Article

A Polling-Based Transmission Scheme Using a Network Traffic Uniformity Metric for Industrial IoT Applications

Yuichi Igarashi ^{1,*} , Ryo Nakano ^{1,†} and Naoki Wakamiya ^{2,†}

¹ Center for Technology Innovation, Research & Development Group, Hitachi, Ltd., 1-280, Higashi-koigakubo Kokubunji-shi, Tokyo 185-8601, Japan; ryo.nakano.xd@hitachi.com

² Graduate School of Information Science and Technology, Osaka University, Osaka 565-0871, Japan; wakamiya@ist.osaka-u.ac.jp

* Correspondence: yuichi.igarashi.hb@hitachi.com; Tel.: +81-42-323-1111

† These authors contributed equally to this work.

Received: 30 November 2018; Accepted: 30 December 2018; Published: 7 January 2019

Abstract: The Industrial Internet of Things (IIoT) applications are required to provide precise measurement functions as feedback for controlling devices. The applications traditionally use polling-based communication protocols. However, in polling-based communication over current industrial wireless network protocols such as ISA100.11a, WirelessHART have difficulty in realizing both scheduled periodic data collection at high success ratio and unpredictable on-demand communications with short latency. In this paper, a polling-based transmission scheme using a network traffic uniformity metric is proposed for IIoT applications. In the proposed scheme, a center node controls the transmission timing of all polling-based communication in accordance with a schedule that is determined by a Genetic Algorithm. Communication of both periodic and unpredictable on-demand data collection are uniformly assigned to solve the above difficulties in the schedule. Simulation results show that network traffic is generated uniformly and a center node can collect periodic data from nodes at high success ratio. The average success probability of periodical data collection is 97.4% and the lowest probability is 95.2%.

Keywords: the industrial IoT; polling-based communication; scheduling

1. Introduction

The Internet of Things (IoT) is a network of intelligent computers, sensors, and objects that collect, share, and analyze huge amounts of data [1–3]. Industrial applications and social infrastructure applications such as PA (Process Automation), DA (Distribution Automation), AMI (Advanced Metering Infrastructure) and predictive maintenance recently use IoT technologies to improve production efficiency, ensure optimal resource consumption, and operate systems more efficiently and economically [4–6].

In DAs, a SCADA (Supervisory Control And Data Acquisition) system performs operations like bus voltage control, bus load balancing, circulating current control, overload control, transformer fault protection, and bus fault protection [7]. In this system, measurements are made by end devices called as Remote Terminal Units in the real field and then data are transferred to a central device called as a SCADA center so that complete process or manufacturing information can be provided remotely. A SCADA center acquires data through a process called polling to simplify communications between a central device and end devices.

In a typical AMI, DLMS/COSEM (Device Language Message Specification/Companion Specification for Energy Metering) [8] is used at the application layer. It is responsible for polling smart

meters connected to the network and for sending the retrieved data to the management system from smart meters.

Both AMI and DA use polling schemes to exchange messages between a central device and end devices. In addition, a real-time system such as DA or PA reacts or responds to the applications within a fixed amount of time in order to avoid system failures. Both processing and reacting should be done within a pre-determined deadline that includes communication latency. Traditional PA systems use data collection protocols like Modbus that were originally designed for use with low-bandwidth wired communications such as serial communications and tolerant of long communication latency [9]. Traditional dedicated wired communication protocols for industrial applications have been well designed to meet requirements regarding the deadline.

The IoT needs connectivity for devices and Industrial Wireless Sensor Networks (IWSNs) [10,11] have been emerging as a new means of wireless communications for the IoT. Recently, several protocols for IWSNs such as WirelessHART [12], ISA100.11a [13], and IEEE802.15.4e [14], have been developed and standardized in order to increase the communication reliability in a wireless network with frequent packet loss and big latency under the heavy network utilization. These standards are time-synchronized and assigned communication timings when devices join the wireless network. Both application data packets and network control packets are transmitted according to the assignment.

When polling-based protocols for industrial applications are used over IWSNs, the number of retransmissions of polling queries over IWSN layer should be considered in order to maintain a high reliability of industrial systems. At the same time, their timing of the retransmission should be carefully considered in order to keep the deadline.

Moreover, industrial applications as described above basically gather heterogeneous periodic information from field devices. The transmission frequency of periodic information collection is normally static. At the same time, they also require unpredictable communication for gathering on-demand data or operating devices by a center device within specific end-to-end deadlines [15–17]. In general, traditional industrial communication protocols over wired communications take into consideration maintaining the deadline even if the unpredictable communication occurs in the network. For frequent data collection in DA, the wired network load can be as much as ten times less than upper bound [18].

To achieve both heterogeneous periodic data collection with high success ratio and unpredictable on-demand communication within a deadline over IWSNs, we propose a data traffic control scheme for polling-based communication in IWSNs and a scheduler which uniformly distributes network load over slots. Our proposed scheme incorporates multiple heterogeneous periodic data collection schedules in a single schedule. The proposed approach aims to balance network traffic of multiple applications as well as to provide high success data collection. In addition, the schedule gives all nodes fair opportunities to receive a polling query from a central node at a certain interval. It is important to generate a polling query to a node fairly, because the available bandwidth for downward traffic is less than that for upward. This mechanism also enables a center node to generate an unpredictable on-demand request to a node and receive a response within a deadline.

The contribution of this paper is to propose a data traffic control scheme for polling-based communications and verify its performance from viewpoints of end-to-end communication success probability and balanced slot utilization. None of the conventional technologies for IWSNs focus on the problem that an unbalanced network bandwidth between uplink and downlink causes unexpected big latency of unpredictable on-demand communication or decrease of success probability for periodic data collection. Standard IWSN protocols like ISA100.11a typically have a scheduler for allocating network resources such as timeslots to all nodes. Since a network manager of IWSN protocols gathers information about network condition from all nodes, IWSN protocols normally require a large available bandwidth for upward traffic. In addition, the scheduler has to deliver the information to all nodes whenever a new node joins the network or a network topology changes. Therefore, it is difficult to support a polling-based unpredictable on-demand communication within deadline. In contrast,

since our scheme only determines when to generate and transmit a packet at a root node for periodic data collection and unpredictable on-demand data collection. It does not need to adjust the schedule because the number of nodes does not change.

The rest of the paper is structured as follows. We first describe assumptions and challenges in Section 2, and Section 3 presents an overview of related work. In Section 4 we propose the data traffic control scheme. Then, we evaluate uniformity and success ratio of collecting periodic data in Section 7. In Section 8, we discuss possibility of providing schedules when an application data traffic is high. Finally, Section 9 concludes the paper and describes future work.

2. Assumptions of Our Target System and Challenge

In this section, we provide assumptions of our target system at first, and describe challenges of this paper.

2.1. Assumptions

- **IIoT application features.** As noted above, our target IoT applications are AMI, DA, PA, and so on. These typical industrial applications normally collect field data from end devices periodically within pre-determined deadlines that include communication latency and internal processing time on both a central node and an end device. In this paper, we assume that the end devices periodically generate data in intervals, and a deadline that the central node should get data from an end device is equal to the next time that an end device generates data within the interval.
- **polling-based communication.** Our target IoT applications normally use polling-based communication to keep a sequence of processes or simplify network management. In general, there are two different polling schedulings, called monocycle and multicycle polling scheduling.
- **Monocycle polling scheduling.** Monocycle polling scheduling is the most common and the simplest. In monocycle polling scheduling, a central node uses a single cycle for the polling required for all devices belong to a single group. Figure 1 shows an example of a monocycle polling scheduling. In this case, the deadline (cycle) is T_1 and three nodes ($n_{1,1}$, $n_{1,2}$, $n_{1,3}$) belong to the group.

Figure 1. An example of polling schedule in a monocycle polling scheduling.

- **Slot assignment for reliable polling communication.** A central node transmits the first queries at t_1, t_2, t_3 to the nodes respectively. Retransmission timings are also allocated in case of communication failure. For example, t_4 and t_7 are timing for retries of the first query to $n_{1,1}$. The number of the retries R_j for node j depends on communication success rate, for example PER (Packet Error Rate) as shown in Equation (1);

$$1 - P_{PER_j}^{R_j} \geq Th_{success}, \quad (1)$$

where P_{PERj} is an end-to-end PER between a central node and node j and $Th_{success}$ as one of system requirements is probability of collecting data from any nodes in a system.

- Multicycle polling scheduling.** Multicycle polling scheduling is a set of heterogeneous monocycle polling schedulings. Figure 2 shows an example of multicycle polling patterns. In this example, three applications in a system have different cycles that are denoted by T_1 , T_2 , and T_3 , respectively. The central node transmits the first queries and their retries in each cycle as well as a monocycle polling scheduling. In Figure 2, lower brightness areas represent some consecutive slots of first queries and higher ones do some consecutive slots of their retries. Besides, white areas stand for open slots for unpredictable on-demand request. The width of the areas depends on the schedules. For example, the first lower brightness area of G_1 includes 3 slots and the second one includes 5 slots.

Figure 2. An example of multicycle polling patterns.

- Wireless communication for IWSNs.** To increase the reliability of wireless networks for industrial applications, Time Slotted Channel Hopping (TSCH) based IWSN protocols such as WirelessHART, ISA100.11a, and IEEE802.15.4e have been developed. They are time-synchronized and assigned communication timings for transmitting packets by a central network manager in order to decrease both interference within the wireless network and interference from other wireless networks using the same radio channels. The communication timing for both downlink traffic from a central node to end devices and uplink traffic from end devices to a central node are assigned. These bandwidths for downward and upward are normally fixed. In addition, the available bandwidth for upward traffic is bigger than that for downward.
- Imbalanced bandwidth between upstream and downstream of IWSNs.** Generally speaking, bandwidth of standard IWSNs for the downstream from a center node to end devices is quite less than that for upstream because main traffic for IWSNs is autonomously transmitting sensor data from devices. For example, SmartMesh IP [19] that is based on the 6LoWPAN and IEEE802.15.4e standards provides only one timeslot for downstream traffic from a center node to a device node in every 2 s by default settings, contrary to several timeslots for upstream from the nodes to a center node.
- Traffic patterns.** Industrial applications basically gather heterogeneous periodic information from field devices. At the same time, they also require on-demand data communication for additional data collection and operation of devices by a remote server within specific end-to-end deadlines [15]. When the unpredictable on-demand data communication occurs, the unpredictable traffic may be given high priority to over periodical scheduled communication and a central node transmits a polling packet for the on-demand request. Consequently, a periodical polling request from a central node to a node is dropped by the unpredictable on-demand data communication. If an open slot is fortunately assigned when the on-demand request occurs, a central node can get data from a node by an on-demand request without any interruptions of periodical data collection. Therefore, it is better for IWSNs to generate network traffic load for periodical data collection uniformly and also keep available bandwidth for unpredictable traffic at any time.

- Traffic control scheme for multiple wireless communications.** Industrial applications require similar requirements for IWSNs, but applications in different domains may use different IWSN protocols. For example, AMI often uses IEEE802.15.4e and PA uses WirelessHART or ISA100.11a. We propose a data traffic control scheme for periodical data collection over IWSNs and our proposal does not completely depend on any specific protocols. Our proposed scheme is a technology that lies between applications and an interface of IWSNs, namely IP (network layer) or MAC (datalink layer) as shown in Figure 3. A center node has the data traffic control function in order to manage all polling traffic.

Figure 3. Position of our data traffic control scheme in function layers.

2.2. Challenge

As described above, it is important to realize both collecting multiple periodic data from sensors within deadlines and transmitting unpredictable on-demand packets. Transmitting opportunity for downstream of IWSNs such as ISA100.11a or WirelessHART, is normally lower than that for upstream, even though industrial applications use polling-based protocols that need equal opportunities for downward packets and upward packets. Therefore, one of our challenge is to provide a communication method that controls both downward and upward network load in IWSNs. The communication scheme should also ensure sufficient responsibility of unpredictable requests and stability of periodic data collection. The second challenge of this paper is to provide a schedule which a central node normally generates uniform network load for periodic data collection by a polling-based communication protocol, in order to enable applications to transmit unpredictable packets effectively using the idle time of the IWSNs.

3. Related Work

In communication systems and wireless sensor network systems, many studies have been proposed to solve scheduling problems [20–25].

In a FieldBus environment, scheduling problems of dynamic or static collecting information flow are to be found in [20,21]. Algorithms capable of solving this problem have to be able to calculate the transmission sequence for all groups. The solution is firstly to decide the primary cycle among multiple cycles. The primary cycle is the shortest period. Then, if there are two polling tasks of different groups at the same time, the group whose polling cycle is T_i always has higher priority than group whose polling cycle is T_j , where $T_i \leq T_j$. In the papers, authors assume that deadline coincides with the polling cycle, and scheduling problem is to ensure that at least one transmission for each group will occur at least once in the cycle. They do not address features of wireless communication.

In [22], the authors propose TDMA link scheduling algorithms for the purpose of maximizing network throughput. In [23], an end-to-end real-time transmission scheduling over the wirelessHART networks is proposed. Both [22,23] are focus on making conflict-free link level schedules. According

to these schedules, end devices can autonomously transmit periodical sensor data to a center node within a deadline according to the schedule but this communication is not polling-based.

In [24], the authors show through analysis and experiments that conflict-free query scheduling has an inherent tradeoff between network throughput and latency. They propose real-time scheduling algorithms for prioritized conflict-free transmission scheduling in order to aim to balance network throughput and latency.

In [25], a new priority-based parallel schedule polling MAC protocol in WSNs combines polling orders with access policies to realize the priority-based scheme and reduce the overhead time through parallel schedule. Sensor nodes can be classified in different clusters and this MAC protocol coordinates all clusters and gives a chance to send data in order. To do this, the node to which a central node wants to transmit a priority packet is given the transmission chance first, and other nodes remain in a sleeping status to save energy.

As will be described in Section 6, we adopt a GA (genetic algorithm) as a heuristic to derive an optimal schedule [26]. GA is one of probabilistic search algorithms and optimization techniques based on the mechanisms of natural selection and evolution. It has been applied to optimization of complex network management problems such as network load balancing management [27] and routing [28]. Although other heuristic algorithms such as simulated annealing, PSO (particle swarm optimization), or even machine learning can also be adopted, we consider a GA-based algorithm in this paper, which could achieve a reasonable and feasible schedule with practically short computation time on an off-the-shelf computer.

Our work is different from related work in the following aspects. As shown in Table 1, other works focus on how to schedule transmitting queries without conflicts in order to minimize network latency, optimize network throughput, or save energy. However, none of them focuses on the problem that unbalanced network bandwidth between uplink and downlink causes unexpected big latency of unpredictable on-demand communication or decrease of success probability for periodic data collection. We firstly propose a polling-based data traffic control scheme which is available to any IWSN protocols that assign unequal bandwidth to upstream and downstream communication. In addition, we show a scheduling using a network traffic uniformity metric for IWSNs to realize both scheduled periodic data collection at high success ratio and unpredictable on-demand communications with short latency.

Table 1. Summary of related work in comparison with our work.

Purpose of Scheduling	Work	Network Type	Communication Technique	Scheduling Type	Unbalanced Bandwidth
Real-time data collection	[20]	Wired	Polling	Conflict-free	Nothing
Multicycle periodic data collection	[21]	Wired	Polling	Conflict-free	Nothing
Optimizing network throughput	[22]	Wireless	Push	Conflict-free	No consideration
Minimizing latency	[23]	Wireless	Push	Conflict-free	No consideration
Minimizing latency and optimizing network throughput	[24]	Wireless	Polling	Prioritized conflict-free	No consideration
Saving energy	[25]	Wireless	Polling	Prioritized	No consideration

4. Data Traffic Control Scheme for Periodic Data Collection

In this section, we provide terminologies at first. Then, we present an outline of how our data traffic control scheme for periodic data collection works. We define two kinds of frames and Table 2 summarizes terminologies and notations.

- **QueueFrame.** The first kind is the **QueueFrame (QF)**, which consists of N_{all} slots whose length is Δt_s as shown in Figure 4. Let N_i be the number of nodes which belong to group i and Let τ_g

be the number of multicycle polling groups in an IWSN. Some nodes may belong to multiple different groups. Then N_{all} in an IWSN is as bellows;

$$N_{all} = \sum_{i=1}^{\tau_g} N_i. \quad (2)$$

We assume that a central node collects data from all nodes at multiple periodic cycle as shown in Figure 2. The length of a QF is $\Delta T_{QF} (= \Delta t_s \times N_{all})$. In our proposal, a central node generates at most N_{all} queries to all nodes in a QF, so that our data collection scheduler gives fair opportunities for all nodes even in a short time span (ΔT_{QF}). We consider the bandwidth of downward traffic to define polling cycle Δt_s . For example, a our preliminary experiment showed downward traffic for SmartMesh IP that is based on the 6LoWPAN and IEEE802.15.4e standards is assigned every about 2 seconds by the default setting. Δt_s should be longer than the frequency for pre-assigned downlink traffic of IWSN protocols. In this paper, we set Δt_s to 3s for simulation setting. The order of queries to nodes are decided by a data collection scheduler. One of the simplest ways to do this is arranging nodes in ascending order of a unique id.

Table 2. Notation and description.

Notation	Description
N_i	N_i is the number of nodes in group i .
τ_g	τ_g is the number of multicycle polling groups in an IWSN.
N_{all}	N_{all} is the number of total nodes which belong to all groups.
QF	QueueFrame (QF) is the minimum frame which consists of N_{all} slots.
ΔT_{QF}	ΔT_{QF} is The length of a QF. A central node can transmit at most N_{all} polling requests in a QF.
Δt_s	Δt_s is the length of a slot. A central node can transmit a polling request in a slot.
T_i	T_i is a collection data interval of group i .
SF	ScheduleFrame is the minimum unit of multicycle polling schedule.
T_M	T_M is a multicycle polling interval.
N_{QF}	N_{QF} is the number of QFs in a SF.

Figure 4. A composition of a QueueFrame (QF).

- **ScheduleFrame.** The second kind is the **ScheduleFrame (SF)**, which consists of QFs, as shown in Figure 5. Let T_M be the least common multiple of collection cycle of all groups as shown in Equation (3). In the interval T_M , the SF consists of $N_{QF}(= T_M \div \Delta T_{QF})$ QFs. Therefore, a SF is the minimum unit of multicyle polling schedule of N_{all} nodes.

$$T_M = \text{lcm}(T_i), \text{ for } i = 1, 2, \dots, \tau_g. \quad (3)$$

Our data collection scheduler decides all polling orders in order to make network traffic of all applications uniform during the T_M and our data traffic control scheme generates polling queries every Δt_s according to the schedule.

Figure 5. A composition of a ScheduleFrame (SF).

Below, we give an outline of how our proposal works over any IWSN protocols that assign unequal bandwidth to upstream and downstream communication. In our scheme, we define two kinds of frames to decide all polling orders in both a short time span and a long time span.

Our scheme incorporates multiple groups in a single group as shown in Figure 4. A central node transmits at most N_{all} queries every Δt_s according to a sequence of nodes in a QF. This polling behavior is as same as a monocycle polling scheduling. We consider that a central node controls transmitting timing of a polling query so that down stream traffic from a central node is less than pre-assigned downlink traffic of an IWSN protocol.

Then, a scheduler provides a SF that is a multicyle polling schedule during the least common multiple of collection cycle for all groups as shown in Figure 5. In a SF, each QF may assign a different sequence of all nodes. For example, a sequence of all nodes in QF_1 may be assigned in ascending order and one in QF_2 may be assigned in descending order.

5. Problem Formulation of Data Collection Scheduler

A polling cycle of a group T_i consists of K_i QFs where

$$K_i = T_i / \Delta T_{QF} \quad (4)$$

$$= T_i / (\Delta t_s \times \sum_{k=1}^{\tau_g} N_k) \quad (5)$$

$$= T_i / (\Delta t_s \times N_{all}). \quad (6)$$

Let $C_{i,j}$ be a total number of combinations of selecting QFs which any node j in any group i uses for data transmit and its retries $R_{i,j}$ is

$$C_{i,j} = K_i \times \sum_{m=1}^{K_i} (K_i - m) C_{\min(K_i - m, R_{i,j})} \quad (7)$$

where $\min(K_i - m, R_{i,j})$ is the smallest number of the two arguments.

Total number of combinations of selecting QFs which all nodes in any group i use is

$$C_{i,all} = \prod_{j=1}^{N_i} C_{i,j}. \quad (8)$$

Then, total number of combinations of selecting QFs for all groups and all nodes $C_{all,all}$ is

$$C_{all,all} = \prod_{i=1}^{\tau_g} (C_{i,all})^{\frac{T_g}{T_i}}. \quad (9)$$

The number $C_{all,all}$ depends on the number of nodes N_{all} in an IWSN, the number of collection cycles τ_g , communication quality, i.e., the number of total communication retries $R_{i,j}$.

On the other hand, maximum number of polling queries for periodic data collection of group i during a polling cycle T_i is $\sum_{j=1}^{N_i} (1 + R_{i,j})$. Let $Q_{i,x}$ be total polling queries of the x th QF of a group i during T_i . When the queries are uniformly generated during T_i , $Q_{i,x}$ is approximated by the Equation (10).

$$Q_{i,x} \approx \frac{1}{K_i} \sum_{j=1}^{N_i} (1 + R_{i,j}) \quad (10)$$

The total polling queries of x th QF of all groups is

$$Q_{all,x} = \sum_{i=1}^{\tau_g} Q_{i,x} \quad (11)$$

$$\approx \sum_{i=1}^{\tau_g} \frac{1}{K_i} \sum_{j=1}^{N_i} (1 + R_{i,j}). \quad (12)$$

Then, we find an optimal schedule from $C_{all,all}$ patterns of schedules in order to generate network traffic load uniformly for periodical data collection. The standard deviation of Equation (12) $\sigma(Q_{all,x})$ is a measure of how uniform polling queries are generated during T_M . Our objective function is

$$\alpha = \min\{\sigma(Q_{all,x})\}, \text{ where } 1 \leq x \leq \frac{T_M}{\Delta t_s \times N_{all}}. \quad (13)$$

6. GA-Based Slot Assignment Algorithm

Our polling-based data traffic control scheme decides the order of all queries to all nodes according to a schedule that ensures sufficient responsibility of unpredictable requests and stability of periodic data collection. In other words, the slots of queries for periodical data collection and slots for unpredictable on-demand should be uniformly assigned. As we described in Section 3, Genetic Algorithm (GA) is classified one of heuristic functions which can find an optimum solution for above our problem, so that we propose a GA-based slot assignment algorithm.

The process of our GA-based slot assignment algorithm includes generating initial population, selection, mutation, crossover, repair, and evaluation as shown in Figure 6. We describe the specific process of the flow as follows.

6.1. Encoding

Our data collection schedule includes two problems for transmitting polling queries. One is how a central node generates polling queries for heterogeneous multiple periodic data collection with high success ratio. Another is how a central node can transmit polling queries for unpredictable on-demand communication within a deadline over IWSNs. We encode a sequence of nodes and kinds of packet (e.g., periodic data collection, retransmissions for periodic data collection, and unpredictable on-demand communication). In our proposal, each individual has two chromosomes. We describe

coding chromosome structure at first and then, we present how to generate the initial population as shown in Figure 6.

Figure 6. Flowchart of our Genetic Algorithm (GA)-based slot assignment algorithm.

6.1.1. Chromosome Structure

An example coding chromosome representing a schedule is shown in Figure 7. Each individual has two chromosomes corresponding to a sequence of node numbers and a sequence of slot types. Both chromosomes have N_{QF} QFs which consist of N_{all} slots as shown in Figure 4, respectively.

As the first chromosome, a sequence of node numbers represents transmission orders from a central node to nodes. In this paper, we substitute a simple number like 1, 2, 3, ... as a node number for (Group number, node number) in order to simplify the notation.

As the second chromosome, a sequence of slot types represents kinds of packets that a central node sends to a node at the slot. We define three slot types in this paper. First type “1” represents that a central node transmits an original polling request packet for periodic data collection. Second one “2” represents that a central node sends the retry packet of the original packet to an end node. Final one “3” means that a central node can transmit a query for an unpredictable on-demand collection to any node at the slot.

For example, Figure 7 shows that a central node transmits a request packet to collect periodical application data from node 1 at slot 1, and a retry request packet to node 1 at slot 8 if the central node did not receive the data corresponding to the previous request to node 1. If the central node received a reply from node 1 by slot 8, the central node does not transmit the retry packet to node 1 at slot 8. In addition, a central node may transmit a polling query to any node for an unpredictable on-demand

collection at the slot 4 but a central node never transmits a request packet to node 4 for a periodic data collection at the slot.

Slot number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...			
Node number	1	2	3	4	5	6	2	1	3	5	6	4	3	1	4	6	...			
Slot type	1	1	1	3	3	1	2	2	3	1	3	1	2	2	3	1	...			
	QF ₁						QF ₂						QF ₃							

Figure 7. An example of chromosome encoding.

6.1.2. Initialization

Each Individual of GA population has two chromosomes. At the beginning of the initialization, a data collection scheduler generates N_{QF} QFs such that all nodes in an IWSN are collated at random in order to encode a chromosome that represents an order of node IDs. We think that the random selection provides a search diversity. Our simulation experiment showed that it did not have a big impact on the performance. After that, a chromosome of slot type is assigned. For the slot type chromosome initialization, first of all, the first queries of all nodes to collect data while a periodic cycle are assigned. The assignment of slot type chromosome denotes 1 in the chromosome as shown in Figure 7. Then, the timing of transmitting queries for retries of the collection while a periodic cycle are assigned. The assignment denotes 2 in the slot type chromosome. The number of retries depends on a path quality (end-to-end PER) between a node and a central node. A central node gets the path quality from a network manager in an IWSN, because a network manager perceives network conditions. If a path quality between a node and a central node is 80%, a central node will require at least a retry. Finally, the slot type of other slots that are assigned to the node are 3. “3” means that unpredictable on-demand data traffic from central nodes can be generated at the slot.

6.2. Crossover and Mutation

Figure 8 shows an example of mutation, crossover and repair. At the crossover process, we select two individuals randomly as parents and pick a QF up from the node sequence chromosome of the parent, respectively. The QF of the first parent is exchanged with the QF of the second parent in order to generate two new children individuals. Let N_{indiv} be the number of individuals that are generated at the initial phase. Then we have at most $N_{indiv} + 2$ kinds of node sequence in T_M . In other words, there are at most $N_{indiv} + 2$ schedules at the moment.

Figure 8. An example of our mutation and selection.

The mutation operation also provides a search diversity by avoiding the local maximums. Mutation operation alters one individual. In our proposal, the alteration of one individual in each

mutation operation is considered. A new individual is produced in the same way of the initialization phase. The new one is replaced to an individual of GA population. Mutation operation does not have to be executed before every crossover operation as shown in Figure 6.

6.3. Repair

Because each slot type chromosome corresponds to each node sequence chromosome, slot type chromosome of the first parent are also exchanged with one of the second one. However, the exchanges of slot type chromosome is more complicated than the one of node sequence chromosome because “1” (a query for collecting data) appears once during a data collection cycle T_i and “2” (a query for retries) should be assigned after the “1”. As shown in Figure 9, slot number 1 and 2 of the slot type chromosome of a child 1 should be repaired after the exchange. In our proposal, the repair follows the 9 cases as shown in Figure 10;

- **Case 1-1:**
The child chromosome does not need to be repaired at all.
- **Case 1-2:**
After the crossover operation, a new child chromosomes has no “1” during the T_i . Then, an anterior “3” in the collection cycle should be changed to “1”, and one of the “2” should be changed to “3” if there are more “2” than required. If there is no “3” before the exchanged slot number, the child chromosome does not need to be repaired at all.
- **Case 1-3:**
After the crossover operation, a new child chromosome has no “1” during the T_i . Then, an anterior “3” in the collection cycle should be changed to “1”. If there are no “3” before the exchanged slot number, the child chromosome does not need to be repaired at all.
- **Case 2-1:**
After this crossover operation, a new child chromosome has additional “1” during the T_i . Then, an anterior “1” and “2” should be changed to “3”. Also, one of “3” after the new “1” should be “2” in the case that there are fewer “2” than required.
- **Case 2-2:**
The child chromosome does not need to be repaired at all.
- **Case 2-3:**
One of “3” after “1” changes to “2”.
- **Case 3-1:**
After the crossover operation, a new child chromosome has additional “1” during the T_i . Then, an anterior “1” and “2” should be changed to “3” or a rearward “1” is changed to “3”. Then, one of the “2” should be changed to “3” if there are more “2” than required.
- **Case 3-2:**
one of “2” changes to “3” in case there are more “2” than required.
- **Case 3-3:**
The child chromosome does not need to be repaired at all.

Figure 9. An example of the crossover process.

		Slot type after crossover operation		
		1	2	3
Slot type before crossover operation	1	Case 1-1	Case 1-2	Case 1-3
	2	Case 2-1	Case 2-2	Case 2-3
	3	Case 3-1	Case 3-2	Case 3-3

Figure 10. Cases of repairing genes after crossover slot type chromosomes.

6.4. Selection and Fitness Function

After the crossover, mutation, and repair operations, there are $N_{indiv} + 2$ individuals. The first N_{indiv} individuals, which have higher fitness values, are selected and transferred to the next generation. The fitness values of chromosomes are calculated from our fitness function (Equation (13)).

One of our objectives is to uniform the data traffic generation from a central node to nodes. In our proposal, each slot in a QF generates at most one query to collect data and the slot type chromosome represents how many queries are generated in a QF. At the same time, our scheduler tries to maximize communication success rate. When the fitness value is calculated, the communication success rate of all nodes are also calculated and checked. If the success rate of all nodes of a child chromosome does not satisfy our target probability, the individual is discarded.

Finally, when the fitness values of the best 5 slot type chromosomes are not changed after crossover operation, and the best fitness value is less than a target threshold value, we consider the GA operation is convergent. After the maximum iteration, we also consider that the GA operation will not be converged. At that time, we return to the initialization procedure. Our experiments showed that after 100 experiments, the average execution time of slot assignment for 10 nodes was 1544 milliseconds. After the convergence, the best slot type chromosome that is ranked in the top 5 and that is expected the highest average communication success rate of all nodes, is selected as the best schedule.

7. Evaluation

7.1. Simulation Settings

We implemented our GA-based slot assignment algorithm in C using the gcc compiler and ran all experiments on Mac OS 10.12.6 with an 2.7 GHz Intel Core i5 CPU with 16GB RAM. To evaluate

the performance impact of our data traffic control scheme, we performed a set of simulations with 10 end nodes placed statically and randomly in a square field. A central node was placed at the lower left-hand corner of the field, and LOADng that is a routing protocol for low power and lossy networks [29] was applied to create multi-hop routes from all nodes to a central node with a shortest-path metric. A network topology was fixed during a simulation. We also assume that packet loss among neighbors is caused by several factors such as propagation models, signal processing technology, transmitting power, antenna characteristics, and reception sensitivity, but except signal interference from other nodes. We then determined PER of a path quality at random for every node. As shown in Table 3, we set the range of PER to 0–20%. Although the link PER dynamically changes in reality, we assume it is stable and constant in this paper. Evaluation under dynamic environment is left as future work. After creating a network topology, our implementation uses the information and evaluates how a central node generates network traffic load for periodical data collection uniformly.

At first, we randomly divided 10 end nodes to three groups that have different data collection cycles $T_1 = 2$ min, $T_2 = 4$ min, $T_3 = 8$ min, respectively. According to our preliminary experiments, the average number of crossover operations for 100 network topologies was 1071 when our fitness values were converged. Then, we set the maximum number of crossover iteration to be 4000. Table 3 summarizes the details of the other parameter settings.

Table 3. Simulation conditions.

Item	Notation	Value
The number of nodes in an IWSN	N_{all}	10 nodes
Data collection cycle 1	T_1	2 min
Data collection cycle 2	T_2	4 min
Data collection cycle 3	T_3	8 min
The least common multiple of collection cycles	T_M	8 min
The number of slots in a QF	S_{QF}	10
Slot length	Δt_s	3 s
QF length	ΔT_{QF}	$30 (S_{QF} \times \Delta t_s)$ s
The number of QFs in a chromosomes	N_{QF}	$16 (T_M \div T_{QF})$
The number of slots in a chromosomes	S_{gene}	$160 (N_{QF} \times S_{QF})$
The number of individuals	N_{indiv}	20
The number of iteration of crossover operation	I_{max}	4000 times
Packet error rate	PER	0–20%
MAX retry count	R_{max}	2
Threshold of a path quality	Th_{path}	90%

7.2. Simulation Results

As described in Sections 5 and 6.4, the standard deviation of Equation (12) is measure of how uniform polling queries are generated during T_M and Equation (13) is our fitness function. Therefore, we calculated fitness values by Equation (13) for evaluation.

First of all, we confirmed that random selection at the initialization step did not have a big impact on the performance. To evaluate this, we created a network topology and then we generated 200 kinds of schedules for a scenario as shown in Table 3. All the results were converged. Then we show one of the detail results as bellow. The fitness values of the top 20 individuals after the GA-based slot assignment are plotted in Figure 11. As shown in Figure 11, minimum standard deviation at the initial random selection where the number of GA operation is 0, was 0.186 and the final one was 0.049. In this case, our proposal generates a schedule for periodic multicycle data collection after 2000 times GA crossover operations. In contrast, the standard deviation of the conventional scheme [25] is 0.49. From uniforming application traffic of periodic data collection point of view, our proposal is superior to the conventional scheme.

The mean value of all slot type during T_M is calculated by the Equation (14). SI_{ideal} is the ideal value when the polling schedule includes all slot assignment of data transmission for all nodes and

their retries. As shown in Figure 12, the best schedule in the 20 individuals assigns the ideal number of slots for data transmission and their retries.

$$SI_{ideal} = \frac{1}{N_{QF} \times N_{all}} \times \sum_{i=1}^{\tau_g} \left(\frac{T_M}{T_i} \times SI_i \right), \quad (14)$$

where the sum of slot type of all nodes which belong to group i is as follows;

$$SI_i = N_i + 2 \sum_{j=1}^{N_i} R_{i,j} + 3(K_i - N_i - \sum_{j=1}^{N_i} R_{i,j}).$$

Figure 11. Transition of standard deviation slot type value of the top 20 individuals, where $N_{all} = 10$.

Figure 12. Comparison between the mean value of slot type of all QueueFrames between the best chromosome and ideal value.

In addition, Figure 13 shows E2E communication “3”y of all nodes after the GA-based slot assignment, and average one of all nodes in each individual. The E2E communication “3”y is calculated by the left-side hand of Equation (1). The abscissa of the graph represents the ID number of individuals, and the ordinate represents the E2E communication “3”ies. The E2E communication “3”ies are over 90% as shown in Figure 13. The average success probability is 97.4% and the lowest one is 95.2% where the 5th individual is selected. The results meet our target requirements, as listed in Table 3.

Figure 13. E2E communication “3”y of all nodes in the top 20 individuals and average one of all nodes in each individual.

8. Discussion

8.1. Possibility of Finding Schedules When Application Data Traffic Is High

In general, the more nodes which belong to the shortest cycle group there are, the central device generates polling requests more frequently during T_M . In the case that most nodes belong to the shortest cycle group, a scheduler should determine a multicycle scheduling that generates polling transaction uniformly among less options than in the other cases. Then, we additionally evaluate our proposal in the worst case for 10 nodes that 8 nodes in T_1 , 1 node in T_2 and 1 node in T_3 respectively. Other conditions listed in Table 3 are the same as the above evaluation. Our proposal scheduler found schedules that meet our target requirements as shown in Figures 14 and 15. The result of the E2E communication “3”y depends not on the grouping ratio but on the network topology.

Figure 14. Transition of standard deviation slot type value of the top 20 individuals, where $N_1 = 8$, $N_2 = 1$, and $N_3 = 1$.

Figure 15. E2E communication "3"y of all nodes in the top 20 individuals and average one of all nodes in each individual, where $N_1 = 8$, $N_2 = 1$, and $N_3 = 1$.

8.2. Possibility for a Scalable IWSN

An IWSN may include over 100 end devices such as sensors or actuators [30], though we evaluated our proposal in Section 7 under a condition that an IWSN includes only 10 end nodes. To evaluate scalability of our proposal scheduler, we increased the number of nodes from 10 to 100. At the same time, we expand polling cycles because our proposal data traffic control scheme has a TDMA-like manner. To be more precise, we enlarged the length of a QF 10 times because the length of a QF is $\Delta T_{QF} (= \Delta t_s \times N_{all})$ as described in Section 4. Accordingly, we also enlarged the length of a SF 10 times. Simulation conditions are described in Table 4. Figure 16 shows that our proposal also provides a solution which has a scalability that is adaptive to a small scale network to a large scale one with the same uniformity.

Table 4. Simulation conditions.

Item	Notation	Value
The number of nodes	N_{all}	100 nodes
Data collection cycle 1	T_1	20 min
Data collection cycle 2	T_2	40 min
Data collection cycle 3	T_3	80 min
The least common multiple of collection cycles	T_M	80 min
The number of slots in a QF	S_{QF}	100
Slot length	Δt_s	3 s
QF length	ΔT_{QF}	$300 (S_{QF} \times \Delta t_s)$ s
The number of QFs in a chromosomes	N_{QF}	$160 (T_M \div T_{QF})$
The number of slots in a chromosomes	S_{gene}	$1600 (N_{QF} \times S_{QF})$
The number of individuals	N_{indiv}	20
MAX iteration of crossover operation	I_{max}	20,000 times
Packet error rate	PER	0–20%
Max retry count	R_{max}	2
Threshold of a path quality	Th_{path}	90%

Figure 16. Transition of standard deviation slot type value of the top 20 individuals, where $N_{all} = 100$.

8.3. Power Consumption

In IIOT systems, many devices such as sensors are powered from batteries. From power consumption point of view, it is important to save energy while devices communicate with others. Due to wireless radio interferences from other systems, PER may increase in IWSNs. Though standard IWSN protocols such as ISA100.11a and WirelessHART increase the reliability of wireless networks by TSCH mechanism, our proposal scheme increases retry queries in harmful network conditions in order to collect data at high success ratio when PER increases. As we described in Section 7, we assume it is stable and constant in this paper. Detailed evaluation is left as future work.

9. Conclusions and Future Work

This paper introduced a data traffic control scheme over IWSNs for polling-based data collection from multiple IIoT applications and its data collection scheduling. Our algorithm enables the uniform generation of network traffic load for periodical data collection. Polling request packets from a central node to end nodes can be uniform even if the polling cycles are multiple. This is achieved through a data traffic control scheme that generates a polling query at a fixed interval according to a schedule which is decided to make the occurrence of transmitting a polling request for periodical data collection, its retry in case of wireless communication failure or an interruption of higher priority request, and non operation by a GA-based algorithm. At the same time, our scheduler can give maximum opportunities for retransmitting polling requests and data collection ratio of periodic data packets achieves higher than 90%. We adopted a GA based algorithm to create a schedule, but our data traffic control scheme, of course, allows an algorithm that is not GA-based to decide a schedule.

In this paper, we did not address dynamic adaptation of our scheme to handle dynamic or unexpected changes in application and system requirements. For example, some applications will be installed after the deployment phase. In this case, a scheduler should modify a current schedule. Updating a whole schedule is one of the simplest solutions, but the applications should wait to execute this for at most T_M . Moreover, in our scenario, all nodes reply a response packet corresponding to a polling request, but an application such as error log monitoring on an end device may transmit more information (including system log data) than that which can be delivered in a reply packet. Therefore, in order to make the data traffic of all applications uniform, a scheduler should consider both uplink traffic and downlink traffic at the schedule creating phase. We plan to tackle these issues as future work.

Author Contributions: Conceptualization, Y.I., R.N. and N.W.; methodology, Y.I., R.N. and N.W.; software, Y.I.; validation, Y.I., R.N. and N.W.; formal analysis, Y.I.; investigation, Y.I.; resources, Y.I.; data curation,

Y.I.; writing—original draft preparation, Y.I.; writing—review and editing, N.W. and R.N.; visualization, Y.I.; supervision, Y.I.; project administration, Y.I.; funding acquisition, Y.I. and R.N.

Funding: This research received no external funding.

Acknowledgments: The authors extend their gratitude to all the reviewers for their constructive comments and suggestions, which have significantly improved this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
IIoT	Industrial IoT
IWSNs	Industrial Wireless Sensor Networks
PA	Process Automation
DA	Distribution Automation
AMI	Advanced Metering Infrastructure
SCADA	Supervisory Control And Data Acquisition
TSCH	Time Slotted Channel Hopping
PER	Packet Error Rate
TDMA	Time Division Multiple Access
GA	Genetic Algorithm
PSO	particle swarm optimization
QF	QueueFrame

References

1. Gershenfeld, N.; Krikorian, R.; Cohen, D. The Internet of Things. *Sci. Am.* **2004**, *291*, 76–81. [[CrossRef](#)] [[PubMed](#)]
2. ITU Internet Reports. *The Internet of Things*, 7th ed.; ITU: Geneva, Switzerland. Available online: www.itu.int/internetofthings/on (accessed on 28 December 2018).
3. Holler, J.; Tsiatsis, V.; Mulligan, C.; Karnouskos, S.; Avesand, S.; Boyle, D. *From Machine-to-Machine to Internet of Things: Introduction to a New Age of Intelligence*; Elsevier: Amsterdam, The Netherlands, 2014; ISBN 9780124076846.
4. Fortino, G.; Savaglio, C.; Zhou, M. Toward opportunistic services for the industrial Internet of Things. In Proceedings of the 13th IEEE Conference on Automation Science and Engineering (CASE), Xi'an, China, 20–23 August 2017; pp. 825–830.
5. Andrei, G.; Madhusanka, L.; Dmitry, K. Secure Communication and Data Processing Challenges in the Industrial Internet. *Baltic J. Mod. Comput.* **2016**, *4*, 1058–1073.
6. Hansson, A.; Khodari, M.; Gurtov, A. Analyzing Internet-connected industrial equipment. In Proceedings of ICSigSys'18, Bali, Indonesia, 1–3 May 2018.
7. Gaushell, D.J.; Darlington, H.T. Supervisory control and data acquisition. *Proc. IEEE* **1987**, *12*, 1645–1658. [[CrossRef](#)]
8. International Electrotechnical Commission (IEC). *Electricity Metering—Data Exchange for Meter Reading, Tariff and Load Control—Part 53-Part 53: COSEM Application Layer*; IEC: Geneva, Switzerland, 2006.
9. Modbus. Available online: <http://www.modbus.org/> (accessed on 28 December 2018).
10. Molina, B.; Palau, C.E.; Fortino, G.; Guerrieri, A.; Savaglio, C. Empowering smart cities through interoperable Sensor Network Enablers. In Proceedings of the 2014 IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA, USA, 5–8 October 2014.
11. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [[CrossRef](#)]
12. International Electrotechnical Commission (IEC). *IEC 62591. Ed 1.0: Industrial Communication Networks—Wireless Communication Network and Communication profiles—WirelessHART*; IEC: Geneva, Switzerland, 2010.

13. ISA100. Wireless Systems for Automation. Available online: <http://www.isa.org/Community/SP100WirelessSystemsforAutomation> (accessed on 28 December 2018).
14. Institute of Electrical and Electronics Engineers (IEEE). 802. 15.4e-2012: *IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer IEEE Std*; IEEE: Piscataway, NJ, USA, 2012.
15. Igarashi, Y.; Matsuura, Y.; Koizumi, M.; Wakamiya, N. Priority-Based Dynamic Multichannel Transmission Scheme for Industrial Wireless Networks. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 9124558. [CrossRef]
16. Zhiming, Z.; Martin, P.; Jones, A.; Taylor, I.; Stankovski, V.; Salado, G.F.; Suciu, G.; Ulisses, A.; de Laat, C. Developing, Provisioning and Controlling Time Critical Applications in Cloud. In Proceedings of the European Conference on Service-Oriented and Cloud Computing, Oslo, Norway, 27–29 September 2017; pp. 169–174.
17. Jetmir, H.; Poorter, E.D.; Moerman, I.; Hoebeke, J. A Survey of LoRaWAN for IoT: From Technology to Application. *Sensors* **2018**, *18*, 3995. [CrossRef]
18. Watt, S.T.; Loehner, H.; Achanta, S.V.; Kivi, A.; Rowland, B. Extending SCADA Networks Using Wireless Communications. In Proceedings of the Power Energy Automation Conference, Spokane, WA, USA, 10–12 March 2015.
19. SmartMesh IP. Available online: <http://www.analog.com/en/products/rf-microwave/wireless-sensor-networks/smartmesh-ip.html> (accessed on 28 December 2018).
20. Raja, P.; Noubir, G. Static and dynamic polling mechanisms for fieldbus networks. *ACM SIGOPS Oper. Syst. Rev.* **1993**, *27*, 34–45. [CrossRef]
21. Cavalieri, S.; Monforte, S.; Corsaro, A.; Scapellato, G. Multicycle Polling Scheduling Algorithms for FieldBus Networks. *Real-Time Syst.* **2003**, *25*, 157–185. [CrossRef]
22. Wang, W.; Wang, Y.; Li, X.; Song, W.; Frieder, O. Efficient interference-aware TDMA link scheduling for static wireless networks. In Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, Los Angeles, CA, USA, 23–29 September 2006; pp. 262–273.
23. Saifullah, A.; Xu, Y.; Lu, C.; Chen, Y. Real-Time Scheduling for WirelessHART Networks. In Proceedings of the 2010 31st IEEE Real-Time Systems Symposium (RTSS '10), San Diego, CA, USA, 30 November–3 December 2010; pp. 150–159.
24. Chipara, O.; Lu, C.; Roman, G.-C. Real-Time Query Scheduling for Wireless Sensor Networks. In Proceedings of the 28th IEEE International Real-Time Systems Symposium, Tucson, AZ, USA, 3–6 December 2007.
25. Zhijun, Y.; Zhu, L.; Ding, H.; Guan, Z. A priority-based parallel schedule polling mac for wireless sensor networks. *J. Commun.* **2016**, *11*, 792–797.
26. Mitchell, M. *An Introduction to Genetic Algorithms*; MIT Press: Cambridge, MA, USA, 1996; ISBN 9780585030944.
27. Pogorilyy, S.D.; Bilous, R.V. Genetic Algorithm for Network Performance Optimization. *Proc. IAM* **2012**, *1*, 121–128.
28. Sun, X.; Wang, J.; Wu, W.; Liu, W. Genetic Algorithm for Optimizing Routing Design and Fleet Allocation of Freeway Service Overlapping Patrol. *Sustainability* **2018**, *10*, 4120. [CrossRef]
29. Yi, J.; Clausen, T.; Igarashi, Y. Evaluation of routing protocol for low power and lossy networks: LOADng and RPL. In Proceedings of the IEEE Conference on Wireless Sensors, Kuching, Malaysia, 2–4 December 2013.
30. Abuagoub, A.M.A. An Overview of Industrial Wireless Sensor Networks. *Int. J. Comput. Sci. Inf. Technol. Res.* **2016**, *4*, 68–80.