a local area, can efficiently utilize memory space by exploiting LEDM associated with ADSS.

2. Real-Time Smart Metering Infrastructure

Based on two-way communication infrastructure, real-time smart metering enables to gather energy consumption information per house, in general every h or in real-time. Real-time smart metering infrastructure has generally hierarchical architecture composed of Server, M2M gateway/concentrator, local managers, routers, and meter devices, as shown in Figure 1. A meter device as an end device in the network reads and stores meter data periodically, and transmits meter data to a local manager through router(s). A router forwards data between meter device and local manger. A local manager plays an important role in aggregating meter data in local area, and transmitting the aggregated data to a concentrator (M2M gateway). A concentrator plays a role in a kind of M2M gateway which connects the smart meter network and the Internet using CDMA, Wi-Fi, or LAN. Therefore, a concentrator can transmit locally aggregated meter data to a remote server through the Internet.

Figure 1. Real-time smart metering infrastructure.



3. LIMFS: Light-Weight Metering File System

A real-time smart metering system should be able to accommodate large volumes of data in the network side as well as a server. That is, a dedicated file system for smart meter devices is required rather than direct flash memory access. More specifically, a smart metering file system should meet the following requirements:

(1) Small overhead
(2) Distributed storage in network
(3) Efficient data management and search
(4) High reliability
(5) Fault-tolerant

In this Section, we introduce a novel real-time smart metering file system called LIMFS, which fulfills the above requirements.

As shown in Figure 2, LIMFS basically contains four independent areas (sectors), which are common in all devices in smart metering network. In the install information area, install and boot information (e.g., device serial number, install location, install time, *etc.*) are stored. Environmental variables (e.g., logical network id, route information, duty cycle information, current time, *etc.*) are stored in the environment variable area. The pointer vector table section is an area to store pointer vector table information, which is used to efficiently manage data area, and it contains data block sequence number, PVT address, physical data address, *etc.* Data area is divided into a number of pages and each page is also composed of a number of data blocks. A block size is variable, and the number of blocks in a page is also varied according to the type of device. A PVT represents a page, so whenever a page is added or removed, the corresponding PVT is also added or removed. Basically, Both the PVT area and data area have circular queue structure, so the file system overhead is very low. Figure 3 shows a simple example of how to manage data area by PVT. Data area A, B, and C are paired with PVT A, B, and C, respectively. Each PVT stores a corresponding block sequence number, PVT area, physical block address, and TIME information, and, in particular, since LIMFS basically allows time-based data block search, the main key value in PVT is TIME.

Figure 2. Common Light-Weight Metering File System (LIMFS) architecture.

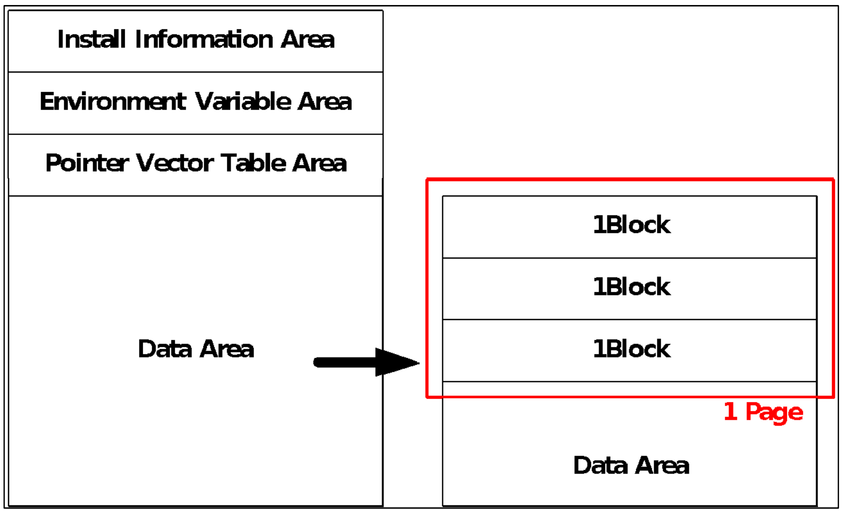
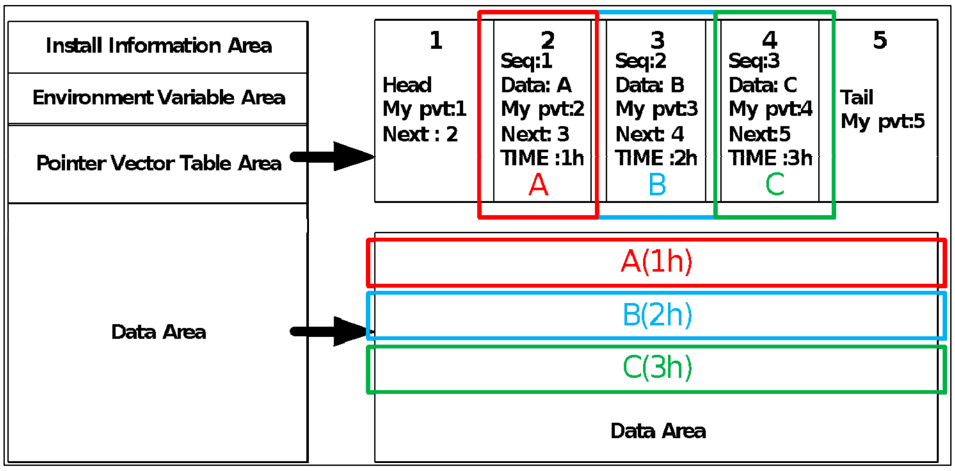


Figure 3. An example of Pointer Vector Table (PVT) and data area.

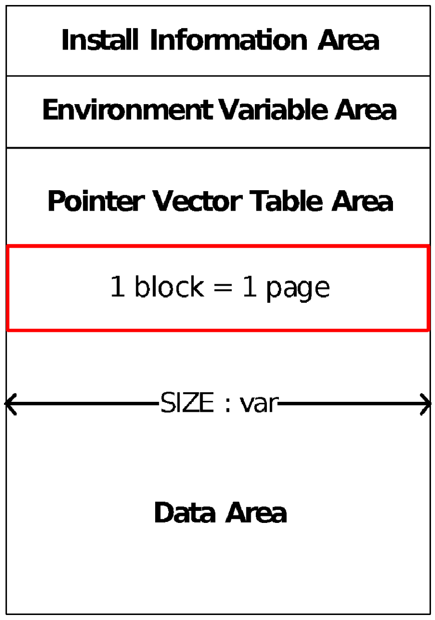


In the following subsections, device-specific architecture based on common LIMFS architecture is presented.

3.1. LIMFS for Meter Devices

As mentioned in Section 2, meter device reads and stores current meter data periodically or by request. A meter data occupies normally 4–8 bytes, but, additionally, reading time and device specific information, *etc.*, should be stored together with data in the data block. A file system for meter device therefore has four common areas and is capable of operating with small memory, since a block becomes a page, as shown in Figure 4. In addition, whenever the device reads meter value, PVT is added and the meter value and related additional information are stored in a corresponding data area.

Figure 4. LIMFS architecture for meter devices.



Accumulated Data Sliding Storage

Whenever a device reads value from a meter, the current meter value is stored in one block, and in general, a single meter value is stored in each block. It is important to note that in a real-time smart metering system, each value of meter data read at each time is of importance, since real-time meter data represents real-time energy consumption of a house. Therefore, all the meter data should be stored and transmitted reliably. However, wireless communications often suffer from several unexpected errors such as collisions, interferences, or jamming signals, so that a meter device is responsible for storing each data safely, and lost data should be able to be recovered by upper devices. Therefore, in order to facilitate lost data recovery, LIMFS utilizes accumulated meter data instead of a current meter data in a single block.

Let D_i denote meter data read at time i, and B_k denote kth block. Then, accumulated data block, B_k can be represented by as follows: $B_k = (D_i, D_{i-1}, D_{i-2}, \ldots D_{i-n})$, where n is window size. In addition, each time meter data is stored, accumulated data is shifted as follows:

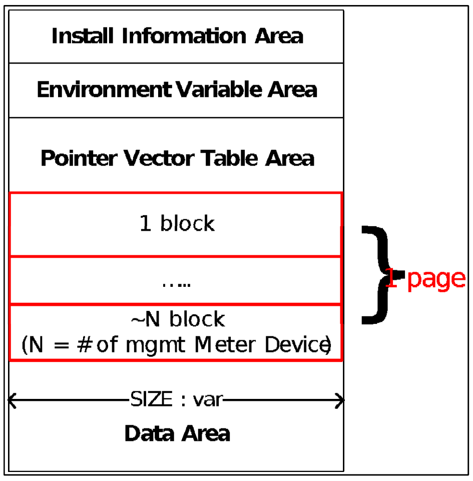
$B_k = (D_{i-3}, D_{i-4}, D_{i-5}, \ldots D_{i-n-3})$
$B_{k-1} = (D_{i-2}, D_{i-3}, D_{i-4}, \ldots D_{i-n-2})$
$B_{k-2} = (D_{i-1}, D_{i-2}, D_{i-3}, \ldots D_{i-n-1})$
$B_{k-3} = (D_i, D_{i-1}, D_{i-2}, \ldots D_{i-n})$, // the latest block

Finally, a meter device transmits an accumulated data block to local manager. So, even though some data is lost in this transaction, the lost data can be recovered from accumulated data stored in the next or later transaction.

3.2. LIMFS for Local Manager

The Local Manager is responsible for storing meter data received from meter devices and transmitting them to a concentrator. Since all data of all meter devices managed by a local manager can be stored in a local manager file system, the local manager requires larger memory size than meter devices. In addition, a page is composed of multiple blocks, and the number of blocks in a page is the same as the number of managed meter devices, as shown in Figure 5.

Figure 5. LIMFS for local manager.


Install Information Area
Environment Variable Area
Pointer Vector Table Area
1 block
......
~N block
(N = # of mgmt Meter Device)
1 page
SIZE : var
Data Area

Latest-First and Error-Ignorant Data Management

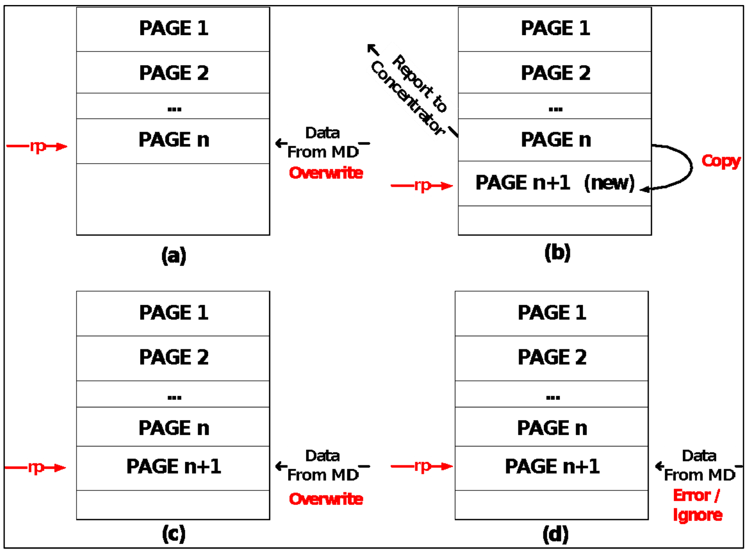
A local manager plays an important role in intermediate network storage in real-time smart metering networks. Even though the flash memory size of the local manager is greater than meter devices, it is necessary to save memory without any data loss. Therefore, we propose a novel storage management scheme in local manager. First, we need to notice additional impact on data report of a concentrator. Every local manager has to transmit data aggregated from all managed meter devices to the server through a concentrator. However, the transmission (report) interval of concentrator might depend on server or network status and capability. That is, a concentrator cannot only send data at every hourly interval but also send data at larger interval (2 h, 3 h, 4 h, or more). Since the LIMFS utilizes ADSS in meter device, meter data information for the past few hours can be obtained from the latest aggregated data received. However, as the report interval of concentrator is increased, the local manager has to store more data. The total available pages (based on time) to be stored in the file system can be obtained as follows:

$$\mathrm{T(I)} = \left(\frac{\mathrm{d}}{\mathrm{k} \times \mathrm{x} \times \mathrm{n}} - 1\right) \times \mathrm{I} + \mathrm{n} \quad (1)$$

where d is total size of data area in bytes, k is a block size in bytes, and window size and the number of meter devices are denoted by n and x, respectively.

This reveals that the increase in report interval of a concentrator might lead to significant memory wastage in local manager, since consecutive pages contain overlapped (redundant) data occurred due to ADSS. Therefore, we propose an efficient file system management to cope well with dynamic changes of report interval. To deal with variable report interval, we utilize a report pointer (rp) indicating the lastly reported page to a server via a concentrator. Each time the data report is requested, a local manager transmits the page currently indicated by a report pointer, and after finishing report transmission, a new page is created and the report pointer indicates the new page. So, whenever meter data is aggregated from meter devices, local manager first checks errors in received packet (e.g., errors include meter reading error, incorrect time data, null data, *etc.*, and each error is marked in error filed in the corresponding block of each device file system), and if there is no error in the packet, aggregated data extracted from the packet is stored onto the page indicated by a report pointer. Even though the page is not a new page (*i.e.*, there already exists previous aggregated data stored in the page), a local manager overwrites aggregated data onto the page indicated by a report pointer. Therefore, the page currently indicated by a report pointer represents lastly aggregated data and it results in significant memory saving in a local manager by offsetting the impact on ADSS by meter devices. It is important to note that if there is an error in the lastly aggregated data, the previous data without an error will be replaced by error data. To avoid this problem, a local manager is capable of identifying errors by checking error fields in the received packet. So, if the received packet has some errors, the local manager does not store the data but ignores it. Figure 6 shows an example of latest-first and error-ignorant data management. Figure 6a illustrates LEDM processing when a report pointer indicates page n, aggregated data is received from meter devices. Figure 6b shows that the local manager performs a data report to a concentrator. Figure 6c shows a process when another aggregated data is received after Figure 6b. Figure 6d describes file system operation when a received packet has an error.

Figure 6. An example of latest-first and error-ignorant data storage.

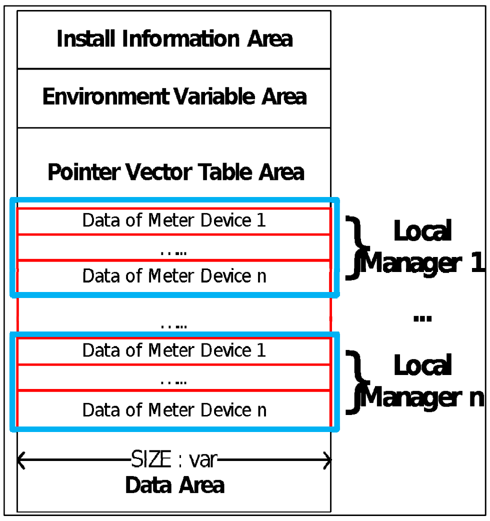


3.3. LIMFS for Concentrator

The concentrator plays a role in connecting smart metering networks to the Internet, and it also aggregates all data from managed local managers. Therefore, as shown in Figure 7, file system for the concentrator is capable of storing temporary data received from local managers, and the temporary data is transmitted using TCP/IP. The size of the data area depends on the number of managed local managers and meter devices. The stored data is requested by server periodically or on-demand, and, without regard to server's request, a concentrator requests managed local managers periodically (in general, at 1 h intervals) and stores data received from managed local managers into file system. The stored data to be transmitted to server are aggregated safely through local managers, routers, and meter devices in a smart metering network. The size of the data is near optimally minimized through

ADSS and LEDM. In addition, since headers of each device are encapsulated into the data block, the server can identify each network status as well as real-time meter data.

Figure 7. Concentrator file system.



4. Performance Evaluations

In order to evaluate the performance of proposed LIMFS, we implemented LIMFS on smart metering systems using Texas Instrument ultra-low power MCU MSP430F6137 and 128 Kbytes FRAM (FM24V10-G). Through experiments, we evaluated block read time, memory efficiency, and recoverability, respectively. Table 1 summarizes the key parameter for the performance evaluation of LIMFS. As mentioned previously, a research on a dedicated file system for AMR systems has not been performed so far, which means that our work deserves the first attempt for the AMR file system. Therefore, since there is no competitor to compare performances with LIMFS, the performances of LIMFS itself are evaluated.

Table 1. Key parameter for experiments.

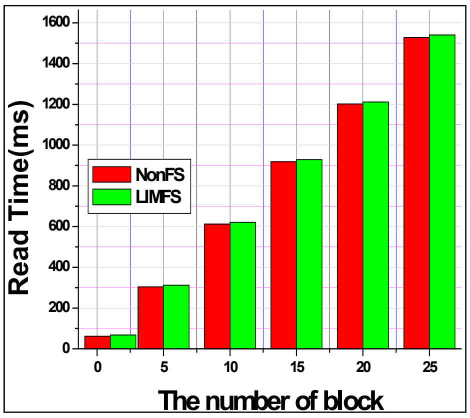
No.	Parameter	Range
1	Packet payload size	1~128 Byte
2	Meter data size	1~80 Byte
3	Meter Device Block size	1~80 Byte
4	Local Manager Block size	1~100 Byte
5	The number of meter data managed by 1 local manager	1~100
6	Window size	1~80 Byte
7	Server report interval	1~12 h

4.1. Block Read Time

First, we measured block read time of LIMFS compared with non-file system in which low level memory access is used. According to our experiment, the block writing time of LIMFS and direct writing showed a similar performance, so we focus only on the block reading time.

Figure 8 shows the comparison result of block read time of LIMFS and non FS, with respect to varying block size. While LIMFS accesses to data area using PVT, since non FS directly accesses to data area, overall block read time is slightly better. However, the result also proves that overhead of LIMFS is small enough to be negligible.

Figure 8. File system block real time.



4.2. Memory Efficiency

We also observed memory efficiency (capacity in h) of local manager. As mentioned in Section 3, while ADSS by meter devices can enhance data reliability and recoverability, it also requires larger memory space in local manager. So, we have experimented the performance and impact of ADSS and ADSS + LEDM (performed in local manager), compared to non FS. The experiment is conducted by following the same conditions: Data area size = 1500 bytes, data block size is 20 bytes, window size = 5, and managed meter device = 1. In addition, in the case of ADSS + LEDM, we also observed capacity with respect to varying report interval of concentrator from 1 to 5 h.

Table 2 shows maximum memory capacity comparison. While Non-FS can store meter data for 75 h, ADSS-only system can store data only for 19 h. However, in the case of ADSS + LEDM, as the report interval of concentrator to a server in increased, the memory capacity is extended as many as non-FS. The result shows that LIMFS copes well with dynamic variation of report interval by utilizing flexible memory management.

Table 2. Maximum memory capacity (h).

File System	**Non-FS**	**LIMFS**					
			ADSS + LEDM				
		ADSS Only	**Report Interval (h)**				
			1	**2**	**3**	**4**	**5**
Capacity (h)	75	19	19	33	47	61	75

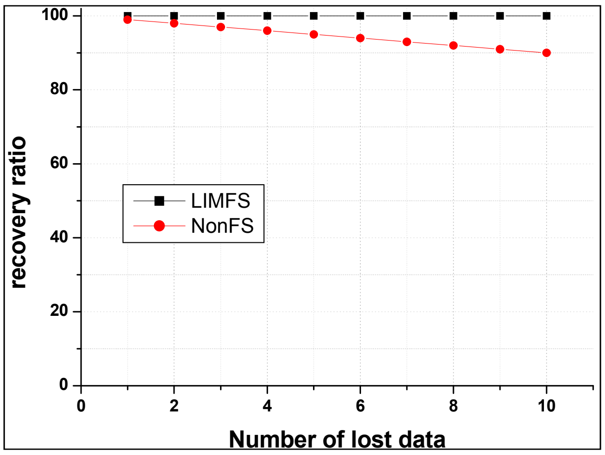
4.3. Recoverability

One of the outstanding features of LIMFS is lost data recovery. As mentioned in Section 2, in real-time smart metering system, every meter data information read every hour is of importance. So, we observed a recoverability of LIMFS and non-FS. Recovery ratio can be obtained as follows:

$$R = \frac{\text{Total number of data} - \text{loss data} + \text{recovered data}}{\text{Total number of data}} \times 100 \quad (2)$$

Figure 9 shows the experimental result of recovery ratio with respect to varying the number of lost data. The result shows that LIMFS is capable of recovering lost data by utilizing ADSS in meter device and LEDM in local manager.

Figure 9. F recovery ratio *vs.* lost data.



5. Conclusions

Real-time smart metering systems require different characteristics, and thus a dedicated file system to cope well with real-time meter data is required. Therefore, in this paper we proposed a lightweight metering file system. Outstanding features of LIMFS include small size and small overhead, distributed real-time storage, accumulated data sliding storage for lost data recovery, and latest-first error-ignorant data management to cope well with dynamic report interval. The experimental results demonstrated that LIMFS has a small enough overhead to be considered negligible, and provides flexible memory capacity according to dynamic report interval, in spite of lost data recovery functionality. Even though LIMFS does not support various functionalities supported by general embedded file systems, the major contribution of this paper is to raise the necessity of a dedicated file system for real-time smart metering systems, and present a guideline of designing smart metering file system by introducing the proposed LIMFS.

Acknowledgments

This research was supported by a grant (12-TI-C01) from Advanced Water Management Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

Author Contributions

The authors listed all contributed to the development of the idea and experiments contained within this paper. More specifically, the first version of this paper was written by the first author, structure and revisions of this paper were led by the corresponding author, and Seok-Jun Choi made a contribution to case-study of this work.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Álvarez, G.; Cepeda, H.; Laniella, I.; Benítez, N.; Amuchástegui, C. Embedded System for a Wireless Automatic Meter Reading Management. In Proceedings of the IADIS International Conference Applied Computing, Salamanca, Spain, 18–20 February 2007.
2. Jain, S.; Lee, Y.-H. Real-Time Support of Flash Memory File System for Embedded Applications. In Proceedings of the Fourth IEEE Workshop on SEUS-WCCIA'06, Gyeongju, Korea, 27–28 April 2006.
3. Wei, C.J.; Yang, J.J. Implementation of Automatic Meter Reading System Using PLC and GPRS. *J. Inf. Comput. Sci.* **2011**, *16*, 4343–4350.
4. Khalifa, T.; Naik, K.; Nayak, A. A Survey of Communication Protocols for Automatic Meter Reading Applications. In Proceedings of the IEEE Communications Surveys & Tutorials, Waterloo, ON, Canada, 5 May 2011.
5. Zaballos, A.; Vallejo, A.; Majoral, M.; Selga, J.M. Survey and Performance Comparison of AMR Over PLC Standards. *IEEE Trans. Power Deliver.* **2009**, doi: 10.1109/TPWRD.2008.2002845.
6. Sinha, A.; Lobiyal, D.K. Performance evaluation of data aggregation for cluster-based wireless sensor network. *Hum.-Centric Comput. Inf. Sci.* **2013**, *3*, doi:10.1186/2192-1962-3-13.
7. Dubey, T.; Sahu, O.P. Self-Localized Packet Forwarding in Wireless Sensor Networks. *J. Inf. Process. Syst.* **2013**, *19*, 477–488.
8. Lee, M. Design of an Intelligent System for Autonomous Groundwater Management. *J. Converg. FTRA* **2014**, *5*, 26–31.
9. Halatchev, M.; Gruenwald, L. Estimating Missing Values in Related Sensor Data Streams. In Proceedings of the Advances in Data Management, Norman, OK, USA, 2005.
10. Poolsawad, N.L.; Kambhampati, M.C.; Cleland, J.G.F. Handling Missing Values in Data Mining-A Case Study of Heart Failure Dataset. In Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery, Chongqing, China, 29–31 May 2012.
11. Mahmood, A.; Shi, K.; Khatoon, S.; Xiao, M. Data Mining Techniques for Wireless Sensor Networks: A Survey. *Int. J. Distrib. Sens. Netw.* **2013**, *2013*, doi:10.1155/2013/406316.