SAPM: Self-Adjusting Pipelining Mechanism for Efficient Bulk Data Dissemination in Smart Homes

Jun-Wei Li, Shi-Ning Li, Yu Zhang, Bin Guo, and Zhe Yang
School of Computer Science, Northwestern Polytechnical University, Xi'an, China
Email: ljw@mail.nwpu.edu.cn, {lishining, zhangyu, guob, zyang}@nwpu.edu.cn

***Abstract*—Wireless Sensor Networks (WSNs) are popular in smart homes to implement context-aware and intelligent services. Due to fixing bugs and adding new functions, the software on sensor nodes needs to be updated depending on efficient bulk data dissemination. Existing page transfer of data dissemination protocols employ pipelining technique to achieve spatial multiplexing. The conventional pipelining inefficiency, however, becomes a bottleneck in terms of dissemination delay and communication overhead. In this paper, we propose SAPM, a novel Self-Adjusting Pipelining Mechanism for efficient bulk data dissemination. By considering the impact of link correlation, detecting accumulated pages in sensor nodes, and carefully selecting the number of pages to request, SAPM shortens the dissemination delay effectively and reduces communication overhead to improve current bulk data dissemination protocols' performance. Our evaluation results on testbed experiments and extensive simulations show that SAPM reduces the completion time of state-of-the-art bulk data dissemination protocols by 27.8% on average compared to naive solution and the number of transmissions by 17.4% on average under high link correlation while having a low memory overhead.**

I. Introduction

Nowadays, the concept of smart homes has been proposed to demonstrate pervasive computing applications in our daily life. Wireless sensor network (WSN) provides a lightweight and rapid solution to build wireless smart homes [1]. These sensor nodes are equipped with various sensors to collect environmental information, such as temperature and light. For example in Fig. 1, these sensor data are forwarded to the sink node, a local gateway in the home. A solution vendor programs and maintains softwares on these sensor nodes to obtain light context in region A (bedroom and corridor) and energy-level context in region B (living room and bedroom) for smart light control and energy management. The sensor nodes in region B are deployed in many places to achieve accurate temperature sensing. As a result, a multi-hop ad hoc network is built by all the sensor nodes. The owners can configure existing contexts through web interface in the gateway [2]. If software bugs are found, or the owners need more control functions for new heating and cooling devices, the softwares on sensor nodes need to be updated immediately to ensure comfortable and high-quality services. When a software update is needed, one patch file will be generated. This patch file will be pushed to the local gateway for dissemination to target sensor nodes.

As requirements from the owners could be dynamic in daily life, the rate of generating and disseminating the patch file will be quite high. However, dissemination of the patch file in smart homes has several challenges [3]. First, the patch file is bulk data compared to light and temperature data, as a patch file could be thousands times of the sensor data. It will be divided into pages before dissemination, as shown in Fig. 2. Second, the 100% completeness must be ensured for the patch file on all target sensor nodes compared to loss-tolerant multimedia data, because even one bit error in the patch file will make sensor nodes fail to work. Such failure is disaster for context-aware applications. Third, link correlation [4], [5] between wireless links is dynamic as the owners walk in rooms and Wi-Fi devices interfere with sensor nodes, its impact on disseminating the patch file has to be examined carefully in smart homes.

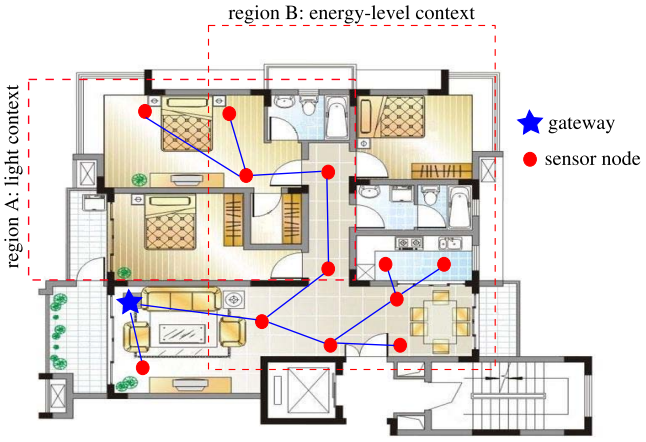

Fig. 1. An example of wireless smart homes. Light and energy-level contexts are configured by sensor nodes in regions A and B. The sensor nodes comprise a multi-hop ad hoc network, which is managed by the gateway.

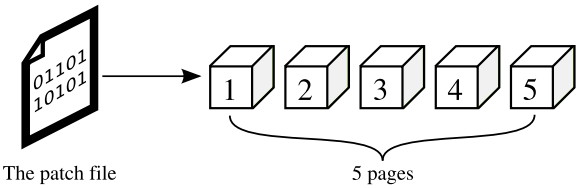

Fig. 2. A patch file is divided into five pages before dissemination.

Deluge [6] is the de facto standard for multi-hop bulk data dissemination. It divides the patch file into equal-sized pages, every page contains same number of packets. As Deluge

978-1-4673-7211-4/15 $31.00 © 2015 IEEE
DOI 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.29

allows pipelining of transmitting these pages on sensor nodes, the patch file could be disseminated in parallel. However, further performance improvement of Deluge has been hampered by its naive solution that fixed one page is requested all the time. This naive solution in conventional pipelining fails to exploit node intelligence of self-adjusting number of pages other than fixed one page to request in case of inefficiency (e.g., collision of packets). As a result, the lifetime of these pages in the pipelines is prolonged in conventional pipelining. The dissemination delay and communication overhead are still high. Recent works, such as Rateless Deluge [7], ReXOR [8], MT-Deluge [9], Splash [10], and ECD [11], which all employ conventional pipelining technique, cannot avoid such situation. In addition, the impact of link correlation on pipelining technique, however, is not explored.

Aiming to address the limitations of existing protocols based on conventional pipelining technique, we propose a novel Self-Adjusting Pipelining Mechanism (SAPM) for efficient bulk data dissemination in smart homes. First, we exploit link correlation to reduce communication overhead by mitigating retransmissions of data packets. Second, we optimize the page transfer mechanism by detecting accumulated pages in sensor nodes and carefully selecting number of pages to request in order to reduce request collisions.

We have implement SAPM in Deluge and Rateless Deluge. We conduct extensive simulations in TOSSIM [12], and also evaluate SAPM in our testbed *Tea* that comprises TelosB [13] nodes to evaluate the performance. The results show that (1) by exploiting link correlation in the page transfer period, SAPM reduces the number of transmissions by 17.4% on average across state-of-the-art dissemination protocols; (2) by applying the optimized conventional pipelining technique, SAPM shortens the dissemination completion time of state-of-the-art bulk data dissemination protocols by 27.8% on average compared to naive solution. It is worth noting that SAPM also has a low memory overhead.

In summary, we make the following contributions:

- we identify the inefficient dissemination issue of conventional pipelining. Considering the impact of link correlation, we design a selection algorithm of number of pages to request for more efficient dissemination.
- we optimize the conventional pipelining mechanism used in existing protocols, aiming to reduce the page accumulation overhead and reduce request collisions.
- we design and implement SAPM in Deluge T2 (the latest version of Deluge) and Rateless Deluge, and evaluate its performance in both testbed experiments and extensive simulations.

The remainder of this paper is organized as follows. Section II reviews related work on bulk data dissemination in WSNs. We describe the problem statement in Section III. Section IV presents the design of SAPM. Section V shows the performance evaluation of SAPM using a real world sensor network as well as extensive simulations. We conclude the paper in Section VI.

II. Related Work

Wireless Sensor Networks have been deployed in smart homes for various purposes, such as smart energy [14], smart building [15], [16], and anomaly detection [17]. As software update for sensor nodes in smart homes may be frequent due to dynamic requirements from the owners, efficient bulk data dissemination is needed to disseminate the patch file generated.

To achieve fast dissemination, Hui et al. propose Deluge [6] to implement parallel transfers of the patch file by allowing spatial multiplexing. MNP [18] and ECD [11] reduce collision of data packets by employing a sender selection algorithm for nodes in same neighborhood, compared to Deluge. On the other hand, network coding is incorporated in data transmission to mitigate packet retransmissions, such as Rateless Deluge [7], SYNAPSE++ [19], ReXOR [8], and MT-Deluge [9]. As contention resolution has been a significant overhead in aforementioned protocols, Splash [10] exploits constructive interference [20] to eliminate contention resolution for efficient data dissemination, but it still uses the same contention-based CSMA/CA in its local recovery phase for lost data packets.

Sha et al. present their empirical study of network performance (QoS) for WSNs in the homes [21], when multiple wireless networks, such as Wi-Fi and Bluetooth, exist. However, dynamic link correlation in the homes causes dynamic network performance. The κ factor [22] studies the impact of link correlation on completion time of bulk data dissemination in Deluge and Rateless Deluge. SYREN [23] exploits link correlation to implement collective ACKs to reduce redundant transmissions of packets. Corlayer [24] presents a transparent link correlation layer to improve existing protocols. CD [25] builds link correlation aware bulk data dissemination by using a structure of *correlated tree*.

Different from the studies to achieve fast data dissemination using conventional pipelining [6]–[11], [18], [19], SAPM make a decision whether dissemination inefficiency happens in conventional pipelining by detecting accumulated pages in sensor nodes. When accumulation is detected, SAPM selects adjustable number of pages to request and considers the impact of link correlation to achieve performance improvement. As more pages other than fixed one page could be transmitted for request packets when accumulation is detected, the communication overhead and dissemination delay could be reduced.

III. Problem Statement

Although the performance of dissemination could be improved by conventional pipelining, the authors of Deluge [6] find that propagation of pages is slow along the diagonal in grid networks, due to heavy contention of packets generated by sensor nodes. Contention also exists in the time consuming (at least 60% of dissemination delay) local recovery phase of Splash [10]. As three types of packets: advertisement (ADV), request (REQ), and data (DAT) are used [26], we identify which type of packets causes the contention mostly.

As ADV packets are broadcast periodically by all sensor nodes, the sensor nodes will not broadcast them at the same time. The probability of contention caused by ADV packets

is low. The contention of DAT packets is due to the well known *hidden terminal problem*, it has been mitigated by Deluge, MNP [18], and ECD [11]. However, the contention of REQ packets is a major issue. First, as REQ packets can be generated by many sensor nodes at the same time, the contention will increase backoff time for another rounds. Second, the time to deal with REQ packets is the main part of dissemination delay. Based on our empirical results on conventional pipelining, we find that the contention of REQ packets makes sensor nodes near the gateway accumulate many pages. For example in Fig. 3, the sender node S accumulates two pages compared to receiver nodes A and B. Receivers A and B send a REQ packet each for fixed one page. As the two REQ packets cause contention in sender S, they have to contend at another rounds for successful request. The naive solution in conventional pipelining, however, requires that fixed one page is requested, the receivers cannot react to such accumulation. This accumulation increases communication overhead and prolongs the dissemination delay as a result.

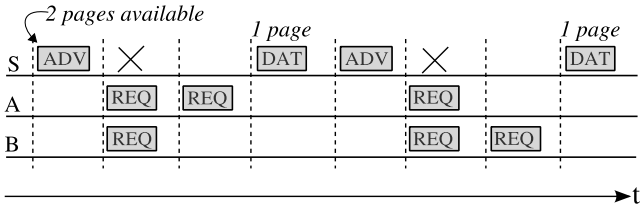

Fig. 3. The problem of conventional pipelining. Two pages are accumulated in the sender node S. The receiver nodes A and B request fixed one page in REQ packets for new pages available in sender node S.

IV. Design of SAPM

SAPM is composed of three phases: detection of accumulation (SA-I), on demand self-adjusting (SA-II), and adjusted page transmission (SA-III), as shown in Fig. 4. When an ADV packet is received by any sensor node, phase SA-I is activated to detect accumulation. Phase SA-II starts when accumulation is detected, the sensor nodes will select adjustable number of pages to request on demand and send a REQ packet. When any sensor node receives a REQ packet for adjustable number of pages, it starts adjusted page transmission in phase SA-III.

Phase SA-I: Whenever a sensor node receives an ADV packet broadcast by a sender, it reads information in the packet and compares the field *available pages* in the ADV packet ($PgCnt_{adv}$) with its own available pages ($PgCnt_{my}$). First, if $PgCnt_{adv}$ is greater than $PgCnt_{my}$ by 1, accumulation may be detected, else the receiver deals with the ADV packet as normal. Second, if the sender node is the gateway, we think that accumulation is not detected at last, else accumulation is detected. This is because when the gateway broadcasts an ADV packet for the first time to start data dissemination, any receiver of this ADV packet will conclude that $PgCnt_{adv}$ is greater than $PgCnt_{my}$ by 1. To accurately detect accumulation, the gateway should be ignored.

Phase SA-II: When accumulation is detected, the receiver node can adjust the number of pages *numPgs* (*numPgs* $\in \mathbb{R}^+$)

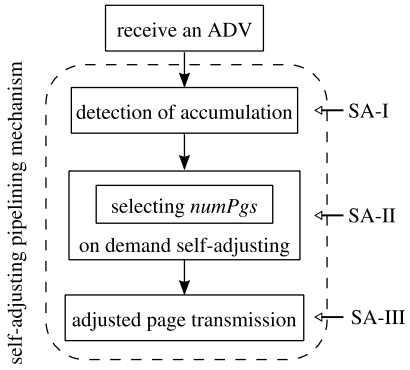

Fig. 4. Flowchart of SAPM. It contains three main phases: detection of accumulation (SA-I), on demand self-adjusting (SA-II), and adjusted page transmission (SA-III).

to request on demand. For example in Fig. 5, the receivers A and B have detected the accumulation in the sender node S, they could send REQ packets for two pages instead of fixed one page, the accumulated pages can be forwarded as soon as possible. As the number of pages in REQ packets can be adjusted (i.e., increased), the number of REQ packets is reduced, the accumulation is mitigated as well. Besides, a new DAT-SYNC packet is introduced that the sender S broadcasts it before actual transmission of data packets to avoid the *hidden terminal problem*. The DAT-SYNC packet contains information that which pages are to be transmitted. On the other hand, the DAT-SYNC packet will contribute to REQ suppressions in neighbor nodes, as more than one page will be received without sending REQ packets.

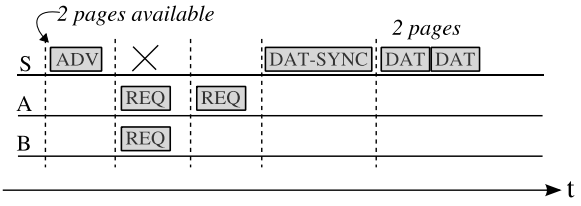

Fig. 5. Adjustable number of pages to request on demand. The receiver nodes A and B request two pages in REQ packets instead of fixed one page for new pages available in sender node S as accumulation is detected.

The benefit of requesting adjustable number of pages on demand can be formalized. Let $P = PgCnt_{adv} - PgCnt_{adv}$. We define T_1 and T_2 as the time of forwarding the accumulated P pages in conventional pipelining and SAPM, respectively. Based on analytical models in [27], [28], we have

$$T_1 = P[t_{adv} + t_{req}(n_{cont} + 1) + t_{pg}], \quad (1)$$

where t_{adv}, t_{req}, t_{pg} represents the time used for transmission of ADV, REQ, and one-page DAT packets. The average number of REQ contentions is n_{cont}. For T_2, we have

$$T_2 = \frac{P}{numPgs}[t_{adv}+t_{req}(n_{cont}+1)+numPgs\cdot t_{pg}+t_{ext}], \quad (2)$$

where t_{ext} represents the time used for extra $(numPgs - 1)$ data packets. The benefit of requesting adjustable number of pages exists if $T_2 < T_1$. We have

$$T_1 - T_2 = (1 - \frac{1}{numPgs})P[t_{adv} + t_{req}(n_{cont} + 1) - t_{ext}]. \quad (3)$$

The benefit depends on selecting *numPgs* to maximize $T_1 - T_2$, as increasing *numPgs* enlarges t_{ext} and reduces n_{cont}, but raises $(1 - \frac{1}{numPgs})$.

The procedure of phase SA-II is described in Fig. 6. In receiver side, an important thing to do is how to select optimized *numPgs* to request based on the field *available pages* $PgCnt_{adv}$ in ADV packet from phase SA-I. A simple algorithm to select *numPgs* to request is given in Procedure 1. After *numPgs* is selected, the receiver will add an additional field in the REQ packet to save *numPgs* and send it to the sender. In sender side, whenever the sender node receives a REQ packet, it checks whether *numPgs* in the REQ packet is greater than 1. If *numPgs* is greater than 1, the sender knows that self-adjusting is activated in the sensor node that sends this REQ packet. Then the sender node broadcasts a DAT-SYNC packet immediately to inform its neighbors of upcoming adjusted page transmission. After that, it will start adjusted page transmission and move to phase SA-III.

in receiver node:

```
if (accumulation == true)
    req = new (REQ msg);
    numPgs = select_by (&ADV);
    req.add (numPgs);
    send (req);
endif
```

in sender node:

```
numPgs = read (REQ msg);
if (numPgs > 1)
    msg = new (DAT-SYNC msg);
    broadcast (msg);
    adjusted_page_transmission(numPgs);
endif
```

Fig. 6. The work flow of phase SA-II in pseudocode: on demand self-adjusting on both the receiver and sender nodes.

Phase SA-III: When the receiver node sets *numPgs* to be greater than 1, it has to allocate extra memory space to store all the data packets of $\lceil numPgs - 1 \rceil$ pages. As the receiver node needs to collect more than one page, the sender will broadcast all the data packets of these pages after DAT-SYNC packet is broadcast. Suppose the receiver sets $numPgs = 2$, it will allocate extra memory space for all the data packets of one page. The impact of link correlation on transmission of these data packets is shown in Fig. 7. Assume one page contains 10 data packets, both links from the sender node S to receiver nodes A and B have a packet reception rate of 60%. For low link correlation of 30%, 80% data packets need retransmissions, whereas only 50% data packets need retransmissions for high link correlation of 80%. Less retransmissions imply less REQ packets sent for missing data packets.

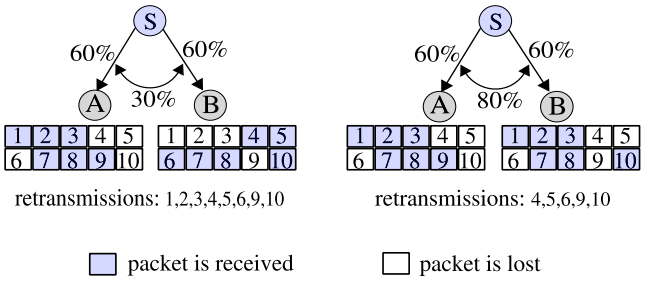

Fig. 7. The impact of link correlation to reduce retransmissions. Number of REQ packets could be reduced further. The sender node S broadcasts all the 10 data packets in one page to receiver nodes A and B.

Selecting *numPgs* in phase SA-II: In phase SA-II of SAPM, it is not known how to select *numPgs* to request for performance optimizations. The challenge relies on selection of an appropriate *numPgs*, as it may not be an integer. As the number of data packets in fixed one page is known at compile time, inappropriate selection of *numPgs*, which is not a multiple of the fixed one page, may cause additional overhead in memory space to store the scattered data packets in middle nodes and computation to divide the patch file based on fixed one page. Thus, we may try selecting *numPgs* from the point of data dissemination. The details of selecting *numPgs* to request are illustrated in Procedure 1.

Procedure 1: Selection of *numPgs* to request.

for *Incoming ADV packet* **do**
  read $PgCnt_{adv}$ from ADV;
  calculate pgDiff = $PgCnt_{adv} - PgCnt_{my}$;
  if pgDiff ≤ 0 **then**
    break; /* do as normal */
  end
  if $0 <$ pgDiff ≤ 1 **then**
    send REQ for one page;
  else if $\lceil \frac{\text{pgDiff}}{2} \rceil \leq 2$ **then**
    $numPgs = 2$; /* medium contention */
  else
    $numPgs = 3$; /* worse contention */
  end
  send REQ for *numPgs* pages;
end

As the sender node accumulates $(PgCnt_{adv} - PgCnt_{my})$ pages (known in phase SA-I), we consider selection of *numPgs* to be a new dissemination of one patch file having $(PgCnt_{adv} - PgCnt_{my} - 1)$ pages beyond. As the empirical

results in [27], [28] imply, number of data packets in one page should be set small to reduce the completion time of dissemination. We choose the same number of data packets again for one page in the new data dissemination. Therefore, we have a sub-optimal *numPgs* of two pages (*numPgs* $= 2$) at last. Other than reduction in the completion time, *numPgs* $= 2$ introduces no additional computation to divide the patch file. For a very large $(PgCnt_{adv} - PgCnt_{my})$, we could increase the *numPgs* temporarily to forward data packets quickly, while increasing memory overhead. The *numPgs* is set to be 3 for a worse contention in Procedure 1.

V. Performance Evaluation

The evaluation of 100% completeness for the patch file received in sensor nodes is direct, because the completeness is assured by handshaking of ADV, REQ, DAT-SYNC, and DAT packets in different phases of SAPM. As SAPM reduces the communication overhead directly, we first evaluate SAPM in a real world testbed across Deluge and Rateless Deluge to validate our analysis. The complex environment in smart homes is due to dynamic PRR (Packet Reception Rate) and link correlation. We conduct extensive simulations using Deluge, Rateless Deluge, and Splash, to examine the performance of SAPM in the dissemination delay when different environments are configured. As Deluge is the de facto standard, Rateless Deluge and Splash stand for coding-based and constructive-interference-based dissemination protocols, the results have wide applicability.

A. Testbed experiments

Before extensive examinations on SAPM, we evaluate it in a real-world testbed *Tea* (Fig. 8) under different link correlation values. Two performance metrics are examined: communication overhead (number of request packets and number of packets transmitted) and memory overhead (the ROM and RAM sizes). We firstly explore the benefit of SAPM (self-adjusting pipelining) in reducing the communication overhead compared to conventional pipelining. Then we observe the memory overhead of SAPM. The testbed *Tea*, consisting of 18 TelosB [13] nodes, is mounted on the ceiling of the main hallway at our research lab (about 1700 square feet area). As the staff walk back and forth in the hallway, link correlation values are dynamic. The environment is similar to that in the smart homes. To compare with conventional pipelining across Deluge and Rateless Deluge, we implement SAPM with default selection of *numPgs* (*numPgs* $= 2$). Link correlation in *Tea* is calculated as in [4].

Communication overhead: Fig. 9 demonstrates the comparison results in number of request packets transmitted when link correlation changes. Compared to conventional pipelining, self-adjusting pipelining generates less number of request packets, when link correlation increases from 0.4 to 1.0, across Deluge and Rateless Deluge. For Deluge, the reduction in number of request packets is 24.8%, 33.0%, 34.1%, and 37.2%, respectively. For Rateless Deluge, the reduction is 19.1%, 19.3%, 20.5%, and 20.7%, respectively. When link correlation increases, the reduction in number of request packets increases too, as high link correlation could reduce number of request packets by lessening retransmissions, as described in Fig. 7.

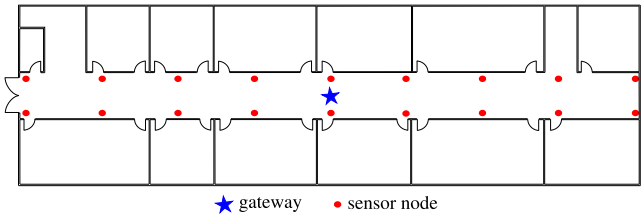

Fig. 8. *Tea*, a linear testbed consisting of 18 TelosB nodes.

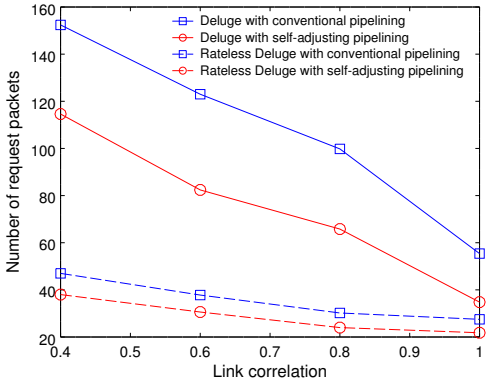

Fig. 9. Testbed results on number of request packets transmitted.

As number of request packets generated is reduced by self-adjusting pipelining, we want to know whether number of packets (i.e., all types of packets) is reduced too. The results on number of packets are shown in Fig. 10. Similar to results on number of request packets, self-adjusting pipelining also reduces number of packets across Deluge and Rateless Deluge. For Deluge, self-adjusting pipelining has a reduction of 3.0%, 12.5%, 17.6%, and 20.7% respectively in number of packets when link correlation increases from 0.4 to 1.0. For Rateless Deluge, self-adjusting pipelining has a reduction of 4.0%, 8.3%, 13.8%, and 14.1%, respectively. Self-adjusting pipelining reduces the number of packets transmitted by 17.4% on average compared to conventional pipelining, when link correlation is high.

Further, to validate our analysis on the impact of selecting *numPgs* in phase SA-II of SAPM, we observe number of request packets transmitted when different values are selected across Deluge and Rateless Deluge. The results are shown in Fig. 11. As a larger *numPgs* will generate less number of request packets, number of request packets decreases when *numPgs* increases from 1.0 to 2.0. However, when *numPgs* increases from 2.0 to 2.4, number of request packets increases. Such exception is due to lots of retransmissions caused by continuous transmission of data packets for more than two pages (*numPgs* $= 2.4$). The retransmissions increase request

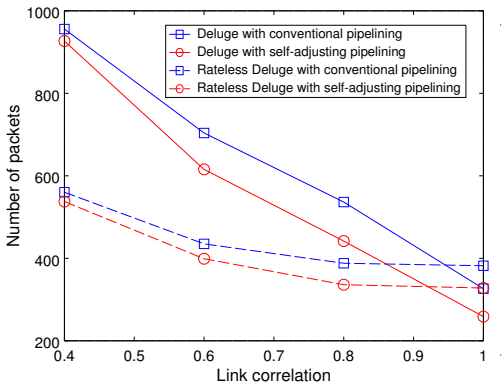

Fig. 10. Testbed results on number of packets transmitted.

packets adversely. On the other hand, a selection of 2.4 for *numPgs* introduces additional overhead to deal with scattered data packets. The default selection of 2.0 for *numPgs* achieves a good performance.

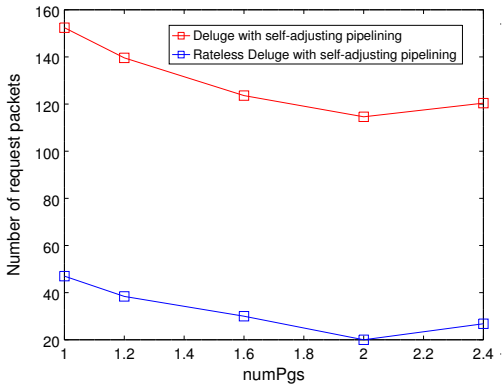

Fig. 11. Testbed results on number of request packets transmitted when different values of *numPgs* are selected in phase SA-II of SAPM.

Memory overhead: To implement self-adjusting pipelining in sensor nodes, extra memory space is spent to store data packets for extra $(numPgs - 1)$ pages. To store these data packets, at least $(20 \cdot \lceil (numPgs - 1) \rceil \cdot 22)$ bytes are allocated in RAM after compiled in TinyOS [29]. Comparison results of both ROM and RAM usage are presented in Table I. The actual ROM and RAM sizes are full protocol running on non-gateway sensor nodes. Self-adjusting pipelining has an extra memory overhead of 490 bytes (about 0.48 KB) and an extra ROM overhead of 1188 bytes (about 1.2 KB), compared to conventional pipelining.

TABLE I
THE CODE SIZE (IN BYTES) ON TELOSB.

	ROM	RAM
Conventional pipelining	35734	3746
Self-adjusting pipelining	36922	4236

B. Simulations

In this section, we make extensive examinations on performance of SAPM (self-adjusting pipelining) through simulations compared to conventional pipelining across Deluge, Rateless Deluge, and Splash. The completion time and scalability are considered. A wide range of link correlation and PRR values are configured to reflect the environment in smart homes. We modify Deluge to model local recovery phase of Splash (source code of Splash is not publicly accessible), as the same CSMA/CA access mode is used. A discrete event-driven simulator, TOSSIM [12], is used in these simulations. For comparisons, we select $numPgs = 3$ as default value for self-adjusting pipelining in local recovery phase of Splash. We present extensive comparison results on the completion time and explore the scalability finally.

The completion time: Self-adjusting pipelining is extensively examined in the completion time when different link correlation and PRR values are set. Different selections of *numPgs* in self-adjusting pipelining are also evaluated for performance optimization. We simulate a grid network having 16 sensor nodes with the gateway at the corner.

Fig. 12 shows the comparison results when PRR is low. Self-adjusting pipelining reduces the completion time by 20.6%, 23.2%, 30.1%, and 28.1% respectively in Deluge, and by 28.5%, 26.7%, 23.1%, and 21.0% respectively in Rateless Deluge, when link correlation increases from 0.4 to 1.0, compared to conventional pipelining. Moreover, it has a great reduction of 59.9%, 62.3%, 63.4%, and 58.8% respectively compared to conventional pipelining in local recovery phase of Splash, meaning at least a reduction of 35.9%, 37.4%, 38%, and 35.3% respectively in the completion time, as time of local recovery phase in Splash is more than 60% of the completion time. Self-adjusting pipelining achieves an average reduction of 29% in the completion time when PRR is low.

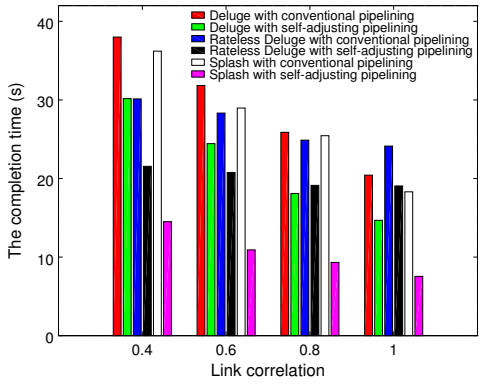

Fig. 12. The simulation results on the completion time when PRR is low (PRR = 0.4). The range ([*min, max*]) of link correlation values is [0.4, 1.0].

Fig. 13 presents the results when PRR is high. Similar to the results when PRR is low, self-adjusting pipelining reduces the completion time by 26.0%, 28.9%, and 34.9% respectively in Deluge, and by 19.4%, 19.2%, and 16.5% respectively in

Rateless Deluge, compared to conventional pipelining, when link correlation increases from 0.8 to 0.94. Also, it has a great reduction of 61.0%, 57.5%, and 57.1% respectively compared to conventional pipelining in local recovery phase of Splash, meaning at least a reduction of 36.6%, 34.5%, and 34.3% in the completion time. Self-adjusting pipelining achieves an average reduction of 27.8% in the completion time when PRR is high.

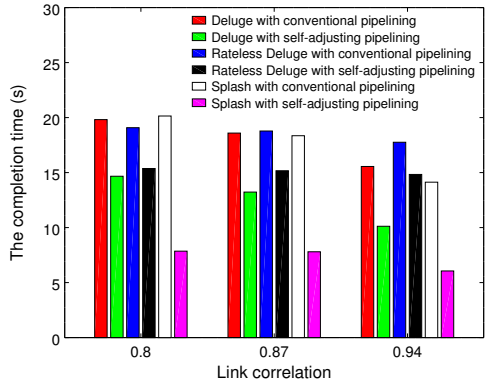

Fig. 13. The simulation results on the completion time when PRR is high (PRR = 0.8). The range ([*min, max*]) of link correlation values is [0.8, 1.0].

To explore full knowledge of *numPgs* in self-adjusting pipelining, we select reasonable values for it and evaluate its impact on the completion time. The results are presented in Tables II and III.

When changing *numPgs* for Deluge and Rateless Deluge, we choose five separate values: 1.0 (conventional pipelining), 1.2, 1.6, 2.0 (default value of self-adjusting pipelining), and 2.4. The observations are shown in Table II. The completion time drops when *numPgs* increases from 1.2 to 2.0, as self-adjusting pipelining (*numPgs* > 1.0) has a reduction of 4.7%, 15.6%, and 20.6% respectively in Deluge and a reduction of 13.0%, 25.5%, and 28.5% respectively in Rateless Deluge on the completion time, compared to conventional pipelining (*numPgs* $= 1.0$). Such reduction is due to the decreased request packets, as shown in Fig. 11. However, when *numPgs* increases from 2.0 to 2.4, the completion time increases because number of request packets increases. Self-adjusting pipelining achieves the best performance for default selection of *numPgs*.

TABLE II
THE RESULTS OF SELECTING *numPgs* IN SELF-ADJUSTING PIPELINING USING DELUGE AND RATELESS DELUGE.

The *numPgs*	The completion time (s)	
	Deluge	Rateless Deluge
1.0	38.000	30.128
1.2	36.209	26.209
1.6	32.075	22.440
2.0	30.168	21.551
2.4	34.537	23.719

When changing *numPgs* for local recovery phase of Splash, we choose three values: 1.0 (conventional pipelining), 3.0 (default value of self-adjusting pipelining) and 5.0. The observations are shown in Table III. The completion time drops when *numPgs* increases. A selection of larger *numPgs* achieves less time, as self-adjusting pipelining has a reduction of 59.9% and 78.0% respectively in the time when *numPgs* equals to 3.0 (default value) and 5.0, compared to conventional pipelining (*numPgs* $= 1.0$). By selecting a large *numPgs* (5.0), self-adjusting pipelining achieves at least a reduction of 46.8% in the completion time, a much better performance compared to default value we set.

TABLE III
THE RESULTS OF SELECTING *numPgs* IN SELF-ADJUSTING PIPELINING USING SPLASH.

The *numPgs*	The completion time (s)
1.0	36.209
3.0	14.508
5.0	7.9562

Scalability: We examine the scalability of self-adjusting pipelining based on whether its reduction on the completion time drops sharply when network size increases. Simulations are conducted in grid networks with variable network size of 16 (4×4), 36 (6×6), and 64 (8×8) sensor nodes. The results are demonstrated in Fig. 14. Compared to conventional pipelining, self-adjusting pipelining has a reduction of 20.6%, 20.1%, and 19.0% respectively in Deluge, a reduction of 28.5%, 26.6%, and 25.9% respectively in Rateless Deluge, and a reduction of 59.9%, 58.5%, and 58.8% respectively in local recovery phase of Splash on the completion time, when network size increases from 4×4 to 8×8. As reduction in the completion time remains almost the same when the network size increases, self-adjusting pipelining scales well when the network size increases.

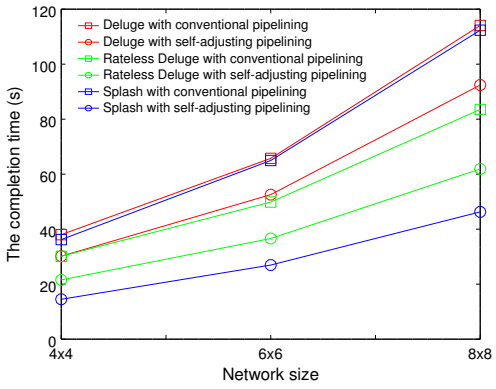

Fig. 14. The simulation results on scalability of self-adjusting pipelining when network size increases.

VI. CONCLUSION

As requirements from the owners are dynamic in smart homes, the rate of software updates is frequent for sensor

nodes deployed in smart homes. The software updates on sensor nodes rely on efficient bulk data dissemination. In this paper, we present SAPM, a novel Self-Adjusting Pipelining Mechanism on sensor nodes to efficiently disseminate the patch file generated for software updates in WSNs. By jointly considering the impact of link correlation, detecting accumulated pages in sensor nodes, and adjusting the number of pages to request on demand, SAPM could reduce communication overhead and shorten the dissemination delay to achieve fast bulk data dissemination. We conduct testbed experiments and extensive simulations to evaluate SAPM when different link correlation and PRR values are configured in order to reflect the environment in smart homes. SAPM improves performance of state-of-the-art dissemination protocols in terms of the completion time of bulk data dissemination and number of packets transmitted. Furthermore, SAPM has a low memory overhead and scales well in large-scale networks.

The benefit of SAPM will become weak in networks when accumulation is low, as design of SAPM is based on detecting accumulation during transmission of fixed data unit in bulk data dissemination. To further improve the performance of dissemination when accumulation is low, contention should be reduced. Nevertheless, SAPM could be applied in many other research works, such as ReXOR, and ECD. We leave the extensive examinations as future work.

ACKNOWLEDGMENT

This work is supported by National Science and Technology Major Project of China under grant No. $2012ZX03005007$ and the National Natural Science Foundation of China under grant No. 61402372.

REFERENCES

[1] C. Li, Z. Li, M. Li, F. Meggers, A. Schlueter, and H. B. Lim, "Energy efficient hvac system with distributed sensing and control," in *Proceedings of IEEE ICDCS*, June 2014, pp. 429–438.

[2] A. Taherkordi, F. Loiret, R. Rouvoy, and F. Eliassen, "Optimizing sensor network reprogramming via in situ reconfigurable components," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 2, pp. 14:1–14:33, Apr. 2013.

[3] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming wireless sensor networks: challenges and approaches," *IEEE Network*, vol. 20, no. 3, pp. 48–55, May 2006.

[4] T. Zhu, Z. Zhong, T. He, and Z.-L. Zhang, "Achieving efficient flooding by utilizing link correlation in wireless sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 1, pp. 121–134, Feb. 2013.

[5] J. Jun, L. Cheng, L. He, Y. Gu, and T. Zhu, "Exploiting sender-based link correlation in wireless sensor networks," in *Proceedings of IEEE ICNP*, Oct 2014, pp. 445–455.

[6] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proceedings of ACM SenSys*, 2004, pp. 81–94.

[7] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes," in *Proceedings of IEEE IPSN*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 457–466.

[8] W. Dong, C. Chen, X. Liu, J. Bu, and Y. Gao, "A lightweight and density-aware reprogramming protocol for wireless sensor networks," *IEEE Transactions on Mobile Computing (TMC)*, vol. 10, no. 10, pp. 1403–1415, Oct 2011.

[9] Y. Gao, J. Bu, W. Dong, C. Chen, L. Rao, and X. Liu, "Exploiting concurrency for efficient dissemination in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 24, no. 4, pp. 691–700, April 2013.

[10] M. Doddavenkatappa, M. C. Chan, and B. Leong, "Splash: Fast data dissemination with constructive interference in wireless sensor networks," in *Proceedings of USENIX NSDI*, 2013, pp. 269–282.

[11] W. Dong, Y. Liu, Z. Zhao, X. Liu, C. Chen, and J. Bu, "Link quality aware code dissemination in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 25, no. 7, pp. 1776–1786, July 2014.

[12] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of ACM SenSys*. New York, NY, USA: ACM, 2003, pp. 126–137.

[13] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proceedings of IEEE IPSN*, April 2005, pp. 364–369.

[14] M. Li and H.-J. Lin, "Design and implementation of smart home control systems based on wireless sensor networks and power line communications," *IEEE Transactions on Industrial Electronics (preprint)*, 2014.

[15] P.-V. Mekikis, G. Athanasiou, and C. Fischione, "A wireless sensor network testbed for event detection in smart homes," in *Proceedings of IEEE DCOSS*, May 2013, pp. 321–322.

[16] A. H. Kazmi, M. J. O'grady, D. T. Delaney, A. G. Ruzzelli, and G. M. P. O'hare, "A review of wireless-sensor-network-enabled building energy management systems," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 4, pp. 66:1–66:43, Jun. 2014.

[17] M. Usman, V. Muthukkumarasamy, X.-W. Wu, and S. Khanum, "Wireless smart home sensor networks: Mobile agent based anomaly detection," in *Proceedings of 9th International Conference on Ubiquitous Intelligence Computing (UIC)*, Sept 2012, pp. 322–329.

[18] S. Kulkarni and L. Wang, "Energy-efficient multihop reprogramming for sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 2, pp. 16:1–16:40, Apr. 2009.

[19] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi, "Synapse++: Code dissemination in wireless sensor networks using fountain codes," *IEEE Transactions on Mobile Computing (TMC)*, vol. 9, no. 12, pp. 1749–1765, Dec 2010.

[20] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proceedings of IEEE IPSN*, April 2011, pp. 73–84.

[21] M. Sha, G. Hackmann, and C. Lu, "Multi-channel reliability and spectrum usage in real homes: Empirical studies for home-area sensor networks," in *Proceedings of IEEE 19th International Workshop on Quality of Service (IWQoS)*, June 2011, pp. 1–9.

[22] K. Srinivasan, M. Jain, J. I. Choi, T. Azim, E. S. Kim, P. Levis, and B. Krishnamachari, "The κ factor: Inferring protocol performance using inter-link reception correlation," in *Proceedings of ACM MobiCom*. New York, NY, USA: ACM, 2010, pp. 317–328.

[23] S. Alam, S. Sultana, Y. Hu, and S. Fahmy, "Syren: Synergistic link correlation-aware and network coding-based dissemination in wireless sensor networks," in *Proceedings of IEEE 21st International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, Aug 2013, pp. 485–494.

[24] S. Wang, S. M. Kim, Y. Liu, G. Tan, and T. He, "Corlayer: A transparent link correlation layer for energy efficient broadcast," in *Proceedings of ACM MobiCom*. New York, NY, USA: ACM, 2013, pp. 51–62.

[25] Z. Zhao, W. Dong, J. Bu, Y. Gu, and C. Chen, "Link correlation aware data dissemination in wireless sensor networks," *IEEE Transactions on Industrial Electronics (preprint)*, 2015.

[26] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of ACM MobiCom*. New York, NY, USA: ACM, 1999, pp. 174–185.

[27] W. Dong, C. Chen, X. Liu, G. Teng, J. Bu, and Y. Liu, "Bulk data dissemination in wireless sensor networks: Modeling and analysis," *Computer Networks*, vol. 56, no. 11, pp. 2664–2676, 2012.

[28] J.-W. Li, S.-N. Li, Y. Zhang, Y. W. Law, X. Zhou, and M. Palaniswami, "Analytical model of coding-based reprogramming protocols in lossy wireless sensor networks," in *Proceedings of IEEE ICC*, June 2013, pp. 1867–1871.

[29] "Tinyos operating system," http://www.tinyos.net.