Multi-Slot Allocation Protocols for Massive IoT Devices with Small-Size Uploading Data

Tsung-Yen Chan, Yi Ren, Yu-Chee Tseng, and Jyh-Cheng Chen

Abstract—The emergence of Internet of Things (IoT) applications introduces new challenges such as massive connectivity and small data transmission. In traditional data transmission protocols, an ID (i.e., IP address or MAC address) is usually included in a packet so that its receiver is able to know who sent the packet. However, this introduces the big head-small body problem for light payload. To address this problem, the Hint protocols have been proposed. The main idea is to "encode" information in a tiny broadcast Hint message that allows devices to "decode" their transmission slots. Thus, it can significantly reduce transmission and contention overheads. In this paper, we extend eHint to support multi-slot data transmissions. Several efficient protocols are proposed. Our simulation results validate that the protocols can significantly increase the number of successfully transmitted devices, channel utilization, and payload of transmitted devices compared with eHint.

Index Terms—Internet of Things (IoT), Machine-to-Machine (M2M) Communication, Multi-Slot Allocation, Random Access, Wireless Networks.

I. INTRODUCTION

INTERNET of Things (IoT) traffic characterized by small data transmission introduces new challenges to research community. There are various types of IoT devices ranging from small tags, sensors, to complicated actuators and machines [1]. Statistics show that 50% of IoT packets are less than 100 bytes [2]. Collecting small data from such massively connected IoT devices introduces new challenges. In most protocol designs, an ID (e.g., IP address or MAC address) is included in a packet so that its receiver is able to identify the sender. When IoT data is small, however, the overhead of its ID becomes relatively large, leading to the big head-small body problem.

Connection-oriented network architectures have been studied in [3]–[6]. When a device intends to send data to the Base Station (BS), a connection is established by using the Random Access (RA) procedure. As the number of devices increases, it may cause significant collision in RA [7], leading to a long delay. To solve the problem, several solutions have been proposed, including Access Class Barring (ACB) [3], dynamic resource allocation [4], slotted access [5], deep learning scheduling [6]. The idea of ACB is to control the number of User Equipment (UE) terminals intending to join RA. The BS broadcasts a parameter (e.g., probability) such that UEs can perform RA probabilistically. In dynamic resource allocation, the BS can dynamically allocate resources of RA channel and data channel. It also derives an optimal trade-off problem to maximize the Machine-to-Machine (M2M) throughput. In slotted access, each Machine-Type Communication (MTC) device is assigned a dedicated RA slot to access RA. The dilemma is that short RA cycles may lead to collision while long RA cycles may lead to long delay.

To address the big head-small body problem, a series of Hint protocols [8]–[10], which remove IDs from data packets, have been proposed. In these protocols, a tiny Hint message is broadcast to allow IoT devices to decode their assigned transmission resources. Interestingly, the assigned location of these resources also imply the sender’s ID, thus eliminating a large part of packet header. Therefore, even the device’s ID is not transmitted, its receiver (usually a BS) is still able to know its identity. In [8], we proposed a set of Hint-based frameworks for small data transmission. Later, a Chinese remainder theorem based Hint protocol [9] is applied to LTE-A networks to reduce the signaling cost in random access procedures. However, the both Hint protocols [8], [9] are based on a strong assumption that the small data has the same size and is carried by the same size channel resource, i.e., a time slot. To release the assumption, the Hint protocols are then enhanced by supporting multi-slot data transmission in [10]. However, when the number of transmitting devices is large, the multi-slot Hint protocol requires intensive computation capacity to find a satisfactory seed, which may lead to transmission latency. To address this issue, in this paper we use a novel iterative approach to reduce computation overhead and enable more devices to transmit. Through extensive simulations, we demonstrate that the proposed iterative approach significantly outperforms eHint [10] in terms of the number of successfully transmitted devices, channel utilization, and the total payload of transmitted devices.

II. MULTI-SLOT ALLOCATION PROBLEM

We consider a set $D = \{d_1, d_2, \cdots, d_m\}$ of $m$ IoT devices covered by a BS. Each IoT device $d_i, i = 1..m$, needs to report its data at a regular pattern to the BS. Our goal is to allocate radio resources to $D$ to transmit data to the BS. We make the following assumptions:

1) The value of $m$ is quite large.
2) Each $d_i$ switches between two modes, active and sleep. When intending to transmit data, $d_i$ goes to the active mode; otherwise, it switches to the sleep mode.
3) The active pattern of $d_i$ is denoted by a binary periodical function $P_i(t)$, where $t$ is time (by frame count). $P_i(t) = 1$ if $d_i$ intends to transmit data at the $t$-th frame; otherwise, $P_i(t) = 0$. For example, if $d_i$ is attached to a temperature sensor which needs to report in every 3
minutes, $P_t(t)$ has a period of 3 minutes. If one more humidity sensor which needs to report in every 5 minutes is attached to $d_i$, $P_t(t)$ is a function of the combination of two periodical functions with periods = 3 and 5 minutes, respectively.

4) Whenever $P_t(t) = 1$, device $d_i$ needs to send $n_t$ slots of data to the BS at $t$. Note that $n_t$ is also quite small (such as less than 3 or 5 slots).

5) When our Hint protocol starts, the BS is informed of its $P_t(t)$ and $n_t$ for each $d_i \in D$.

To solve the multi-slot allocation problem, the communication channel is divided into a sequence of fixed-length frames. Each frame consists of three parts: (1) Broadcast (Bcast): It is for the BS to broadcast and announce resource allocation information (i.e., Hint) to devices. (2) Allocated (Alloc): It consists of multiple slots for uplink data transmission (no transmitter’s ID required in our case). (3) Random (Rand): It is for any unscheduled/unexpected transmission not arranged in Alloc. Our goal is to design an efficient access protocol for such transmission.

Fig. 1 shows the frame structure. In this example, the active periods of $d_1$, $d_2$, and $d_3$ are $T_1$, $T_2$, and $T_3$, respectively. Their requirements are $n_1 = 2$, $n_2 = 1$, and $n_3 = 3$ slots, respectively. At frame $t$, all devices will transmit. At $t + 1$, only $d_1$ will transmit. At $t + 2$, $d_1$ and $d_2$ will transmit. The BS will schedule their transmissions in Alloc, through the announcement in Bcast. Details will be discussed later. For exceptions (such as transmission errors or emergency traffic), devices can use Rand.

III. PROPOSED PROTOCOLS

The main idea of the Hint protocol is to arrange a common function shared by the BS and all devices. The function takes two inputs: (i) a small piece of information computed by the BS, and (ii) a device’s ID. The BS will broadcast the information computed in (i) by using Bcast. Each device can use the broadcast information and its ID to retrieve the transmission slots allocated to it through the function. The broadcast Hint is thus more efficient than typical 1-by-1 transmissions in Alloc_1 and Alloc_2, and uses two seeds $s_1$ and $s_2$ to achieve this goal. Two vectors $v_1$ and $v_2$ will be used. Also, the length of $v_1$ and $v_2$ are set to $|v_1|$ and $|v_2|$, respectively, where $|v_1| = \sum_{d_i \in M(t)} n_i$ and $|v_2| = \sum_{d_i \in M(t) - M'} n_i$.

It works as follows:

1) The BS repeats the following steps a few times.
   - Randomly pick a seed $s_1$ and compute $h(s_1, d_i) \mod |v_1|$ for each device $d_i \in M(t)$.
   - Select a subset $M' \subseteq M(t)$ of devices such that only devices in $M'$ can correctly decode $v_1$ for safe transmission.
   - A seed $s_2$ is randomly picked. Compute $h(s_2, d_i) \mod |v_2|$ for each device $d_i \in M(t) - M'$.
   - Select a subset $M'' \subseteq M(t) - M'$ of devices such that only devices in $M''$ can correctly decode $v_2$ for safe transmission.

2) The BS then selects the $s_1$ and $s_2$ from the iteration in Step 1, leading to the largest number of safe transmission slots. (Note: once $s_1$ and $s_2$ are selected, the corresponding $M'$ and $M''$ are also selected.)

3) $v_1$ and $v_2$ are encoded as follows:
   - (Collision-free) For each $d_i \in M'$ such that $k = h(s_1, d_i) \mod |v_1|$, set $v_1[k] = 1$ and $v_1[k + 1:$
Suppose that $d_1, d_2, \ldots, d_{n_t-1}$ \in $M'$ such that $x = h(s_2, d_i) \mod |v_2|$, set $v_2[x] = \lceil 1 \rceil$ and $v_2[x + 1 : x + n_t - 1] = \lceil 2 \cdots 2 \rceil$. 

- (Empty/Collision) The rest of elements of $v_1$ and $v_2$ are all set to “0”.

4) The BS broadcasts $<s_1, v_1, s_2, v_2>$ in Bcast to all devices.

5) For each device $d_i \in M(t)$, receiving $<s_1, v_1, s_2, v_2>$ in Bcast, $d_i$ computes $k = h(s_1, d_i) \mod |v_1|$ and $x = h(s_2, d_i) \mod |v_2|$. To check whether $d_i$ is allowed to transmit or not, there are two cases:

- Case of $n_t = 1$: If $v_1[k] = \lceil 1 \rceil$ and $v_1[k + 1] = \lceil 0 \rceil$ or \lceil 1 \rceil, $d_i$ is allowed to transmit; otherwise, it checks $v_2$. If $v_2[x] = \lceil 1 \rceil$ and $v_2[x + 1] = \lceil 0 \rceil$ or \lceil 1 \rceil, $d_i$ is allowed to transmit; otherwise, $d_i$ cannot transmit.

- Case of $n_t \geq 2$: If $v_1[k] = \lceil 1 \rceil$, $v_1[k + 1 : k + n_t - 1] = \lceil 2 \cdots 2 \rceil$, and $v_1[k + n_t] = \lceil 0 \rceil$ or \lceil 1 \rceil, $d_i$ is allowed to transmit; otherwise, it checks $v_2$. If $v_2[x] = \lceil 1 \rceil$, $v_2[x + 1 : x + n_t - 1] = \lceil 2 \cdots 2 \rceil$, and $v_2[x + n_t] = \lceil 0 \rceil$ or \lceil 1 \rceil, $d_i$ is allowed to transmit; otherwise, it cannot transmit.

6) For each device $d_i \in M(t)$ that is allowed to transmit safely in $v_1$, $d_i$ uploads its data at $\text{Alloc}_{j : j + n_t - 1}$, where $j$ is the number of 1’s and 2’s in $v_1[0 : k - 1]$ and $k = h(s_1, d_i) \mod |v_1|$. If $d_i$ is not allowed to transmit in $v_1$, but is allowed to transmit in $v_2$, $d_i$ uploads its data at $\text{Alloc}_{z : z + n_t - 1}$, where $z$ is the number of 1’s and 2’s in $v_1$ and in $v_2[0 : x - 1]$ and $x = h(s_2, d_i) \mod |v_2|$. If $d_i$ is not allowed to transmit in $v_2$, it may consider to contend for transmission in $\text{Rand}$.

Fig. 2(a) shows the message flow of 2VF. As shown in Fig. 2(b), there are 7 devices intending to transmit in frame $t$. Hence, $|v_1| = n_1 + n_2 + n_3 + n_4 + n_5 + n_6 + n_7 = 13$. Suppose that devices $d_1$, $d_4$, and $d_7$ are selected to transmit in $v_1$ (i.e., $M' = \{d_1, d_4, d_7\}$). Then $v_1 = \{012201201200\}$. Although $d_2$, $d_3$, $d_5$, and $d_6$ are not allowed to transmit in $v_1$, they can perform VF again. Hence, $|v_2| = 3 = \{01212121212\}$. Although $d_2$, $d_3$, $d_5$, and $d_6$ are not allowed to transmit in $v_2$, they can perform VF again. Hence, $|v_2| = 3 = \{01212121212\}$. Although $d_2$, $d_3$, $d_5$, and $d_6$ are not allowed to transmit in $v_2$, they can perform VF again. Hence, $|v_2| = 3 = \{01212121212\}$. Although $d_2$, $d_3$, $d_5$, and $d_6$ are not allowed to transmit in $v_2$, they can perform VF again. Hence, $|v_2| = 3 = \{01212121212\}$. Although $d_2$, $d_3$, $d_5$, and $d_6$ are not allowed to transmit in $v_2$, they can perform VF again. Hence, $|v_2| = 3 = \{01212121212\}$. Although $d_2$, $d_3$, $d_5$, and $d_6$ are not allowed to transmit in $v_2$, they can perform VF again. Hence, $|v_2| = 3 = \{01212121212\}$. Although $d_2$, $d_3$, $d_5$, and $d_6$ are not allowed to transmit in $v_2$, they can perform VF again. Hence, $|v_2| = 3 = \{01212121212\}$.

\[ h(s, ID) \mod |w| \text{ or } |w| \]

Fig. 2: (a) The message flow of the 2VF. (b) An example of 2VF.

Fig. 3: Alloc vector of IVF.

its limitation. There are two main concepts in IVF (refer to Fig. 3):

- Iterative: The VF process is executed $l$ times. The $k$-th iteration takes the devices unable to transmit in the previous iterations and tries to find a seed $s_k$, where $k = 1, 2, \ldots, l$.

- Time-bound: A timer $T_k$ is set for iteration $k$ such that the search for $s_k$ would stop even if a satisfactory $s_k$ cannot be found after this time bound. In this case, the best $s_k$ so far will be used. Note that this only sacrifices the transmission ratio, but our protocol still works correctly.

We summarize how IVF works as follows:

1) The BS computes $\text{Alloc}_{k}$ for $k = 1, 2, \ldots, l$ as follows. In iteration $k$, we define the length of vector $v_k$ as

\[
|v_k| \geq \begin{cases} 
\sum_{d_i \in M(t)} n_i & \text{if } k = 1 \\
\sum_{d_i \in M(t) - \bigcup_{k=1}^{l-1} M_k} n_i & \text{if } k = 2, 3, \ldots, l 
\end{cases}
\]

where $M_k$ is the set of devices that can transmit in $\text{Alloc}_{k}$ in iteration $k$. The BS repeatedly chooses a
seed $s_k$ and computes $v_k$ until (i) a satisfactory seed $s_k$ is found, or (ii) the time bound $T_k$ spent on the search has reached. In the later case, the best seed $s_k$ so far and the corresponding $v_k$ is selected. After these $l$ iterations, the BS broadcasts $\langle \{s_k, v_k\}_{k=1}^l \rangle$ in $\text{Beast}$.

2) Upon receiving $\text{Beast}$, a device $d_i$ with $P_i(t) = 1$ can transmit in one of the $l$ spaces ($\text{Alloc}_1, \text{Alloc}_2, \ldots, \text{Alloc}_l$) according to its hash results $h(s_k, d_i)$ and vectors $v_k$. If it is not allowed to transmit in $\text{Alloc}$, it can contend in $\text{Rand}$.

IV. PERFORMANCE EVALUATION

We evaluate 2VF and IVF through simulations in terms of three metrics, the number of successfully transmitted devices in $M(t)$, the length of $\text{Alloc}$ (i.e., $|\text{Alloc}|$), and Channel Utilization ($CU$). The results are compared with VF and traditional polling protocol which collects data from each device one-by-one. Each slot size is assumed to be 32 bits and the length of $s$ is set to 16 bits. Device address is assumed to be 64 bits. The value of $n_i$ is small and it randomly falls in the range of $[1 \sim 3]$ or $[1 \sim 5]$ slots.

The $CU$ is defined as: $CU = \frac{\text{payload}}{\text{payload} + \text{packet header}}$. For 2VF and IVF, $CU = \frac{|\text{Alloc}| + 2\times 16 + 2\times |v| + |v'|}{|\text{Alloc}|}$ and $CU = \frac{|\text{Alloc}| + 16 + 2\times |v|}{|\text{Alloc}|}$ for traditional polling protocols. $CU = \frac{|\text{Alloc}| + 16 + 2\times |v|}{|\text{Alloc}|}$ for IVF, we execute VF until all the devices are allocated in $\text{Alloc}$ successfully.

Fig. 4(a) and Fig. 4(b) demonstrate the number of successfully transmitted devices in $M(t)$ with $n_1 = [1 \sim 3]$ and $[1 \sim 5]$, respectively. Both the figures show that IVF and 2VF outperform VF significantly. Considering Fig. 4(b) as an example, when $|M(t)| = 200$, IVF and 2VF gain 44% and 30% transmitted devices compared with VF, respectively.

Fig. 4(c) and Fig. 4(d) demonstrate the impacts of $|M(t)|$ on $|\text{Alloc}|$ by varying $n_1 = [1 \sim 3]$ and $[1 \sim 5]$, respectively. We observe that both 2VF and IVF outperform VF with large margins, and IVF performs the best. Specifically, $|\text{Alloc}|$ increases as the number of devices intending to transmit grows. We also observe that the three curves are linear. This means that the proposed protocols are scalable and resilient to the increasing of $|M(t)|$.

Fig. 4(e) and Fig. 4(f) show the impacts of $|M(t)|$ on $CU$ when $n_1 = [1 \sim 3]$ and $[1 \sim 5]$, respectively. Overall, the Hint protocols (i.e., VF, 2VF, and IVF) outperform the traditional protocol significantly since we use a Hint message instead of notifying devices by $64 \times |M(t)|$ times. The $CU$ of the traditional scheme is around 0.5 and 0.6 in Fig. 4(e) and Fig. 4(f), respectively, when $|M(t)|$ increases. The traditional protocol with $[1 \sim 3]$ has higher $CU$ than $[1 \sim 5]$ since $[1 \sim 5]$ has more payload to transmit. The values of $CU$ gradually converge for VF, 2VF, and IVF with the increase of $|M(t)|$.

Overall, Figs. 4(a)-(f) demonstrate that the iteration based Hint protocol has much better performance compared with [10] in terms of the number of successfully transmitted devices, the total payload of transmitted devices, and channel utilization.

V. CONCLUSIONS

In this paper, we propose two advanced Hint protocols, 2VF and IVF, which enable more devices to upload their data in a collision-free manner. Compared to the eHint [10], the proposed 2VF and IVF provide more flexibility and address the issue when the number of devices intending to transmit is large. We demonstrate, through extensive simulations, that the proposed 2VF and IVF outperform both eHint [10] and traditional polling protocol in terms of $CU$ and slot allocation capability. As to future work, designing a smart seed with low computation cost is worth of exploring.

REFERENCES


