Towards a Wireless Building Management System Requiring no Change to Upper-layer Protocols

Qinghua Luo[*†], Abraham Hang-yat Lam[*§], Dan Wang[*], Daniel Wai-tin Chan[‡], Yu Peng[†] and Xiyuan Peng[†]
[*]The Hong Kong Polytechnic University,[†]Harbin Institute of Technology,[§]Building Integration Perfection Ltd., [‡]Nanchang University
Email: {csqluo, cshylam, csdwang}@comp.polyu.edu.hk, chanwaitin.daniel@gmail.com, {pengyu, pxy}@hit.edu.cn

***Abstract*—There are many recent efforts in developing wireless or partially wireless Building Management Systems (BMS). A key difference from developing a brand new wireless sensor application, where one can design the system from scratch, is that there exist wired building management systems and there is a full set of upper layer protocols developed and standardized. On-going studies on wireless BMS usually develop smart sensor hardware and re-design protocols from bottom up. Such approach requires a long time for standardization and adoption. In this paper, we study from a new direction by proposing a general framework that converts existing wired sensor network into wireless without changing upper layer protocols and the existing hardware. The key ideas are an asynchronous-response framework to maintain the control plane of the upper layer protocol intact, and a modular design to prioritize and schedule data flows in case of link quality and throughput variation. Our design can become a supplement for existing studies towards developing wireless building management systems. We evaluate our system using both comprehensive simulations and experiments with real BMS hardware, software and protocols running on top. We made a field deployment by integrating our system into the BMS of the FG-building of The Hong Kong Polytechnic University. The system operated smoothly during our five-hour deployment.**

I. Introduction

After a decade of research, we have decent understanding on the design within a wireless sensor network, e.g., OS, programming languages, routing, MAC, etc. People are now actively studying application scenarios so that wireless sensor networks can be more pervasively used in our society. Currently, research on wireless sensor network applications mainly focus on proposing new frontiers, from volcano monitoring [37], interactive sensor maps [40], new energy conservation systems [26], to participatory sensing [7].

There is another type of applications that is also promising, i.e., using wireless sensor networks to enhance existing wired sensor systems or convert them to wireless systems. Wired sensor networks have already been used for ages in such domains as building management systems, structural health monitoring, industry and manufactory, to name but a few. A wireless sensor network has great advantages for its wireless communication, storage and processing power of the smart sensors; all these can lead to a system that is much cheaper, more readily deployable and flexible.

Recently we are developing a (partially) wireless Building Management System (BMS). A BMS acts as the brain of a building in controlling and monitoring its mechanical and electrical equipments (see Fig. 1). A wireless BMS system can be cheaper and more flexible [39]. Especially, when there needs room re-configuration (e.g., room partitioning), room facility modifications (e.g., add/remove lighting, air-conditioning), etc, substantial cost savings are expected over a wired system. As the smart sensors have processing power, it is also more flexible for the system to incorporate new innovations, such as new energy conservation schemes [30].

We consider a key difference between proposing a new application of wireless sensor networks and converting an existing wired sensor network into wireless is that developing new applications has more freedom. Designers can choose sensor hardware, select suitable wireless communications to support transmission, design protocols for sensor networking and conduct application-oriented system integration. On the contrary, for existing wired sensor systems, they usually have a full set of upper layer protocols developed, standardized and in operation for some time. For example, for BMS, there is a Building Automation and Control network (BACnet) [3] protocol that specifies the interaction of the sensing devices (e.g., lighting, heating, ventilation and air-conditioning, etc), the Direct Digital Controllers (DDCs, i.e., the data "relays") and the operation centers of a building. BACnet is developed, standardized and assumed-to-be on top of a wired network.

To develop wireless sensor systems for existing applications, recent studies usually re-design the system and make non-trivial modifications on the upper layers. For example, sMAP is proposed as a new physical information collection framework and can be used in buildings [12]. ZigBee Alliance is developing standards to support BACnet using ZigBee communication [3]. System redevelopment can provide us a chance to re-think system requirements and enhance system functionalities, e.g., see new requirements for home automation and building automation specified in RFC [6][28].

System redevelopment, however, usually takes a long time for standardization and wide adoption/deployment. It also faces possible backward compatibility problems. In this paper, we look from a new angle by proposing a framework that can convert existing wired sensor network into wireless with no change of the upper layer protocols and no intruding extension for the existing hardware. We emphasize that, however, our scheme does not substitute system new designs. We believe that the two can well co-exist with each other.

Our framework faces two key difficulties: 1) we need to maintain the control plane of the upper layer protocols in operation. We will show that this cannot be easily realized as protocol commands can have time constraints that are difficult to meet using wireless links. Even worse, the constraints cannot be achieved by simply increasing the bandwidth of wireless communication; and 2) the throughput and quality of wireless communications are worse than that of wires. We need to maintain the data plane of the upper layer protocols in the sense that it can satisfy application requirements.

In this paper, we propose a novel asynchronous-response scheme to maintain the control plane of the upper layer protocol intact. We also have a modular design for wireless data plane to prioritize and schedule data transmission in case of link quality and throughput variation. We evaluate our framework through 1) experiments using real DDCs that are connected with Arduino sensors. The experiments show the effectiveness of our asynchronous-response idea, and system performance under real building protocols and software; and 2) a field deployment of our system, integrated into the existing BMS in the FG-building of Hong Kong Polytechnic University. Our system operated smoothly in our 5-hour deployment.

We believe our framework and experience can be generalized for other wired applications beyond BMS. To the best of our knowledge, our work is the first towards a (partial) wireless system for BMS without modification of building protocols. We develop our program codes as an open source in [25].

The remaining part of this paper proceeds as follows. In Section I-A, we discuss some BMS background and taxonomy used in this paper. We give an overview of our design in Section II. Section III has the design details on the asynchronous-response module and the wireless data transmission modules. In Section IV, we present implementation details, which are imperative for effective system operations. We evaluate our framework in Section V and a real world deployment is shown in Section VI. In Section VII, we present related work and finally, we conclude our paper in Section VIII.

A. *BMS Background and Wireless Communications in BMS*

We first briefly introduce BMS architecture and building protocols (see Fig. 1). The BMS acts as the brain of a building in controlling and monitoring the mechanical and electrical equipments of a building. In BMS, physical data are recorded by sensing devices. These sensing devices are passive sensors (e.g., smoke detectors). To make our presentation clear, in the follows, we call them *sensing devices* the passive sensors in BMS; and *sensors* the active smart sensors that have the ability of processing, storage and communication, as widely understood by computer scientists. The sensing devices in BMS are connected to the Direct Digital Controllers (DDC). The DDCs form the hardware backbone of BMS. There are two types of DDCs: system DDCs (usually more powerful) and field DDCs (or in short, DDCs). The physical connection of the DDCs is RS-485, a physical layer standard. On top of RS-485, there is an MS/TP (Master Slave/Token Passing) protocol for DDC connections. The system DDCs are connected to the operation center using Ethernet. The protocol of BMS is BACnet, standardized by ASHRAE [3]. BACnet defines the interaction behavior of the BMS devices. There are variations in BMS architecture. We believe, however, this aforementioned architecture is one most widely adopted architecture.

There are many wires in BMS. The Ethernet uses wire. This section is usually well planned along with the building construction and requires less flexibility. Converting this section into wireless is thus, less fruitful.

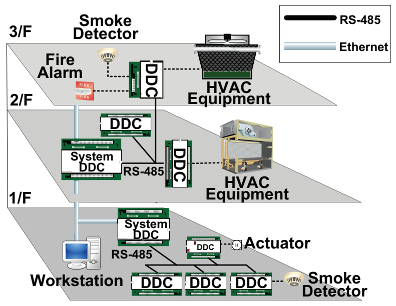


Fig. 1. The BMS architecture

The connection between the DDCs and the sensing devices uses wire. There is a large number of different sensing devices (e.g., smoke detector, thermostat, etc). These sensing devices are passive, vendor oriented and the connection is point-to-point. Thus, we consider converting this section into wireless is more specific and if these sensing devices are enhanced by the smart sensors, it is easier to individually convert this section into wireless and integrate into the BMS architecture.

The connection between DDCs uses wire. This section is more flexible than the Ethernet section. The distance between DDCs can be long (between floors as shown in Fig. 1) and the DDCs form a subnet. As such, converting this part into wireless has large gain, yet it also faces non-trivial challenges. This is the focus of this paper. The protocol governs this part is MS/TP, developed specifically under BACnet by ASHRAE. Every DDC in this subnet has two roles, master or slave. There is a token in the subnet and a DDC can send data frames or command frames when it holds a token.

It is worth noting that we do not say to tear down existing building BMS. This may not save money. (Partially) wireless BMS are more expected for future buildings. As BMS uses devices from many vendors, new designs and standard development clearly take longer time and more costs.

II. An Overview of the Design

The physical change made by our system is illustrated from Fig. 2 (a) to Fig. 2 (b). We attach a sensor to a DDC through RS-485. We leave more details on sensor hardware selection and development to Section IV. Note that we make no modification on the DDC hardware.

In this example, the communication from DDC-1 to DDC-2 is replaced by communication from DDC-1, relayed by Sensor-1 and Sensor-2, to DDC-2. Our experience shows that a straight forward replacement does not work. The are two problems. First, for each frame sent by DDC-1, there is a delay constraint. More specifically, if this frame is not received (or replied) within a certain amount of time, it is considered expired.[1] In MS/TP of the BMS system, this time is 10bit propagation-time on RS-485 communication. While this delay

[1]This is a common design for a system to get rid of outdated or lingering packets/frames. For example, in Internet routing, there is a maximum hop number constraint for each packet. A packet should be dropped if the number of hops exceeds this number.

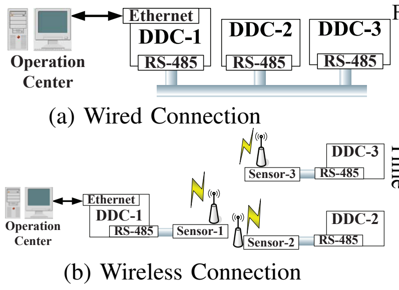


Fig. 2. Converting wired connection to wireless

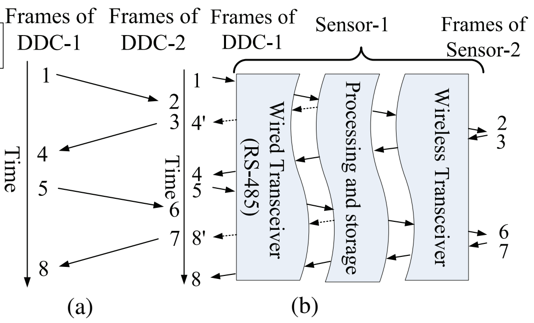


Fig. 3. Asynchronous Scheme Illustration

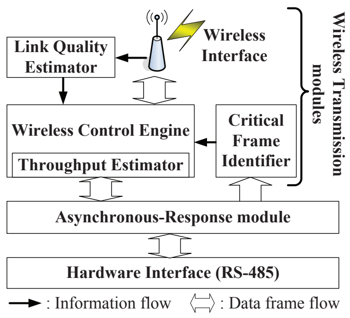


Fig. 4. A Modular Framework

can be easily satisfied by a wired link, for a wireless link such as 802.15.4, the processing delay and the propagation delay make it impossible to meet such delay. As a matter of fact, we measured a 1500bit time delay in our experiments. Even worse, this cannot be improved by increasing bandwidth. This introduces difficulty in maintaining the control plane of the upper layer MS/TP protocol. Second, the transmission of wireless links is slower and more unstable than wired links. Thus, the data throughput from the application layer may exceed the wireless link capacity at certain time. This affects the data plane and data storage and scheduling are needed.

Our key proposal is a novel asynchronous-response scheme (see Fig. 3). The sensors run MS/TP protocol stack to communicate with their corresponding DDCs. For a command from DDC-1 (refer to Fig. 2), Sensor-1 will send this command to Sensor-2. In the meantime, if it needs to meet the MS/TP timing constraint, Sensor-1 will also send a valid MS/TP protocol command to DDC-1. This response is asynchronous to the request sent/received from Sensor-2. Sensor-2 will send the request to DDC-2 and then respond to Sensor-1 after receiving response from DDC-2. To the best of our knowledge, we are the first to introduce such design in our scenario.[2]

We develop a wireless BMS framework using a modular design (see Fig. 4). The asynchronous-response module reads frames from RS-485. It passes outgoing frames to the wireless transmission modules for further process and transmission.

As said, the data traffic may be greater than the capacity of the wireless link at certain times. Therefore, we need to schedule data frames while satisfying the application requirements (e.g., timely update of the readings of the sensing devices, accurate critical event report, etc). We thus have a set of wireless transmission modules to support data transmission. We have a link quality estimator module which monitors the wireless communication quality. If the link quality deteriorates, more retransmission is allocated. We have a critical frame identification module. With the understanding of application traffic pattern, we can identify critical frames and assign high priority for these traffic in case of need. We have a wireless control engine assisted by a throughput estimator. It monitors the data traffic information from other modules and makes data scheduling and transmission decisions.

[2]Asynchronous designs have been used in other domains, e.g., Ajax was proposed to improve the response time of web pages by asynchronizing the user-browser communication with the browser-web server communication.

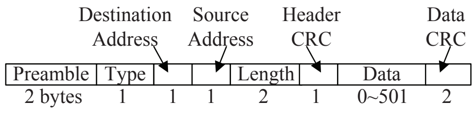


Fig. 5. MS/TP Frame format

Here we specifically omit a routing module, which is important if the communication range is constrained and relays are needed. We make this paper focus on the main challenges towards holding the upper layer protocols unchanged. The routing module is thus delayed to future work.

III. Design Details

A. The Asynchronous-Response Module

We look into the details of the MS/TP protocol. The frame format is shown in Fig. 5. Each DDC has an address. In our implementation, we also give each sensor associated with the DDC an address. There are eight public commands and some proprietary commands. We do not study the proprietary commands, as they can be handled through individual vendors if necessary. For the public commands, we group them into two categories: 1) MS/TP link and system maintenance frames; and 2) data transmission frames (see Table I).

TABLE I
MS/TP Frame specification and classification

Type	Name	Category
00	Token	Link maintenance
01	Poll For Master	Link maintenance
02	Reply to Poll For Master	Link maintenance
03	Test_Request	System maintenance
04	Test_Response	System maintenance
05	BACnet Data Expecting Reply	Data transmission
06	BACnet Data Not Expecting Reply	Data transmission
07	Reply Postponed	Data transmission

We describe these commands (following specification from [3]) and how we handle them individually.

- **00 Token**: This command is to pass network mastership to the destination node. Only the token holding node can send data. *In our implementation*, when Sensor-1 receives **00 Token** from DDC-1, it checks if there are valid data frames asynchronously received and stored. If there exist, it sends these data frames to DDC-1. Sensor-1 returns **00 Token** to DDC-1 after it finishes sending data or if it does not have any data to send.
- **01 Poll For Master**: This command is transmitted by nodes during configuration and periodically during normal network operation. It is to discover the presence of other nodes in the network and to determine a successor node in the token ring.

- **02 Reply to Poll For Master**: This command acts as a reply to the Poll For Master frame. It is to indicate that the node sending the frame wishes to enter the token ring, or is alive.
  These two commands are used for checking whether a successor node exists (i.e., just joined the network) or is alive. For example, when a DDC (e.g., DDC-1) sends **01 Poll For Master**, its successor node (e.g., DDC-2), if exists or alive, replies **02 Reply to Poll For Master**. DDC-2 can continue this process to check its own successor node.
  In our implementation, Sensor-1 will periodically send **01 Poll For Master** to its successor Sensor-2 asynchronous to whether DDC-1 queries Sensor-1 or not. Sensor-2, when receives this command, will send to DDC-2 to check its livelihood. When Sensor-1 receives command **01 Poll For Master** from DDC-1, it replies **02 Reply to Poll For Master** directly if in the previous round, it finds that DDC-2 is alive.
- **03 Test_Request**: This command is used to check if loops exist in the MS/TP to MS/TP transmission paths.
- **04 Test_Response** This command is used to reply to Test_Request frames.
  These two commands are used at the network set-up stage. Since there are many MS/TP subnets in a BMS, these are used to check if there exists loop from one MS/TP subnet to another MS/TP subnet.
  In our implementation, when Sensor-1 receives command **03 Test_Request** from DDC-1, it replies command **07 Reply Postponed** (more details later), which means that the Test_Response frame is not ready and is postponed. In the meantime, it also passes this command on to the sensor according to the destination address in the frame (e.g., Sensor-2). When a sensor (e.g., Sensor-2) receives **04 Test_Response**, it relays the command to the destination sensor according to address in frame, or to DDC-2 if the destination address is itself.
- **05 BACnet Data Expecting Reply**: This command is used by master nodes to convey the data frame which expects reply.
- **06 BACnet Data Not Expecting Reply**: This command is by master nodes to convey a data frame that does not expect reply.
  These two commands are used for data transmission.
  In our implementation, Sensor-1 sends **07 Reply Postponed** to DDC-1 when it receives **05 BACnet Data Expecting Reply**. In the meantime, it relays this command to the sensor according to the destination address in the frame. When a sensor receives **06 BACnet Data Not Expecting Reply** it relays this command to the sensor according to the destination address in the frame.
- **07 Reply Postponed**: This command is used by master node to defer sending a reply to a previously received BACnet Data Expecting Reply command.
  In our implementation, when Sensor-1 queries Sensor-2 and Sensor-2 queries DDC-2, Sensor-2 can receive **07 Reply Postponed** when the data of DDC-2 is not ready (For example, DDC-2 does not get data from its associated sensing devices, e.g., a thermometer). A tricky part here is that Sensor-2 will *not* send **07 Reply Postponed** to Sensor-1 as it knows that Sensor-1 has already asynchronously sent a **07 Reply Postponed** to DDC-1. Sensor-2 will also pass the token to DDC-2 by sending **00 Token**; otherwise, DDC-2 does not have the right to send data. In case that DDC-2 is still not ready, DDC-2 will pass the token back to Sensor-2 (see previous explanation on the token). The token passing continues between Sensor-2 and DDC-2 until the data are ready.

Our evaluation in Section V, VI show that our asynchronous response module successfully supports system operation.

B. The Wireless Transmission Modules

We assume that a wireless link is slower and more unstable than a wired link. Note that we do not say that every wired application can become wireless without modification of the upper layers. If the difference between the wire and wireless communication speed is big and the application requirement is stringent, holding upper layers unchanged can be impossible.

For BMS, the data traffic (especially the averaged data traffic) is moderate. Based on our experience we often see that even the whole traffic is manageable by wireless capacity. We will use scheduling and priority to achieve smooth data transmission under traffic and link quality variation.

We classify two different traffic categories: 1) data traffic for regular monitoring of the sensing devices; and 2) data traffic for emergency report. We present our wireless transmission modules and show how these traffic are supported.

1) Critical Frame Identification: There are emergency reports in BMS. More specifically, the BACnet can define an emergency by setting a threshold for a sensing device. For example, an emergency can be defined as temperature above 140°F(60°C). When an emergency happens, a DDC will detect the emergency by its associated sensing devices. The DDC then generates emergency critical frames to report to the operation center. We develop critical frame identification algorithms (CFI) to identify these reports and these frames will be prioritized in transmission.

We present two CFI methods: determine critical frames by 1) specific data fields; and 2) frame pattern recognition.

Specific Data Fields: In the data field of an MS/TP frame, there is a special "service choice field". For critical events, this field will be labeled to 1 (i.e., security), 2 (i.e., critical), or 4 (i.e., fire). By inspecting this data field, we can identify critical frames. Using specific data fields is simple yet we admit that this violates framework layering to certain extend as we have to inspect the data content.

Pattern Recognition: In some applications, the data field in the frame may be encrypted or the data is not allowed to open. We thus identify the critical frames using pattern recognition. We found that the data frame pattern in BMS is very regular. We show an example in Fig. 6. The frame length of the critical frames is different from that of the regular frames. This is reasonable as the traffic in BMS is regulated according to specific buildings and monitoring procedure. Thus, we develop a simple pattern recognition scheme as follows.

We use frame length as the criteria to differentiate regular frames and critical frames. Since the data pattern is correlated to individual buildings, we need a first round training for the frame lengths. In the training period, we run the system for a period of time when no critical event happens. During this period, we record the set of all regular frame lengths. In the operation period, whenever a frame has a length that is not in the set, we will mark it as a critical frame.

We will show in our experiments (Section V-B), that both CFI methods can achieve 100% accuracy.

2) Link Quality Estimation: Wireless link management has long been a research topic. We are working on a token passing protocol. Thus, we do not face serious interference and collisions. We need to handle link quality deterioration, however. The main factors that affect link quality are distance

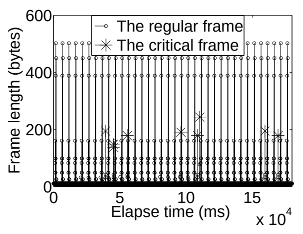


Fig. 6. Critical Frame Identification

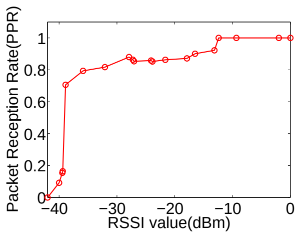


Fig. 7. The Correlation between PRR and RSSI

and blockage. Since the BMS system is designed in a building, we believe that the distance can be more or less measured in advance. The blockage is caused by temporary (e.g., a few days to months) room separation, decoration, etc where walls or Christmas trees are installed. Such blockage should be detected and transmission adjustment is required.

There are many methods to detect link quality change. Based on hardware, there are RSSI [33], LQI, SNR [15], etc. Based on software, there are PRR [32], RNP [9], etc. In our application, we need a light-weight scheme because the sensor CPUs are loaded with many tasks and their processing power is not strong for complicated schemes. We choose RSSI mainly to show how this module fits in our framework to provide input for the wireless control module. Other schemes can be used as well. RSSI has drawbacks [5] yet its advantage is that it does not require additional hardware and much computation.

The RSSI average has correlation with packet reception rate [33]. We conducted our own indoor field test and confirmed such correlation (see Fig. 7). We handle link quality deterioration by increasing the number of retransmissions. We compute the number of retransmissions as follows. Let r be the number of retransmissions. Let $\mathcal{L}$ be the packet loss rate. We obtain the relation between RSSI strength and packet loss rate $\mathcal{L}$ from pre-measured statistics. A pre-measurement is reasonable as the link distance in our system does not change unexpectedly. To compute an expected frame success rate Δ, we have

$$1 - \mathcal{L}^r \geq \Delta \Rightarrow r \leq log_{\mathcal{L}}(1-\Delta) \qquad (1)$$

We also put a threshold $\mathcal{R}$ on r. If $r > \mathcal{R}$, the sensor indicates that the link is broken by not sending any data on this link. The operation center will not get any data frames from this DDC, and will show a broken icon on the link on the monitoring screen. This threshold is used to protect the sensors from retransmission overloads.

Note that if there is serious wireless link blockage in case of room renovation, even if a wired network is used, system re-deployment may also be needed. Severe building renovation (and initial building deployment) should consider the BMS restructuring. Such planning is out of the scope of this paper and worth a separate study.

3) Wireless Control Engine: The wireless control engine transmits the critical frames and the control frames directly. For the data frames, there is a throughput estimator sub-module which monitors the traffic intensity from the application layer. If the data traffic is less than the residual wireless capacity, all frames will be transmitted.

We consider the case where the data input from RS-485 is greater than the data output to wireless link. Let V_{in} be the data input speed and V_{out} be the data output speed. We delay the details on computing V_{in} and V_{out} in Section IV-C. Since $V_{in} > V_{out}$, some data have to be dropped. In BMS, this means filtering out some regular traffic. From the user application point of view, the refresh interval of the sensing devices will be increased, e.g., we refresh the thermometers every 10 seconds instead of every 2 seconds.

We consider two user requirements: 1) *fairness:* if the refresh interval needs to be increased, all the sensing devices increase equally, and 2) *importance:* there might be certain important rooms/locations that need higher refresh rate, by compromising the refresh rate of other rooms/locations.

We first study the fairness requirement. In BMS, each DDC can connect to tens or even hundreds of sensing devices. The readings of multiple sensing devices can be combined in a data frame. Since the monitoring is regular, each data frame always has the readings of the same sensing devices. Therefore, as long as the data frames are transmitted fairly, we can guarantee fairness between different sensing devices. The transmission is divided into *cycles*. Each data frame has a serial number for a cycle and their frame lengths are different (see Fig. 8 (a)). In wired transmission, each cycle transmits the same sequence. For example, in Fig. 8 (a), each cycle will transmit 10 data frames. In wireless transmission, each cycle may not have enough capacity. Thus, we need to maximally utilize the wireless communication capacity and transmit each frame with equal interval in terms of cycles. For example, in Fig. 8 (b), we cannot transmit all 10 frames in one cycle. We show a transmission schedule that each frame is fairly separated with an interval of 2 cycles.

We next formally show how such schedule should be developed. Let $F_i, i \in \{0, \cdots, N-1\}$ be the frames where N denote the total number of frames. Let $l_i \in \{0, \cdots, N-1\}$ be the length of frame F_i. Let L_C be the maximum bytes that a cycle can send in wireless communication. Let $\mathcal{P}$ be an arbitrary period consists of $\mathcal{C}$ cycles and L_k be the total amount of bytes transmitted in cycle k. Let $L_m = \sum_{k \in \mathcal{C}} L_k$. Let T_i be the total number of times that F_i is transmitted in $\mathcal{P}$. Let $\mathcal{T}$ be a pre-defined threshold to bound the difference of T_i, $\forall i, j, |T_i - T_j|$, i.e., the fairness.

Definition 1 *The Transmission Sequence Problem (TSP): Find a transmission sequence for $F_i, i \in \{0, \cdots, N-1\}$ in any arbitrary period $\mathcal{P}$ that can be divided into $\mathcal{C}$ cycles, such that the total transmission $L_m = \sum_{k \in \mathcal{C}} L_k$ is maximized and the difference of the frames F_i transmitted is bounded by $\mathcal{T}$.*

Theorem 1 *TSP is NP-hard.*

Proof: We reduce TSP to bin packing problem, which is proven to be NP-hard. The bin-packing problem is defined as: given a bin size V, and a list of $a_1, a_2, ..., a_n$ of items to pack, find an integer B, and B partitions of $S_i \subseteq \{1, 2, \cdots, N\}$, where $\sum_{i \in S_k} a_i \leq V (k = 1, 2, \cdots, B)$, find the minimal of B.

For each instance of the bin packing problem, we construct

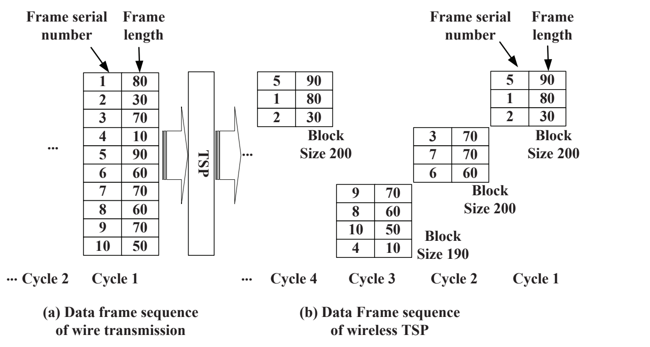


Fig. 8. The Data Frame Transmission

an instance of transmission sequence problem as follows. For each bin size V, create $L_C = V$; for each item size a_i, create frame length $l_i = a_i$. Let $\mathcal{T} = 0$. This indicates that each frame can only show up equally. This is the key of the proof as now we only need to think how to transmit N different frames in a minimum number of cycles. Let $\mathcal{P}$ be the period of the minimum number of cycles that transmit N frames.[3] Clearly, to find the minimum $\mathcal{C}$ is the same as to find a minimized B. This means that if TSP is solved, the bin-packing problem is solved. ■

We need an online algorithm for TSP. We first developed an offline algorithm and then extend it into an online version. Our offline algorithm follows the First Fit Decreasing (FFD) algorithm that is used for bin-packing problem [20]. The principle of FFD algorithm is to first sort all the items in an descending order, and then use a greedy method to put the items into bins. Our algorithm follows a similar principle by first putting large frames into cycles. It checks in each iteration the number of times a frame transmitted so that $\mathcal{T}$ is never violated. Our online algorithm applies the offline algorithm for each cycle.

Algorithm 1 $TSP()$

```
Input: $F_i$, $l_i$, $L_C$, $\mathcal{T}$
Output: $\text{Bin}_k, L_k, k \in N$
 1: $T = \{T_1, T_2, ..., T_i, ..., T_N\} = \{0\}$
 2: $k = 0, \text{Bin}_k = \emptyset, L_k = 0,$
 3: $l' = sort(l) = \{l_q, ..., l_i, ..., l_p\}$
 4: $F' = \{F_q, ..., F_i, ..., F_p\}$, $T' = \{T_q, ..., T_i, ..., T_p\}$
 5: for ($i = 1$ to $N$) do
 6:     for ($k = 1$ to $N$) do
 7:         if ($L_k + l'_i \le L_C$) and ($T'_i - T'_{min} < \mathcal{T}$) then
 8:             $\text{Bin}_k = \text{Bin}_k \bigcup F'_i$, $L_k = L_k + l'_i$, $T'_i = T'_i + 1$
 9:             break
10:         end if
11:     end for
12: end for
13: return($\text{Bin}_k, L_k$)
```

Lemma 1 *The complexity of algorithm TSP() is O(N^2).*

Proof: The complexity of the quick sort subroutine is O($N \log(N)$) and the complexity of nested loop structure is O(N^2). Consequently, the complexity of TSP() is O(N^2). ■

The fairness requirement is per DDC based. For the importance requirement, if certain sections of the building needs a higher refresh rate, we choose to give the DDC associated with this section a longer timeout when the token arrives at it so that it can transmit more. More specifically, let $\mathcal{N}$ be the number of DDCs. Let $p_i, i \in \{1, \mathcal{N}\}, p_i > 0$ denote the priority of the ith DDC. The lower the priority is, the longer timeout the DDC has. Let t_r be the refresh interval, we set the timeout u_i of the ith DDC to be $u_i = \frac{(1/p_i)\times t_r}{\sum_{j=1}^{\mathcal{N}} 1/p_j}$.

IV. Implementation Details

A. Sensor Connection with a DDC

The output of DDCs is RS-485. We choose the Arduino sensors as we can use the I/O Expansion shield [41] developed by DFROBOT community, which directly supports RS-485.

For the sensors that do not have RS-485 input/output (e.g., Imote2), we need to develop an extension board. To develop the extension board for Imote2, we select the STDUART as the data interface and GPIO 94 control signal as the RS-485 transceiver chips of the Imote2. We select MAX3485 as the RS-485 transceiver. For electrical security, we add an optically coupled isolator between the MCU and RS-485 transceiver. For optically coupled isolator, we select TLP521. The Imote2 version sensor can be found in [25].

B. Fast Forward of Frame Transmission

For an Arduino sensor, it needs to connect from RS-485 to DDC to process MS/TP frames. In the mean time, it also needs to process wireless data frames. There is a big gap between the speed of wireless interface (which is slow) and the CPU speed (which is fast). As a result, it can take a long time if an Arduino sensor sends a frame of more than 300 bytes, such as 05 BACnet Data Expecting Reply, 06 BACnet Data Not Expecting Reply, 03 Test_Request, 04 Test_Response. During this period of time, its CPU cannot effectively process the MS/TP frame from RS-485. During our experiments, the Arduino sensor can become unstable or even malfunction if we operate data transmission frame-by-frame.

To handle this problem, we use a *fast forward strategy for frame transmission*. More specifically, the CPU sends in the granularity of each byte instead of each frame to the wireless interface. Since CPU process is much faster than wireless interface, the interval of each byte is small. From the wireless transmission point of view, its neighboring sensor still sees an integrated data frame. With fast forward frame transmission, the Arduino sensor operates reliably and effectively.

C. A Wireless Token Ring Network

In wired network, when a DDC sends a frame, it broadcasts in the physical link. After we connect each DDC with a wireless sensor, if we still use broadcast for sensor to sensor communication, we need to specially handle interference and collision. In our implementation, we use unicast between the sensors by constructing a wireless token ring among sensors. A sensor can transmit only when it has a token. We emphasize that this token ring is in a lower layer and should not be confused to the Token Passing protocol among DDCs.

[3]To be more rigid, this should be transformed into a decision problem where given $\mathcal{P} = i$, determine whether N frames can be transmitted. We took this for granted for the sake of conciseness.

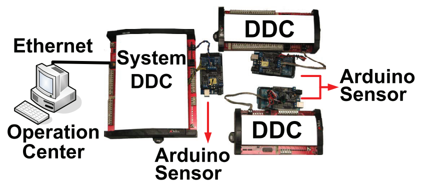


Fig. 9. The experiment environment

We describe how our implementation computes V_{in} and V_{out} of Section III-B3. V_{out} is computed by the wireless interface speed multiplied with a piece of token time in this wireless token ring network. V_{in} is computed by the total amount of data frames in a cycle divided by cycle length time.

V. Performance Evaluation

We evaluate our system through experiments, simulations, and a real deployment. The experiments evaluate system functions that are difficult to simulate, e.g., the asynchronous-response module and also the system performance under real environments. The simulation can scale our evaluation. It also helps to evaluate many algorithm details, e.g., for the wireless control engine. We also conduct field deployment of our system in the FG-building of Hong Kong Polytechnic University.

A. Experiment Setup

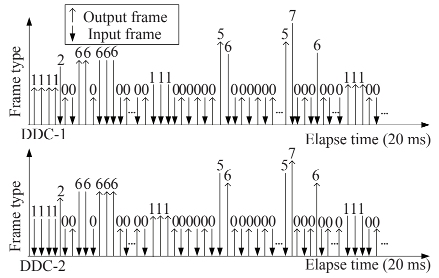
The hardware used in our experiment are shown in Fig. 9. We have three DDCs manufactured by Delta Controls Ltd., one system DDC and two field DDCs. Each DDC is connected with an Arduino Mega 2560 sensor through RS-485 as discussed in Section IV-A. The wireless module adopted in our design is XBee [41], which is based on IEEE 802.15.4 standard. A PC is used to act as the operation center using real building management software ORCAview 3.33 [42]. The system DDC connected with the PC using Ethernet, and it communicates with other DDCs using regular MS/TP protocol. Through ORCAview, we can monitor, manage and configure the sensing devices of the DDCs. The traffic injected into and received from system DDCs are from ORCAview, which represents real traffic of BMS. The DDC hardware and software ORCAview are all off-the-shelf products and no modification is made.

In our experiment, we put our operation center (PC) and system DDC at a fixed place. We put the three DDCs with a height of 1 meter and they formed an equilateral triangle, separated with each other by 10 meters. The default transmission power is set to 0dBm. We conducted an preliminary measurement on the link quality using different distances where the distance changes from 5 meters to 40 meters. Especially, we put one field DDC and system DDC out of sight to each other. The results are in Fig. 7 which shows a sharp decrease in the PRR.

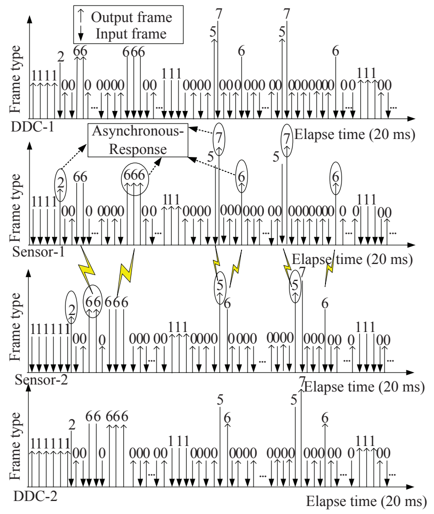
Our objectives are to evaluate the operation of our asynchronous-response module and system performance.

B. Experiment Results

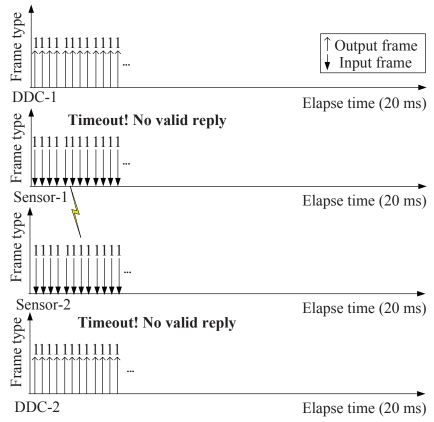
We first evaluate our asynchronous-response module. We draw three figures: 1) command sequence when wired system is used, 2) command sequence when our wireless system (with asynchronous-response) is used, and 3) command sequence



(a) The frame sequence of two DDCs



(b) The frame sequence of asynchronous-response framework



(c) The frame sequence of directly wireless replacement

Fig. 10. The frame sequence of different system.

when a wireless system without asynchronous-response is used. For all three figures, we perform the same operations. We see the results in Fig. 10. Each command is plotted as an arrow and the number shown on top of the arrow is the command type as discussed in Section III.

In Fig. 10 (a), we see two sub-figures, one shows the commands of DDC-1 and one shows the commands of DDC-2. We see that the operations start with a few **01 Poll for Master** followed by obtaining a token **00 Token**, followed by a few data transmission **06 BACnet Data Not Expecting Reply**, followed by token, poll for master and again data transmission **05 BACnet Data Expecting Reply**.

In Fig. 10 (b), we see four sub-figures, each one shows the commands of DDC-1, Sensor-1, Sensor-2 and DDC-2. We

TABLE II
ASYNCHRONOUS-RESPONSE RATIO

Refresh time	5s	10s	30s	60s
AR ratio	2.5%	1.27%	0.64%	0.32%

label the asynchronous-response frames in circle. For example, we see that when DDC-1 sends **01 Poll for Master**, Sensor-1 replies **02 Reply Poll for Master** asynchronous to relaying this command to DDC-2 (via Sensor-2). We also see that Sensor-1 sends **07 Reply Postponed** to maintain the operation when it does not receive in time data reply from DDC-2 (via Sensor-2).

In Fig. 10 (c), we plot a comparison, where we do not use the asynchronous-response module. There are also four sub-figures. We see that the communication breaks after very few command due to no valid reply before timeout.

We also show the ratio between the number of asynchronous-response frames and MS/TP frames in Table II. We see that when the sensing device refresh time increases, the ratio decreases. This indicates that most asynchronous-response frames are used to support data frames.

We next study the performance of our system. Since the application data are generated by the DDCs, we cannot freely manage them. We adjust the wireless speed (physical speed from hardware) to simulate the imbalance between traffic from the wire to the wireless. In Fig. 11, we plot the wireless throughput under different refresh intervals of 15 seconds, 5 seconds and 1 second. Note that different refreshing time intervals represent different traffic intensity from the application layer where 1 second refreshing interval has the highest data traffic. We see from Fig. 11 (a) that the traffic throughput is the same at 126.5Kbits for all different wireless speeds. This means that the data traffic is small (126.5Kbits generated every 15 seconds) and wireless capacities are enough to transmit the data frames. In Fig. 11 (b), we see that for wireless speed 76800bps and 57600bps, the throughput remains 126.5Kbits. However, when we decrease the wireless speed, the throughput decreases. This shows that our scheme starts to adjust if the output capacity is less than the input traffic flow. We can also see that almost all capacity is used. For example, if the physical wireless speed is 19200bps, 95Kbits is transmitted which is the full capacity in a 5-second refresh interval. We see this more clearly in Fig. 11 (c) where the total application layer traffic is the highest.

In Fig. 13, we plot the real wireless throughput rate (Kb/s) as opposed to absolute throughput (Kb) in Fig. 11 and Fig. 13 (a)(b)(c) corresponds to Fig. 11 (a)(b)(c). We see that when there are fewer data to transmit (Fig. 13 (a)), the throughput rate are the same (84300bps) no matter which physical wireless speed is used. The throughput rate increases as the amount of data increases, but will be bounded by the physical wireless speed (Fig. 13 (b)(c)).

We then study different link quality. Our system conducts retransmission as explained in Section III-B2. From Fig. 12, we see that PRR is improved. Especially, unless the link quality is extremely bad, we achieve 100% of PRR.

We next evaluate our system under critical frames. Through ORCAview 3.33, we simulate a temperature sensing device as an AI (Analog Input) port in our real DDC. We set the threshold for an alarm to be 140°F(60°C). The refresh interval is 10 seconds and we change the values of this sensing device randomly. As such, when the value is greater than the threshold, an alarm critical frame will be generated. We run our experiments for three hours and we compare our results with real results from ORCAview 3.33.

Fig. 14 shows the results of our two CFI methods: using specific data fields and pattern recognition. It is not surprising to see that using specific data fields can achieve 100% identification rate. When using frame length pattern recognition, the identification rate improves when training time increases. After the training time is greater than 10s, the identification rate also achieves 100%. In practice, we believe it is enough if the training time is 2-3 times of refresh interval.

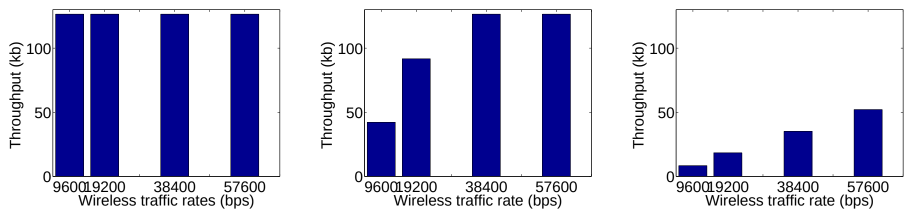
C. Simulation

1) Simulation Setup: In the simulation, we mainly evaluate the performance of our scheme under different conditions. The length of the data frames is randomly generated in [50, 503], which represents the real frame lengths. For wireless links, we set their speeds to standard XBee speed $1200, 2400, 3600, \cdots, 9600, 12000$bps. The total data traffic injected from the wired link is set to 0.5 - 8 times that of the XBee speed. The refresh interval of the sensing devices t_r is configured as 2s, 5s, 10s and 30s. The token timeout t_o is configured as 1s, 2s, $\cdots$, 9s. This t_o represents the time slice a DDC can send data when it holds a token. The default values of XBee speed is 1200bps, the total data traffic injected is 8 times, the data traffic from the wired link is 19200bps, $t_r = 15$s and $t_o = 1$s, the data quality is 0dBm, $\mathcal{T} = 5$.

Our main evaluation metrics are fairness and throughput ratio R, which is defined as the ratio between wireless throughput and data traffic injected from the wired link. R is used to remove the inconsistence of the absolute value of the throughput. As there is no similar work, we compare our results with 1) sequential transmission and 2) random transmission (when the wireless capacity is less than the traffic from the wired link).

2) Simulation Results: We first study throughput ratio. In Figure 15, we plot the throughput as a function of the sensing device refresh interval. Note that the smaller the refresh interval (i.e., more frequent refreshment), the higher the data traffic load. Clearly if the refresh interval t_r is small, the throughput ratio is small; for example, when $t_r = 4$s, the throughput ratio is 0.47 as not all the frames can be sent. When $t_r = 9$s, all the frames are sent and the throughput ratio $R = 1$. We also see that all three schemes perform equally. This is reasonable as there is no constraint on the wireless transmission.

In Figure 16, we compare the difference of the number of transmitted times of each frame, i.e., fairness, of the three transmission strategies. We can see that the maximum transmission difference of TSP strategy is bounded 5, our default threshold, while that of Sequential and Random strategy [10] is much bigger. Especially, the transmission difference of



(a) Refresh time 15s (b) Refresh time 5s (c) Refresh time 1s

Fig. 11. The throughput of system under different wireless speed in a refresh interval

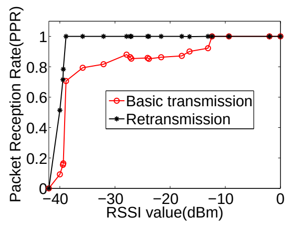
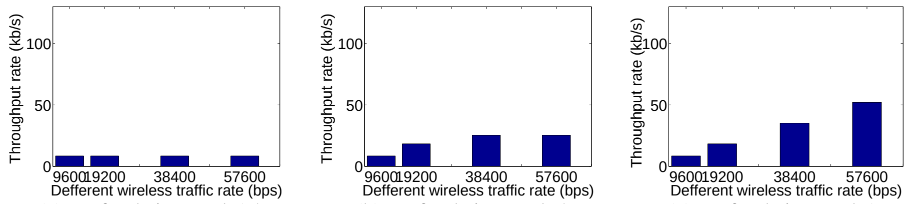


Fig. 12. The PRR under different link quality



(a) Refresh interval 15s (b) Refresh interval 5s (c) Refresh interval 1s

Fig. 13. The throughput rate of system under different wireless speed

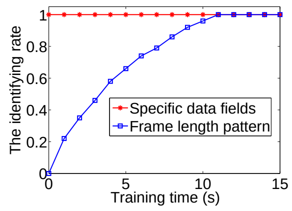


Fig. 14. CFI rate under different training time

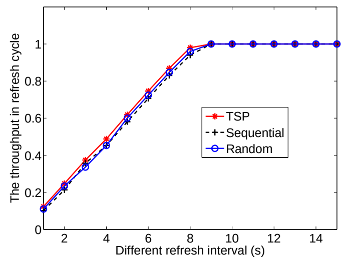


Fig. 15. The throughput of system in a refresh interval

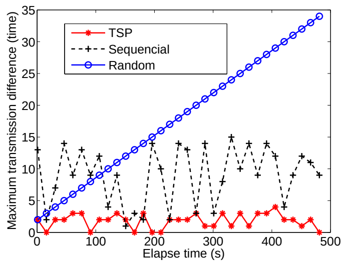


Fig. 16. The difference of the frames transmitted

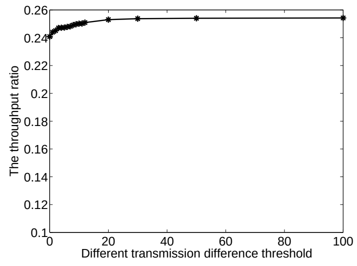


Fig. 17. The throughput ratio under different $\mathcal{T}$

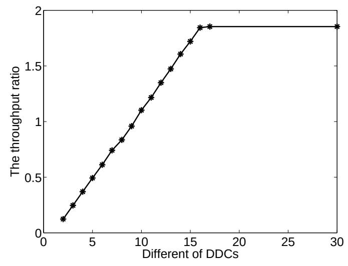


Fig. 18. The throughput under different number of DDC

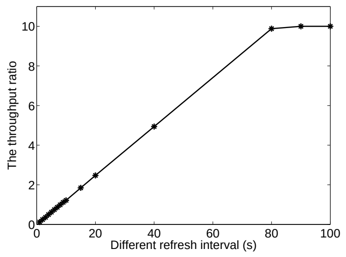


Fig. 19. The throughput ratio under different refresh rate

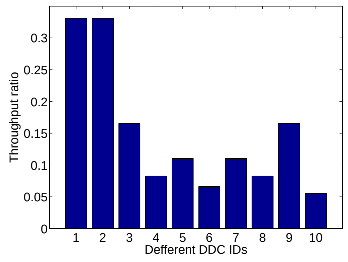


Fig. 20. The throughput under different priority

Sequential strategy increases with elapse time as it always transmits the first few frames.

In Figure 17, we adjust frame transmission threshold $\mathcal{T}$ and show the throughput ratio. We can see that if we allow a bigger $\mathcal{T}$, the throughput increases. Nevertheless, the increase is very small. Note that though a large $\mathcal{T}$ may bring freedom in frame selection, it only affects the last "bin". Therefore, from a system point of view, our algorithm is good enough as compared to more advanced algorithms.

In Figure 18, we show system throughput ratio as a function of the number of DDCs. We see that the total system throughput ratio increases when the number of DDCs increases. The ratio is stabilized at 15 DDCs. This is because the refresh interval is 15s and the wireless token ring timeout is 1s. Thus, the wireless capacity is fully utilized when there are 15 DDCs and the system throughput is maximized. In Figure 19, we set the number of DDCs to be 10, and we adjust the user refresh interval. We see that the wireless capacity is more released and the system throughput ratio increases when refresh interval increases.

In Figure 20, we evaluate the throughput ratio and fairness of 10 DDCs with different priorities. We randomly generate a set of the priority for each DDC. More specifically, the priorities for the DDCs are 1, 1, 2, 4, 3, 5, 3, 4, 2 and 6 respectively. We show the throughput of different DDCs where the x-axis is different DDC IDs. We see that the throughput ratio of each DDCs closely corresponds with their priorities.

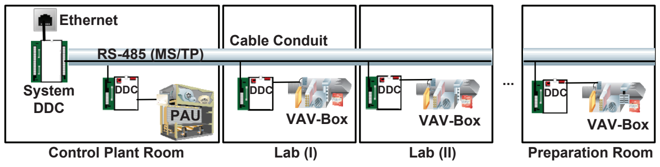


Fig. 21. The deployment of our system

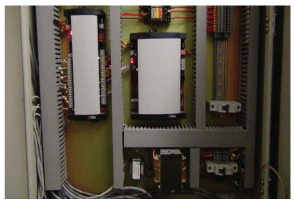


Fig. 22. Original wired MS/TP system

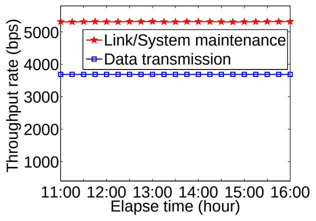
Fig. 23. Our on-site wireless deployment

VI. A Deployment Experience

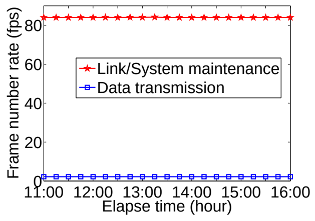
We deploy our system in room FG-417M (a BMS control room) of FG-building of The Hong Kong Polytechnic University. The 4th floor of FG-Building has a Learning Resource Center, a Data Communication Room, a Server Room, 3 Nursing Labs and 2 Mental Health Nursing Labs. There are 8 DDCs each of which controls a VAV (variable air volume) Box and other sensing devices. There is a DDC controlling a PAU (Pre-Cooling Air Handling Unit). These 9 DDCs are connected to a system DDC which is then connected to the BMS operation center. The configuration/topology map of the DDCs is shown in Figure 21. Since we do not have enough hardware, we only attach two Arduino sensors to the DDC controlling the PAU and the system DDC. We show our physical deployment in Fig. 23. There are two DDCs (the other 8 DDCs are spread in other rooms) in this control room (see Fig. 22), and we connect them to our sensors. Our XBee speed is set to be 57600bps. Note that our system is integrated into the BMS.

Our system was deployed on March 22nd, 2012 and ran for 5 hours from 11:00am to 4:00pm. Our system ran successfully and did not interrupt the normal operation of the whole BMS. From this deployment, our system not only works well, but also can be easily integrated into existing BMS systems.

In Fig. 24, we show the frame flow we captured every 15 seconds. We detail the traffic into data frames and maintenance frames. Fig. 24 (a) shows the amount of throughput and Fig. 24 (b) shows the number of frames. We see that 1) the traffic is stable and 2) control packets can be dominant. The total number of maintenance frames is 38.2 times greater than data frames and the throughput is 44.07% more.



(a) Throughput rate of system



(b) Frame rate of system

Fig. 24. Throughput rate and frame rate of the system.

In Figure 25, we show the throughput rate in the deployment system. The throughput rate between DDC-1 and Sensor-1 is 9000bps (see Figure 25 (a)). The wireless throughput rate between Sensor-1 and Sensor-2 is 3800bps (see Figure 25 (b)). The throughput rate between Sensor-2 and DDC-2 is again 9000bps (see Figure 25 (c)). These results conform to Figure 24 well.

VII. Related Work

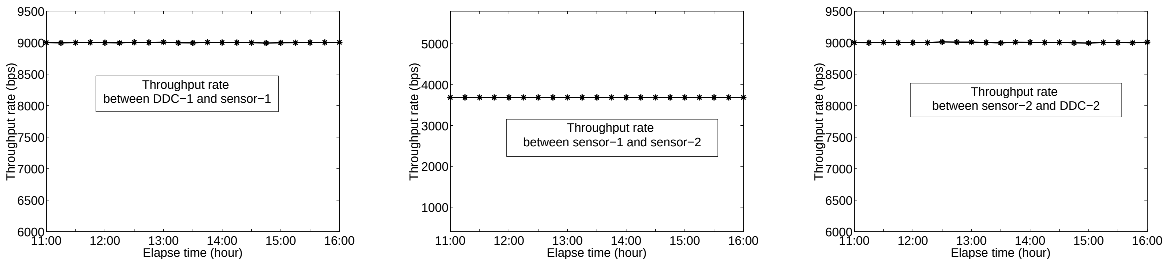
Building system development has been an active research topic recently for wireless sensor system community. There are two main research directions, one is on energy conservation [18], [17], [14], [22], [27], [31], [29], [26], [30], and the other is sensing system development which provides finer monitoring granularity or additional functions (e.g., to improve automation for comfort) [19], [12], [24], [2], [16], [35].

To save energy or improve energy efficiency, a system based on wireless sensor network is developed for high-fidelity monitoring of electrical usage in building [18]. Through occupancy prediction and real time occupancy monitoring via a sensor network of cameras, an HVAC control scheme is used to save energy [14]. There is a joint optimization of energy consumption [22] using occupant and equipment information processing based on cyber-physical systems. A general framework in building systems is proposed to share information to optimized energy management [27]. A model targeting at practical, wide-scale deployment to produce an ongoing breakdown of building energy consumption is presented in [31]. To improve the energy usage efficiency, forecasting the power demands based on occupant levels is proposed in [29]. Smart thermostat is proposed in [26] which automatically senses occupancy and sleep pattern to save energy. A new cyber-physical system-based control strategy for energy management in data centers is in [30].

To enhance building functions, an architecture of a Human-Building-Computer Interface system is deployed to increase personal comfort within building while reducing energy usage [17]. An automatic and robust breadcrumb system for fire fighter applications inside building is developed in [24]. A learning-based model predictive control (MPC) is presented in [2] to learn and compensate for the amount of heating to improve human comfort. A large-scale residential sensing system is deployed by [16] successfully.

sMAP is presented in [19], [12] as the architecture and specification of physical information collection, that can be used for sensors, meters and actuators in building environments. This is flexible that can incorporate new innovations. There are also standardization efforts [28], [6] for the requirements of future building automation and home automation. Our work can deal with existing building management systems and we believe the smart sensors can spare additional processing power to integrate with these new architecture.

There are abundant studies to improve wireless network throughput. A key difficulty is an accurate separation of interference, collision and link quality deterioration. A good



(a) Between DDC-1 and sensor-1 (b) Between sensor-1 and sensor-2 (c) Between sensor-2 and DDC-2

Fig. 25. The different throughput rate in real deployment system.

comparison of different schemes is dated back to [38] and a good related work survey can be found in [34]. More specifically, there are studies to improve throughput by stable link selection [21], [36], [23], [1], communication rate control [38], [34], [8] or retransmission according to link quality changes [11], [13]. From routing point of view, ExOR [4] is used to minimize retransmission and ETX [11] and ETT [13] are used for muti-hop routing.

We believe our framework can benefit from these advances schemes. However, we want to point out that unlike wireless computer networks, the sensor applications data traffic can be easier to handle. For example, we have seen that the BMS traffic is stable and we do not face interference much. More importantly, it is suggested that we look into the user requirements, and make adjustment accordingly, as these requirements can also be very specific.

VIII. Conclusion and Future Work

In this paper, we developed a framework which can convert existing wired building management system to a (partial) wireless system without modification on the existing building protocols. This is orthogonal and supplement to a brand new design which may take a long time to standardize and high development costs. The key of our approach is an asynchronous-response scheme that can support the upper layer protocol stack and a modular framework to support data transmission.

We believe our idea can be generalized to other applications, yet we also do not believe and claim that every existing wired system can become wireless without changing upper layer protocols. Our experience on BMS indicates one corner stone is that the speed difference between the wire and wireless links should be moderate. It is an interesting future work to rigidly study what characteristics are necessary for such conversion. For BMS specifically, as our study is the first, we also admit limitations for future work. First, we will conduct a verification using formal methods to show that our scheme is equivalent to the wired system in control plane. Second, we will develop a routing module when transmission relays are necessary. Note that anchor points could be needed as it is known that the network performance degrades if there are many wireless hops.

References

[1] M. H. Alizai, O. Landsiedel, J. link, S. Gotz, and K. Wehrle, “Bursty traffic over bursty links”, in *Proc. ACM SenSys’09*, Berkeley,Nov. 2009.

[2] A. Aswani, N. Master, J. Taneja, D. Culler, and C. Tomlin, “Reducing transient and steady state electricity consumption in HVAC using learning-based model predictive control”, in *Proceedings of the IEEE*, vol. 100, Iss. 1, pg. 240 - 253, 2012.

[3] ANSI/ASHRAE Standard 135-2010, BACnet: A Data Communication Protocol for Building Automation and Control Networks, in *American Society of Heating, Refrigeration, and Air-Conditioning Engineers Inc.*, Atlanta, GA, USA, 2010.

[4] S. Biswas and R. Morris, “ExOR: Opprotunistic multi-hop routing for wireless networks”, in *Proc. ACM SIGCOMM’05*, Philadelphia, PA, USA, Aug. 2005.

[5] N. Baccour,A. Koubaa,H. Youssef, et al, “F-LQE:A fuzzy link quality estimator for wireless sensor networks”, in *European Workshop on Wireless Sensor Networks’10*, Coimbra, Portugal, 2010.

[6] A. Brandt, J. Buron, and G. Porcu, “Home automation routing requirements in low-power and lossy networks”, in *RFC 5826*, Apr. 2010.

[7] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, “Participatory sensing”, in *Proc. WSW’06* in conjunction with *ACM SenSys’06*, Boulder, Colorado, USA, Oct. 2006.

[8] K. V. Cardoso and J. F. Rezende, “Increasing throughput in dense 802.11 networks by automatic rate adaptation improvement”, in *Wireless Networks*, Vol. 18, Iss. 1, pg. 95 - 112, 2012.

[9] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin, “Temporal properties of low power wireless links: modeling and implications on multi-hop routing”, in *Proc. ACM MobiHoc’05*, Urbana-Champaign,USA,2005.

[10] K. K. Chintalapudi, “i-MAC - a MAC that learns”, in *Proc. ACM IPSN’10*, Stockholm, Sweden, Apr. 2010.

[11] D. S. D. Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing”, in *Proc. ACM MobiCom’03*, San Diego, CA, USA, Sep. 2003.

[12] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler, “sMAP: a Simple Measurement and Actuation Profile for Physical Information”, in *Proc. ACM SenSys’10*, Zurich, Switzerland, Nov. 2010.

[13] R. Draves, J. Padhye, and B. Zill, “Routing in multi-radio, multi-hop wireless mesh networks”, in *Proc. ACM MobiCom’04*, Sep. 2004.

[14] V. L. Erickson and A. E. Cerpa, “Occupancy based demand response HVAC control strategy”, in *Proc. ACM BuildSys’10*, Zurich, Switzerland, Nov. 2010.

[15] K. Farkas, T. Hossmann, F. Legendre, B. Plattner, and S. K. Das, “Link quality prediction in mesh networks”, in *Computer Communications*, vol. 31, Iss. 8, pg. 1497 - 1512, 2008.

[16] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse, “The hitchhiker’s guide to successful residential sensing deployments”, in *Proc. ACM SenSys’11*, Seattle,USA, Nov.2011.

[17] J. Hsu, P. Mohan, X. Jiang, J. Ortiz, S. Shanker, S. D. Haggerty, and D. Culler, “HBCI: Human-Building-Computer Interaction”, in *Proc. ACM BuildSys’10*, Zurich, Switzerland, Nov. 2010.

[18] X. Jiang, M. V. Ly, J. Taneja, P. Dutta, and D. Culler, “Experiences with A High-Fidelity Wireless Building Energy Auditing Network”, in *Proc. ACM SenSys’09*, Berkeley, CA, USA, Nov. 2009.

[19] X. Jiang, S. D. haggerty, and D. Culler, “sMAP: simple monitoring and actuation profile”, in *Proc. ACM IPSN’10*, Stockholm, Sweden, 2010.

[20] D. S. Johnson, “Near-optical bin-packing algorithms”, in *Doctoral Thesis*, MIT Press, Cambridge, 1973.

[21] K. Kim and K. G. Shin, “On accurate measurement of link quality in multi-hop wireless mesh networks”, in *Proc. ACM MobiCom’06*, Los Angeles, CA, USA, Sep. 2006.

[22] J. Kleissl and Y. Agarwal, "Cyber-physical energy systems: focus on smart buildings", in *Proc. ACM DAC'10*, Anaheim, CA, USA, Jun. 2010.

[23] S. Lin, G. Zhou, Y. Wu, K. Whitehouse, J. Stankovic, and T. He, "Achieving stable network performance for wireless sensor networks", in *Proc. ACM SenSys'08*, Raleigh, NC, USA, Nov. 2008.

[24] H. Liu, J. Li, Z. Xie, S. Lin, K. Whitehouse, J. A. Stankovic, and D. Siu, "Automatic and robust breadcrumb system deployment for indoor firefighter applications", in *Proc. ACM MobiSys'10*, San Francisco, California, USA, Jun. 2010.

[25] Technical Report, Demo and Open source software, available at http://www.comp.polyu.edu.hk/~csdwang/Projects/wBACnet.htm, 2012.

[26] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse, "The smart thermostat: using occupancy sensors to save energy in homes", in *Proc. ACM SenSys'10*, Zurich, Switzerland, Nov. 2010.

[27] A. Marchiori and Q. Han, "Distributed wireless control for building energy management", in *Proc. ACM BuildSys'10*, Zurich, Switzerland, Nov. 2010.

[28] J. Martocci, P. D. Mil, and W. Vermeylen,"Building Automation Routing Requirements in Low-Power and Lossy Networks", in *RFC 5867*, 2010.

[29] G. R. Newsham and B. J. Birt, "Building-level occupancy data to improve ARIMA-based electricity use forecasts", in *Proc. ACM BuildSys'10*, Zurich, Switzerland, Nov. 2010.

[30] L. Parolini, N. Tolia, B. Sinopoli, and B. H. Krogh, "A cyber-physical systems approach to energy management in data centers", in *Proc. ACM ICCPS'10*, Stockholm, Sweden, Apr. 2010.

[31] A. Rice, S. Hay, and D. R. Cook, "A limited-data model of building energy consumption", in *Proc. ACM BuildSys'10*, Zurich, Switzerland, Nov. 2010.

[32] M. Senel, K. Chintalapudi, D. Lal, A. Keshavarzian, and E. J. Coyle, "A kalman flter based link quality estimation scheme for wireless sensor networks", in *Proc. IEEE GLOBECOM'07*, Washington,USA, Nov. 2007.

[33] K. Srinivasan and P. Levis, "RSSI is under appreciated", in *Proc. IEEE EmNets'06*, Cambridge, MA, USA, May. 2006.

[34] M. Vutukuru, H. Balakrishnan, and K. Jamieson, "Cross-Layer wireless bit rate adaptation", in *Proc. ACM SIGCOMM'09*, Spain, 2009.

[35] Q. Wang, X. Liu, W. Chen, L. Sha, and M. Caccamo, "Building robust wireless LAN for industrial control with the DSSS-CDMA cell phone network Paradigm", in *IEEE Transactions on Mobile Computing*, Vol. 6, Iss. 6, pg. 706 - 719, 2007.

[36] Y. Wang, M. Martonosi, and L. Peh, "Predicting link quality using supervised learning in wireless sensor networks", in *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, Vol. 11, Iss. 3, pg. 71- 83, 2007.

[37] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network", in *Proc. USENIX OSDI'06*, Seattle, Washington, USA, Nov. 2006.

[38] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghvan, "Robust rate adaptation for 802.11 wireless networks", in *Proc. ACM MobiCom'06*, Los Angeles, CA, USA, Sep. 2006.

[39] BACnet Unplugged ZigBee and BACnet Connect, ASHRAE Journal 2008, http://www.bacnet.org/Bibliography/AJ-6-2008.pdf.

[40] http://atom.research.microsoft.com/sensormap/.

[41] http://arduino.cc/en/Main/ArduinoBoardMega2560

[42] http://www.deltacontrols.com/solutions-products/products